Knowledge-Based Image Coding Technique

Adnan M. Alattar

Center for Communication and Signal Processing
Department of Electrical and Computer Engineering
North Carolina State University
Raleigh, North Carolina
November 1989

CCSP TR-89/24

ABSTRACT

Alattar, Adnan Mohammed. Knowledge-Based Image Coding.
(Under the direction of Sarah A. Rajala)

A new image coding technique for producing good image quality at a very high compression ratio is developed. The new technique is best suited for coding teleconference and Picture Phone images, which require a compression ratio on the order of 1000:1 when transmitted through a low-bit-rate channel of 64 Kbits/s. The main idea behind this new technique is to break the image into complicated primitives using a priori knowledge about the image. These primitives are not necessarily coded and transmitted. Instead, they are matched to a previously created database, and the necessary information about the best match is coded and transmitted. This information is used at the receiver to retrieve a replica of the original primitive from a duplicate database. The decoded primitives are eventually assembled, as in computer animation, to produce a faithful reconstruction of the original image. Basically, the coder consists of a primitive extractor followed by a primitive matcher that interacts with a database to produce a sequence of messages. The database is a collection of the primitives typically found in the images for a certain application. The messages are coded and transmitted through the channel. The primitive extractor also interacts with a fact base which contains a priori information about the image. The quality of the images produced from this technique is very good at extremely high compression ratios (over 1000:1).

BIOGRAPHY

Adnan Mohammed Alattar was born in Khan Younis, Palestine, on November 25, 1961. He received his B.S.E.E. from the University of Arkansas at Fayetteville in 1984 and his M.S.E.E from North Carolina State University at Raleigh in 1985.

During his graduate studies at North Carolina State University, he was employed as a research assistant in the Center for Communications and Signal Processing (CCSP) and as a teaching assistant in the Department of Electrical and Computer Engineering.

Adnan Mohammed Alattar is a member of the IEEE.

ACKNOWLEDGEMENTS

The author is grateful to his advisor Dr. S. A. Rajala for her guidance and support at all stages of this research. He would also like to thank Miss Marcie L. Lucas for editing this manuscript.

The author also wishes to thank his parents, brothers, and sisters for their encouragement, patience, and support during the course of this study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Although digital representation of pictures has several advantageous features such as processing flexibility, relatively errorless transmission, ease of storage and retrieval, and compatibility with digital computers and digital networks, it has the disadvantage of requiring a very large number of bits. For instance, over 2 million bits are needed to represent a picture with 512x512 pixels when each pixel is quantized to one of 256 gray levels. Transmitting or storing such a large amount of data requires a large bandwidth or large storage space, which may not be available. The consequence of these two requirements is the high cost involved. Fortunately, image data redundancy can be reduced before transmission or storage, allowing reductions in the bandwidth and storage requirements.

In the last two decades, numerous image coding techniques have been developed in order to reduce the image data redundancy [1,2,3,4]. As shown in Fig. 1.1.a, an image coding technique can be viewed as a two-step process involving a message extractor and a message coder. In the first step, certain messages are extracted from the input image, and in the second step, the messages are coded using a source-coding scheme. Early image coding techniques emphasize message coding over message extraction, while recent techniques emphasize both steps. Based on this distinction, it is natural to classify image coding techniques as either first or second generation.

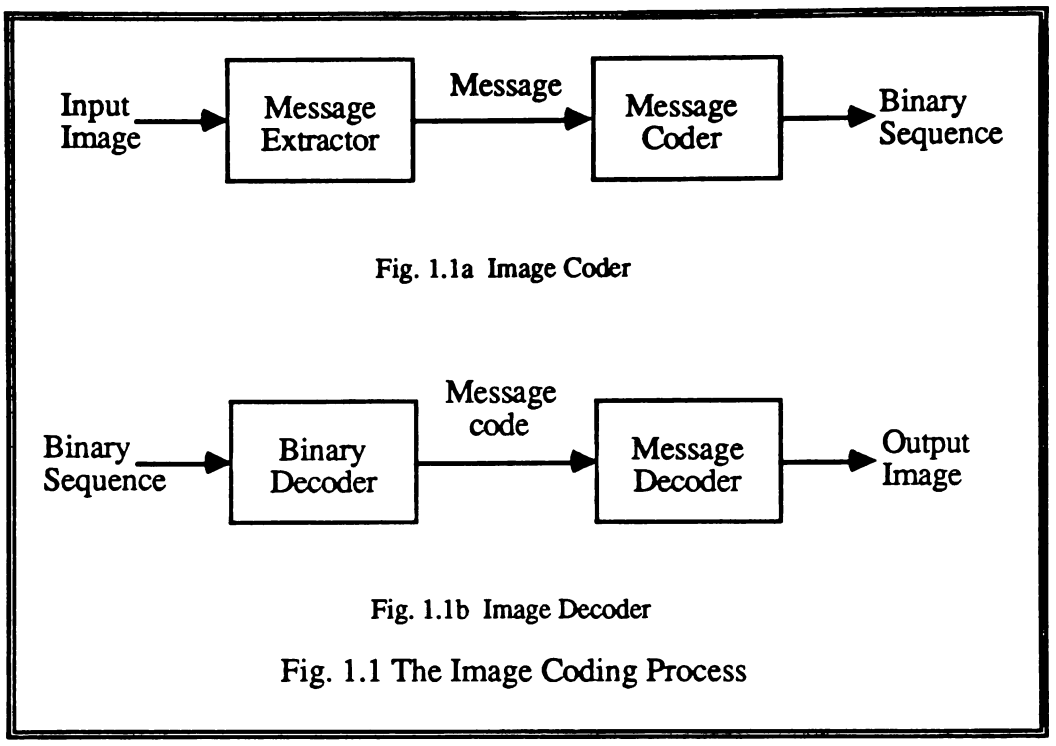


Fig. 1.1 The Image Coding Process

Other factors further distinguish between first and second generation techniques. First generation techniques consider the individual pixel values as the messages [1,2]. These values are algebraically manipulated to reduce the statistical correlation between them, and are then coded and transmitted. Popular coding techniques such as pulse coded modulation (PCM), transform coding, and predictive coding are in this category. Second generation techniques [5], however, use global image features as messages, and therefore achieve higher compression ratios than those achieved with first generation techniques. Segmentation-based image coding [3,5,6] is an example of a popular and advanced second generation technique.

Generally speaking, first generation techniques produce better image quality [1,2] than second generation techniques, but second generation techniques achieve higher compression ratios [5,6]. While first generation techniques achieve a maximum compression ratio of less than 20:1, second generation techniques achieve compression ratios on the order of 50:1. These compression ratios are still insufficient for most modern applications. For instance, digital transmission of monochrome video signals over a low-bit-rate channel of 64 Kbits/sec or less requires a compression ratio over 750:1 [7]. When attempting to achieve such a compression ratio with existing image coding techniques, unacceptable image quality may be obtained.

In this research effort, a new image coding technique is developed, which achieves an extremely high compression ratio while producing good image quality. The new technique is called knowledge-based image coding, and may be thought of as a generalization of segmentation-based techniques. In this new technique, a priori knowledge about the image is used to break the image into its primitives, which are chosen to represent the features of the image. Although, such a priori knowledge is available for most applications, its use has long been overlooked. Diverse fields such as computer animation, the art of drawing, expert systems, and database systems are incorporated into both the coding and decoding stages of the new technique, as discussed in chapter 2.

The rest of this chapter is a brief review of the major traditional and second generation image coding techniques.

1.1. Traditional Image Coding

Most traditional image coding techniques are based on information theoretic principles, and usually depend on local, pixel-oriented features of the image. The goal of these techniques is to exploit the statistical redundancy among the pixels in order to achieve data compression. Five distinct types can be identified among traditional image coding techniques, namely, pulse coded modulation, transform coding, predictive coding, interpolative and extrapolative coding, and binary coding [1,2,3,4]. These are discussed in the following five subsections.

1.1.1. Pulse Coded Modulation

Pulse Coded Modulation (PCM) is the most basic type of image coding [8]. It is a by product of representing analog images in digital format. In this technique, an analog image signal is spatially filtered and sampled on a rectangular grid, which is typically 512x512. Then, the signal amplitude is quantized and assigned a code word of fixed or variable length, usually 8 bits. At the receiver, the received code word is decoded to reconstruct the original amplitude.

The quantization of the signal amplitude introduces a quantization error, which can be reduced by careful design of the quantizer. For instance, if the probability density function of the input signal is known, then an optimum quantizer can be designed [9]. However, if the input signal is non-stationary, an adaptive quantizer that adapts to the statistical properties of the input signal may be designed. Although acceptable image

quality can be obtained using 3 bits/pixel, more than 5 bits/pixel are usually used to adequately reduce the quantization noise, which manifests itself in the image as snow.

Vector Quantization (VQ) [10] may be considered a generalization of PCM in which adjacent pixels are grouped into vectors to serve as the messages to be coded and transmitted. The pixels in each vector are quantized together. Usually, the quantization process is done by replacing the vector with a suitable choice from a pre-existing set of quantized vectors; this set is called the codebook. The design of an optimum codebook is not an easy task. Usually, a training set of images is used to generate vectors suitable for the codebook. Suitable vectors are those that minimize the average quantization error in the class of images to be coded. One method of generating the codebook is called the Lloyd algorithm [11]. It requires an initial codebook, which is iteratively updated until no significant reduction in the average quantization error is obtained. A more efficient algorithm, which does not need an initial codebook, is called the nearest neighbor algorithm [12]. It starts with the entire training set of vectors and reduces it to the desired size by iteratively merging pairs of closely-valued vectors into single vectors. The value of each new vector is equal to the mean of the vectors in the merged pair.

The major advantage of VQ is the simple structure of the receiver, which consists only of a duplicate codebook. The disadvantages include the design complexity of an optimum code book and the fact that images dissimilar to those in the training set may not be well-represented by the code vectors in the codebook.

1.1.2. Transform Coding

In transform coding [13], a linear transformation is used to transform an input image to another domain, where the correlation among the transformation coefficients is less than that among the original pixels, and where the energy is concentrated in a few of the coefficients. The values of the high-energy coefficients are quantized and coded as messages for transmission; the other coefficients are discarded. At the receiver, the inverse transform is performed on the received coefficients to reconstruct a replica of the original image. However, the quantization error and the loss of energy in the discarded coefficients produce image distortion.

Zonal sampling and threshold sampling [1] are used to select the high energy coefficients. In zonal sampling, the coefficients in a specific zone are selected, regardless of their energy. In threshold sampling, only those coefficients with magnitudes above a certain threshold are selected. The performance of threshold sampling is better than that of zonal sampling, but because of the overhead information regarding the location of the selected coefficients in threshold sampling, more compression is achieved with zonal sampling than with threshold sampling.

The efficiency of a transform coder depends on the efficiency of the particular transform in decorrelating the coefficients while concentrating the total energy into a few coefficients. The Karhunen-Loève (KL) transform is the optimal transform for a given class of images [14,15], but computationally unrealizable, since it requires knowledge of the autocorrelation matrix of the image source. Fourier [16], cosine (DCT) [17], and Hadamard [18] are examples of other unitary transforms commonly used in image transform coding. The performances of these transforms are suboptimal relative to the

performance of the KL transform. In fact, these transforms achieve a bit rate of slightly less than 1 bit/pixel, which can be improved 25% by using adaptive transform coding [19]. Adaptation in transform coding includes changing either the way the coefficients are quantized or the type of transform used.

A special type of transform coding is Singular-Value-Decomposition (SVD) [20]. This transform is optimum for a particular image. The vectors of the SVD of the input image are used as the transformation basis vectors. Since these vectors vary from one image to another, they have to be transmitted in addition to the transformation coefficients. This reduces the achievable compression ratio and the importance of the SVD as an image compression technique.

Transform coding can be considered a special case of a more general class called sub-band coding [21]. In sub-band coding, filter banks are used to divide the image into sub-bands that are quantized and coded separately. Normally, Quadrature Mirror Filters (QMF's) are used to divide the image into its sub-bands. The use of the QMF's eliminates possible aliasing error between the sub-bands. At the receiver, inverse filter banks are used to reconstruct the original image. The performance of the sub-band coder at a rate of 0.67-2.0 bits/pixel is shown in [21] to be better than that of the DCT and VQ coders with subjective error properties, and the complexity of the sub-band coder is comparable to that of transform coders.

1.1.3. Predictive Coding

The third class of traditional image coding techniques is predictive coding. It exploits the strong spatial and temporal correlation between adjacent image pixels to predict the value of the current pixel from the values of previously encoded pixels. The prediction error serves, in this case, as the message to be transmitted. Since the prediction error is statistically less correlated than the original pixels, and its effective dynamic range is smaller, it can be quantized more coarsely to achieve image compression.

Similarly, the prediction of the current pixel value from previously decoded pixels occurs at the receiver. The value of the current pixel is reconstructed by adding its transmitted error to its predicted value. In general, the prediction can be linear or non-linear, and one-dimensional or two-dimensional. One-dimensional prediction utilizes only those pixels in the same line as the pixel being predicted, while two-dimensional prediction utilizes pixels in the previous lines as well.

Delta Modulation (DM) [22] and Differential Pulse Code Modulation (DPCM) [23] are examples of two commonly used predictive coding techniques. Both techniques are based on linear prediction, but they differ in the number of bits used to quantize the prediction error. In DM, only two quantization levels are used, so it is necessary to increase the sampling rate to several times the Nyquist rate in order to produce an acceptable image quality. This reduces the importance of DM as an image compression scheme. On the other hand, by using more than two quantization levels, DPCM produces data rates around 2 bits/pixel with very good image quality. This number of bits/pixel may be decreased by using an adaptive predictor [24] and an adaptive quantizer [25,26], both of which change their characteristics as a function of the input data.

A higher compression ratio can be achieved for time-varying image sequences by using interframe prediction. Interframe prediction uses pixels from the current or previously transmitted fields or frames to predict the value of the current pixel. A special type of interframe predictive coder utilizes conditional replenishment [27]. The structure of this coder is very simple; the pixels in the current frame are subtracted from those in the previous frame and the difference is thresholded, quantized, and transmitted. When the motion in the image sequence is very low, the performance of conditional replenishment with 1 bit/pixel is comparable to that of an 8-bit PCM coder. However, the performance is seriously degraded when the motion is vigorous.

Some very important predictive coders are known in the literature as motion compensation coders. These coders are specially designed for coding time-varying image sequences, where motion between frames is assumed small and translational. When a non-compensated predictive coder is used between the frames of such sequence, a large prediction error occurs. Motion compensation coders reduce this error by first estimating a displacement vector for each pixel in the image, in order to find its corresponding location in the previous frame; they then predict the current pixel value based on the pixel values in a neighborhood of the corresponding location in the previous frame. There are two basic methods for estimating the displacement vector: Pel Recursion, and Block Matching [3]. In Pel Recursion the motion is estimated for each pixel based on the spatial and temporal gradients, but for Block Matching, the image is first divided into small blocks and a displacement vector for each block is estimated by finding the best match of that block in the previous frame.

Transform and predictive coders can be combined into hybrid coders to achieve high compression ratios at 0.5 bits/pixel. In these coders, the image is spatially transformed using a one-dimensional or two-dimensional transform coder, such as the discrete cosine transform. Then a predictive coder is used to reduce the correlation between the transformed coefficients from line to line or from one frame to another [28,29].

1.1.4. Interpolative and Extrapolative Coding

Interpolative and extrapolative coding techniques comprise the fourth class of traditional image coding. In these techniques, a subset of the image samples is selected and coded using transform or predictive coding. The sub-sampling can be performed over the spatial or the temporal domains, and the locations of the samples can be fixed or variable. If the locations are variable, then overhead information which indicates these locations is transmitted along with the samples' amplitudes. The decoder may use linear or non-linear interpolative/extrapolative techniques to fill in for missing samples [1,2,7,30].

The performance of the interpolative/extrapolative techniques depends upon the technique used for coding the selected samples, and upon the interpolative/extrapolative scheme used to fill in for the missing samples. Generally speaking, these techniques perform better than non-adaptive, but not as well as adaptive DPCM and transform coding [7]. Interpolative/extrapolative techniques achieve bit rates on the order of 2 bits/pixel, so they are very useful for applications requiring low bit rates, when it is not possible to code each sample.

1.1.5. Binary Coding

Although most of the previously discussed coding techniques are applicable to binary images, binary coding techniques, which usually rely on the presence of only black and white pixels, have been specially designed for binary images [4,31]. Binary coding techniques are classified as either information-lossy or information-lossless. Conceptually, these classes are the same, except that the lossy techniques are preceded by an image modification stage. In this stage, selected pixels are deleted from or added to the image in order to achieve a higher compression ratio. The modification stage is designed such that no visible image degradation is introduced.

Run length, Huffman, arithmetic, and block coding are examples of very popular binary image coding techniques [4,31]. In run length coding, a set of consecutive pixels of the same color, called a run, is considered as a message, and is coded by assigning it a code word based on its length and color. Run length coding can be easily extended to include two dimensional runs. Huffman coding is an efficient way of assigning variable-length code words, based on the probabilities of the lengths of the runs; for example, the most probable length is assigned the shortest code word and the least probable length is assigned the longest code word. The performance of Huffman coding is very close to the entropy of the source.

An arithmetic coder is a sequential coder in which one code word is assigned to the entire image [32]. This code represents a real number on the number line between 0 and 1. This number is obtained by successively dividing a sub-interval of the number line

between 0 and 1, and retaining one of the portions as the new interval. The division is based on the bit to be coded and the probabilities of 0 and 1 in the image. An image compression of 8:1[1] is achievable with an arithmetic coder.

Block coding is a special case of vector quantization for multilevel images. First, a code book is designed that contains a set of distinct two-dimensional binary blocks. Then the input image is divided into blocks of the same size as those in the codebook. Each of the image blocks is compared to the codebook blocks, and the code of the best match is transmitted. If there is no match, the bits in the block are transmitted, along with a flag to indicate this event. A compression ratio of 10:1 is achievable for sparse facsimile documents.

1.2. Second Generation Image Coding

As previously mentioned, second generation image coding techniques emphasize both message extraction and message coding to achieve extremely high compression ratios [5]. They emphasize the use of global image features, and employ symbolic representations of the image. They can be divided into three closely-related techniques: Perception-Based, Segmentation-Based, and Model-Based image coding techniques. These are discussed in the following three subsections.

---

[1] In this dissertation the mentioned compression ratios are directly related to number of bits per pixels and are independent of the image size. For instance, 8:1 compression ratio means that each pixel in the image is represented with 1 bit after compression, if it was originally represented with 8 bits.

1.2.1. Perception-Based Image Coding

Most perception-based image coding techniques are designed to exploit various properties of the human vision system (HVS). Since the HVS is not fully understood, only simple properties have been incorporated into traditional image coding techniques to improve their performance. For instance, filters with designs based on properties of the HVS are used in pre- and post-coding filtration, which is used in most traditional image coding techniques to eliminate visually displeasing artifacts. Various HVS-based adaptive and non-adaptive quantization algorithms have been devised for both predictive [33,34] and transform coding to reduce quantization error [35]. These algorithms are designed either to take advantage of human visibility thresholds or to minimize a visually-weighted mean-square-error. Also, the proposed CCITT (Comitè Consultatif International Tèlègraphique et Tèlèphonique) standard for quantizing the discrete cosine transform (DCT) has been experimentally designed to take advantage of properties of the HVS [1,2].

Some other perception-based image coding techniques have been fully designed based on properties of the HVS. In these techniques, the information that is relatively more important to the HVS is isolated and coded with high quality, while relatively less important information is coded with high compression. This is achieved in the pioneering Synthetic Highs image coder [36] by dividing the image into low- and high-pass signals that are coded independently. In a similar approach by Yan and Sakrison [37], the image is divided into edges and a residual image. The edges are assumed to present the most important information to the viewer, and are therefore coded with high quality, while the less important residual image is coded with high compression. Recently, Ikonomopoulos

and Kunt [5,38] have used a set of directional filters to decompose an input image into a low-pass image and a set of directional edge images. The low-pass image is coded using the Fourier transform, while the edge images are sub-sampled. A compression ratio of 30:1 has been achieved with their technique.

The Pyramidal Image coding technique [39] is similar in principle to the Synthetic Highs coding technique. In the Pyramidal Image coding technique, the image is successively filtered using unimodal Gaussian low-pass filters, each with a cutoff frequency approximately half that of its predecessor. An error signal is generated at each stage by subtracting the low-pass signal at that stage from the low-pass signal at the previous stage. Each error signal can be thought of as the result of convoluting the original image with two Gaussian-like functions which, in combination, are similar to the impulse response of the lateral inhibitation phenomenon of the human visual system. The resulting error sequence is quantized and coded. At the receiver, these errors are summed to form the original image. Good image quality at compression ratios on the order of 10:1 has been obtained with this technique.

A method called the Anisotropic Coding technique [40] has also been developed, which utilizes properties of the HVS in order to achieve high image compression. The basic idea of this technique is similar to that of predictive coding. A predictive filter, which is equal to the weighted sum of three Wiener filters, is used to predict the image. The characteristics of the three filter components and the weighting functions are highly related to the HVS characteristics. For instance, two of the filters are high- and low-pass isotropic Wiener filters. The combination of these two filters leads to a band-pass behavior similar to the behavior of the early stages of the human visual system. The third

component is an anisotropic Wiener filter that enhances local, rectilinear features of the image. The prediction error is coded using the DCT, and the weighting functions are sub-sampled. Compression ratios on the order of 35:1 are achievable with this technique, and good image quality is obtained.

1.2.2. Segmentation-Based Image Coding

One of the most popular approaches to second generation image coding techniques is the segmentation-based technique [3,5,6]. In this technique, a segmentation algorithm is used to partition the input image into disjoint regions, each of which is uniform and homogeneous with respect to certain characteristics. The segmentation process is carried out in three steps: preprocessing, region growing, and elimination of artifacts. Then, the contour and some information about the textural structure of each segment are coded and transmitted. At the receiver, this information is used to reconstruct the image by painting the textural structures inside the contours of their corresponding segments. Granularity in the form of pseudo-random noise is added to the reconstructed image to give it a more natural look.

One way to code the contours is to approximate them by circular and straight line segments [5]. Another way is to consider the contour image as a binary image and to use binary image coding techniques to code it. Constant intensity value or textural structure can be assumed for the interior of each segment. When constant intensity value is assumed, the intensity value of each segment must also be transmitted. However, when a textural structure is assumed, a two-dimensional polynomial is fit to the pixels' intensity

values inside the segment [5]. The coefficients of the polynomial are coded and transmitted to the receiver.

Fractals can also be used to code the interior of each segment. Barnsley and Sloan [41,42] have used a library of Iterated Function System (IFS) codes to achieve extremely high compression ratios. Each IFS code is a set of iterative affine transforms that are able to reproduce a fractal like image segment that approximates certain type of texture. Therefore, the texture inside an image segment is coded by first finding a suitable IFS code from the system's library, that is able to reproduce a similar texture at the receiver. The necessary information about the IFS function for each segment are then coded and transmitted instead of the texture itself. Although compression ratios in the order of 1000:1 are achieved using this technique, the produced images resemble impressionistic rendering of their originals, rather than photographic copies. Moreover, this technique requires intensive computations and processing time in the order of 100 hours/frame.

In segmentation-based image coding, the achievable compression ratio and the resulting image quality depend on the properties used to segment the image, and on the performance of the algorithm used to code the contours. The use of uniform and homogeneous properties of the segments has the disadvantage of producing a large number of image segments with boundaries that are not necessarily the contours of the objects in the image. Properties of the HVS, however, can be incorporated into the segmentation algorithm to reduce the number of segments. Rajala and Lee [6] have devised an algorithm based on the centroid linkage region growing technique and Weber's law to segment an image, and have used the arithmetic coder to code the

contours. They also have extended their algorithm to time-varying images, and have achieved a 50:1 compression ratio with visually pleasing images.

1.2.3. Model-Based Image Coding

Model-based image coding techniques stem from the image formation process. As depicted in Fig. 1.2, an image is formed by projecting a 3-D scene onto a 2-D plane using a lens. The resultant image is perceived by the viewer to represent the scene world. Similarly, in model-based image coding, a 3-D model for an object in an input image is first constructed from the object's basic properties, such as shape, surface color, and texture. These properties are transmitted to the receiver. At the receiver, a duplicate 3-D model is constructed using the transmitted properties of the object. The 3-D model is used to synthesize a 2-D projection which is similar to the original input image. To code a time-varying image sequence, motion parameters of the object in each frame are also estimated and then transmitted. Then, it is necessary to update the receiver with only the information necessary to reflect the local or global motion of the object on the image plane.

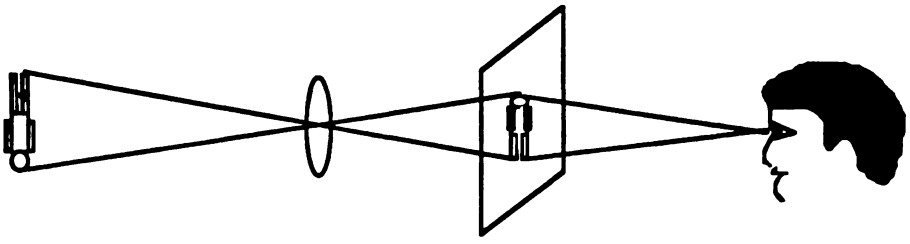


Fig. 1.2 Image Formation and Human Perception

Although the idea behind this technique is simple and intuitive, its feasibility and realization are quite difficult. Three major difficulties are encountered. The first is the 3-D modeling of an object from its 2-D projection; the second is incorporating the unique properties of the object into the 3-D model; and the third is the complexity of the computations required for these purposes. The general problem, however, can be reduced by restricting the input to a certain class of images. For instance, the input image can be restricted to the class of head and shoulders images, which are normally encountered in applications such as "face-to-face" telecommunications, which includes teleconference and picture phone services. Such applications require very high compression ratios, but have a high tolerance for degraded image quality. Therefore, the head and shoulders class of images lends itself to model-based image coding.

For this purpose, a pre-defined 3-D wire frame model with plane surfaces can be assumed for the head of the human being [43,44]. Several hundreds of polygons are used in the model as plane surfaces. These surfaces are rendered to give the model a natural

look. The model also includes several facial expression points that can be manipulated to create required facial expressions.

Several techniques have been proposed based on this pre-defined model [45,46,47,48,49], but limited success has been achieved. The British Telecom Research Laboratories (BTRL) [47], for instance, has simulated a system in which the information needed to recognize a subject is sent at the beginning of a conversation and used by the receiver to construct a model of the subject's head. At the receiver, the head is animated, based on transmitted codes that indicate the movements of the actual head and face. BTRL has demonstrated color moving images at a data rate of a few hundred bits/second. Forchheimer and Fahlander [48,49] also used a pre-defined model; however, their work was concentrated on solving problems such as global and local motion estimation and image synthesis. They devised an iterative algorithm to estimate the shape and motion parameters simultaneously. They also achieved real-time synthesis of images based on their assumed model. Nevertheless, functional model-based image coding system has not yet been fully developed. Research in this area is still in the infancy stages and many challenges remain.

1.3. Outline of the Dissertation

In this dissertation a knowledge-based image coding technique is developed in order to achieve extremely high compression ratios, while producing good image quality.

This dissertation is organized into seven chapters, starting with the introduction. Chapter (2) discusses an approach to knowledge-based image coding techniques. In

chapter (3), a model-based algorithm for facial feature extraction from photographs is developed. This algorithm is used by the knowledge-based image coding technique when applied to head and shoulders images. Chapter (4) discusses this application of the knowledge-based image coding to still head and shoulders images. Chapter (5) extends the application of the knowledge-based image coding technique to time-varying image sequences, similar to those encountered in teleconference and picture phone services. Chapter (6) covers the implementation, simulation results, and performance evaluation of the knowledge-based image coding technique. Chapter (7) contains the conclusion and final remarks.

CHAPTER 2

KNOWLEDGE-BASED IMAGE CODING

Image coding can be viewed as the process of encoding the relevant information in the image; one part of this information is embedded in the a priori knowledge about the class of images, while the other part must be coded and transmitted explicitly. The sum of this information is constant. Therefore, the more a priori knowledge that is included in the image coding system, the less information is transmitted to the receiver. Unfortunately, it is often difficult to represent the available a priori knowledge in a form that is compatible with the implementations of most coding processes. General a priori knowledge is not easily incorporated into the coding process.

Nevertheless, simple a priori knowledge has been used implicitly and heuristically in most existing image coding techniques to improve their performance and to achieve very high compression ratios. For instance, first generation image coding techniques, such as PCM, transform coding or predictive coding, incorporate a priori knowledge in the form of certain assumptions and constraints about the statistical properties of the class of images to be coded. These properties include the correlation among the pixels in the image, and the probability density function of the image source. Second generation techniques incorporate a priori knowledge about the human vision system (HVS), and about the type of images to be coded. For example, perception-based image coding techniques are based on a set of assumptions about the HVS, and model-based techniques are based on assumed 3-D models of the objects in the image.

In this chapter, a knowledge-based image coding system is developed, which utilizes high level knowledge, such as knowledge about the class of images to be coded and its formation and motion models. This requires a sophisticated design and explicit and implicit utilization of a priori knowledge in forms suitable for the particular image coding application. The structure of this coder is similar to that of an ordinary knowledge-based system. However, it also incorporate techniques from computer animation in order to manipulate the explicit a priori knowledge. Therefore, the relevant features of general knowledge-based systems and animation will be reviewed first, followed by a description of the proposed new knowledge-based image coding system.

2.1. General Knowledge-Based Systems

A general knowledge-based system is depicted in Fig. 2.1. It consists of four basic blocks: an environment/system interface, a system/environment interface, a knowledge base, and an inference engine. These four blocks are not always distinct; for instance, in some knowledge-based systems, the knowledge base is combined with the inference engine, and is not easily distinguished from the rest of the system. In fact, this is the case for most image analysis algorithms which use some form of knowledge or another.

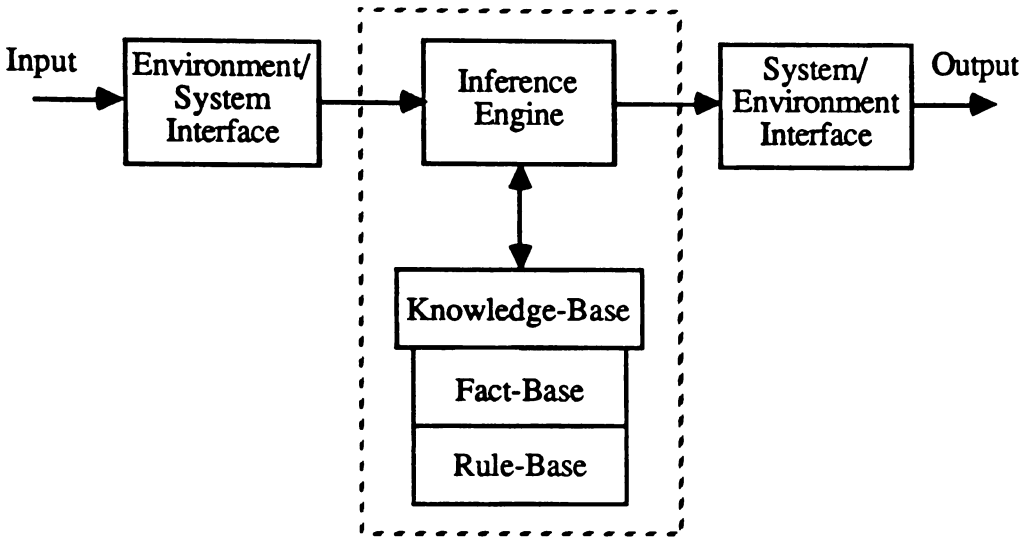


Fig. 2.1 Knowledge-Based System

2.1.1. Knowledge-Based System Interfaces

The environment/system and system/environment interfaces are very similar to the input and output units, respectively, of most modern computer systems. The environment/system interface is responsible for extracting information from the environment, while the system/environment interface is responsible for outputing the result in a form compatible with the environment, which can be a human user or a certain type of application. The physical structures of the interfaces depend on the type of application. For example, they can be the keyboard and the screen of an ordinary computer terminal, as in general applications, or a speech recognizer and a speech synthesizer as in a natural language processing system.

2.1.2. The Knowledge Base

The knowledge base is the most important part of a knowledge-based system. This unit forms the source of the system's of intelligence. That is, it is used by the inference engine to produce intelligent results. Usually, two types of knowledge are stored in the knowledge base: declarative and procedural knowledge. Declarative knowledge consists of facts about the objects, events and situations for a given field of application, while procedural knowledge includes heuristic rules and procedures for using the declarative knowledge. The two types of knowledge may be separated or integrated depending on the method used to represent the knowledge.

The knowledge in the knowledge base is represented in a form that allows reasoning and has the capability of naming, describing, relating, organizing and constraining a set of elementary concepts. Thus, the form of knowledge representation usually depends on the type of available knowledge and on the particular requirements of the application for which the system is designed. Various forms, such as logic, semantic networks, schemata, and production rules, have been used to represent knowledge [50-53]. These forms are not mutually exclusive, however.

Logic is the oldest form of knowledge representation. Both declarative and procedural knowledge can be easily represented with logic. Propositional calculus is a type of logic that lends itself to representing declarative knowledge with propositions or logical statements. These statements can be related by the logical relations, such as *and*, *or*, *not* and *imply*. The *imply* operation is equivalent to the *if-then* statement. Hence, it allows the representation of procedural knowledge also. Since propositional logic is limited to full propositions or statements that are true or false, its ability to represent real

world knowledge is limited. Another type of logic that is more attractive is based on predicate calculus. It uses the same concepts and rules as propositional logic. However, it has the ability of representing knowledge in finer detail by breaking a logical statement into components such as objects, their characteristics or some of their assertions. One weakness of logic as knowledge representation is that it does not address the issues of how to organize and interconnect the knowledge in the knowledge base. Therefore, other forms of knowledge representation are used in conjunction with logic in order to structure the knowledge base.

For instance, in semantic networks the knowledge is structured as a group of interconnected nodes. The nodes represent facts or concepts, while the interconnecting arcs represent semantic relationships between the concepts. Inference is made by traversing the arcs from one concept to the next related one. A semantic network is very useful when the knowledge can be categorized into hierarchical clusters. It also has the advantage of the inheritance property, in which children nodes inherit the properties of their parents. However, as the amount of knowledge increases, the complexity of the network becomes burdensome and the reasoning process becomes very complicated.

Frames and scripts are two other forms for structural representation of knowledge. Both forms are known in the literature as schema representation of knowledge. In frame representation, each frame represents an object, and the facts about that object are represented as attributes of the frame. A special attribute, called the relation attribute, is used to link the frames together. Both declarative and procedural knowledge can be represented with frames. The procedural knowledge can be stored in separate frames that are linked to declarative frames or to the attributes of the declarative frames.

Scripts are very similar to frames, except that knowledge is represented with scripts, each of which describe a scenario of related events. These events always occur in conjunction with each other. They are invoked when the entry condition of the script is satisfied. This entry condition is indicated in the script, as well as the order of occurrence of the related events and their outcomes.

In production rules, the declarative and procedural types of knowledge are separated into a fact-base and a rule-base, respectively. The declarative knowledge is represented as propositions based on propositional logic, and the procedural knowledge is represented as a series of *if-then* rules. Some of these rules pertain to the field of application, while others, called meta-rules, pertain to other production rules or even to themselves. A meta-rule guides the execution of the knowledge-based system by determining under what conditions certain rules should be applied in preference to others. Facts can be added or deleted from the fact base as the system executes. Facts are added whenever the assertion of a proposition is established, otherwise deleted. Production rules are the basis of most modern expert systems.

2.1.3. The Inference Engine

The inference engine is responsible for the reasoning process in a knowledge-based system [53]. It uses the input data, the knowledge-base, and logical deduction to produce intelligent results. These results are obtained by invoking rules in an orderly sequence to eventually produce a logical conclusion. This process is called the inference process.

The inference engine uses rules such as *Modus Ponens*, *Modus Tolens*, and *Resolution*, along with predicate calculus for the inference process [54]. These rules are already programmed into some artificial intelligence language and the development tools. The *Modus Ponens* rule states that if propositions A and (A implies B) are true, then proposition B is true. The *Modus Tolens* rule says that if propositions (not B) and (A implies B) are true then (not A) is true. The *resolution* rule says that if the propositions (A or B) and (not B or C) is true then (A or C) is true. (A or C) is called the resolvent of (A or B) and (not B or C).

Forward and backward chaining are two major methods which are used to control the inference process [53]. In forward chaining, the inference engine evaluates the predicates of each rule in the knowledge-base, then evaluates those rules which have predicates that match the consequences of the original rules. In backward chaining, the inference engine proceeds from the consequences of the rules to the predicates. It first establishes some hypothesis, then evaluates the predicates that lead to these hypothesis. Whenever the truth of a hypothesis is established, the hypothesis is converted to a new fact that is added to the database. In both forward and backward chaining, if the truth of a predicate can not be determined, the system seeks additional data. If this data is not available, the system ignores that predicate and proceeds the evaluation of another.

A search mechanism is essential in order to achieve forward or backward chaining. Several basic search techniques, such as depth-first, breadth-first, and controlled search, have been proposed [55]. Depth-first and breadth-first methods follow a predetermined path to search for the goal; constraints are used in a controlled search to determine an optimal search path and to minimize the searching effort. In depth-first

searching, the knowledge tree is searched by exploring the nodes of the tree in a vertical fashion, starting from the root; i.e., the search proceeds from one level to the next by successively selecting one node from the given level and then exploring one of its children, until the goal is reached. If the desired goal can not be reached, the tree is traversed backward to the nearest unexplored ancestor node and the search continues forward from there in the same fashion. In a breadth-first search, nodes on the same level are explored until the goal node is reached. If all the nodes in one level are explored, but the goal node is not reached, then the search is continued in the next level.

The same principle as in depth-first or breadth-first searching is used in the controlled search. The only difference is that selected nodes are explored instead of exploring each node in a predetermined manner. The selection process is always based on some heuristic measure that indicates how fast a node will lead to the desired goal. Beam-search, best-first, hill-climbing, branch-and-bound, and A^*, are famous examples of controlled search techniques.

2.2. Computer Animation and Image Coding

Animation can be defined as a technique in which the illusion of movement is created by photographing a series of individual drawings onto successive frames of a film. These drawings have been previously generated such that each frame in the series is an alteration of the previous frame. When the film is projected at a certain rate, typically 24 frames/second, the illusion of motion occurs due to the persistence property of the human vision system, which is the natural ability of the human eye to retain a picture for just a fraction of a second after viewing it.

There are three major techniques for producing animated films: conventional animation, rotoscoping, and modeled animation [56].

2.2.1. Conventional Animation

Conventional animation is oriented towards the production of two-dimensional cartoons. The frames of the film to be animated are hand-drawn in a tedious and time consuming process consisting of two steps. The first step is to prepare special frames called the storyboards or key frames. These frames represent the beginnings and ends of important motions, illustrate important characters' expressions, or set the tone of the animation. The second step is the in-betweening step, in which the gaps between the key frames are filled in with a number of frames that make the motion appear to be continuous smooth.

To create one frame of an animated film, the characters or pieces of a character are hand-drawn on a clear plastic sheet of cellulose (called a cel for short). The cels are stacked together and photographed. Successive frames are then created by changing or moving appropriate cels of the stack and re-photographing it. For example, the facial features and expressions may be changed to give the impression that a character in the film is speaking. Also, transition and zooming techniques are used to switch from one scene to another or to move into or a way from a scene, respectively.

Nowadays, computers are used to assist animators in the creation of both key frames and in-between frames [57]. The animators usually use an interactive graphics editor to create key frames in a digital format, or to interpolate in-between frames from

the key frames. For interpolation purposes, animators divide each key frame into a number of small strokes. Then they establish a correspondence between the strokes in different key frames. Therefore, the number of strokes in every two consecutive key frames, as well as, the number of points in any two corresponding strokes, must be the same. These two conditions are rarely satisfied in any animated sequence. However, a preprocessing stage can force both conditions to be satisfied.

There are three major interpolation techniques that can be used to produce in-between frames [56,57]. The linear interpolation technique is the simplest of all three. In this technique, the motion from one successive key frame to another is assumed to be constant, which forces the trajectory of each point in the image to be a straight line. Therefore, the location of each point in any in-between frame can be easily determined from the positions of its corresponding two points in its two boundary key frames. Unfortunately, linear interpolation suffers from motion discontinuity at the key frames, because the motion between successive key frames is not actually constant. This discontinuity, however, can be alleviated by performing the interpolation on the basis of physical laws that drive the movement, or by using a P-curve as an approximation to the temporal behavior of the position.

Motion discontinuity can also be reduced by performing the interpolation based on the skeleton of the key frames [58]. In this approach, the key frames are first reduced to their skeletons, which are then used to interpolate the skeletons of the in-between frames. A computer is used to add the details of the objects to the interpolated skeletons according to a pre-described model. This approach is similar in principle to how human creators of in-between frames mentally use the key frames as guides without using all the

detailed information contained within them. The advantage of this method is that the computer can create high-quality in-between frames from consecutive key frames, which are very similar to each other.

A more complicated interpolation technique has been proposed by Reeves [59] to reduce motion discontinuity at the key frames, and to allow interpolation based on multiple motion paths. The main idea is to associate a time- and space-varying curve with selected points of an animated object. This curve then controls the trajectory and dynamics of these points in the film.

2.2.2. Rotoscoping

Rotoscoping is an animation technique in which the time of creating an animated film is reduced by using films of real-life actions. These actions are traced to produce cels which are then inked, printed and filmed. In so doing, it is possible to eliminate the time required to initially design the key frames and in-between frames of an animated film, as in normal animation. The films which result from rotoscoping have a fluidity and style not normally seen in conventional animation; it is usually easy to tell the difference between rotoscoped and hand-designed animation, though determining which looks better is often a matter of context and taste. Animation and live films may be lumped together to produce the so called mixed media.

2.2.3. Modeled Animation

Modeled animation is a recent approach in the animation industry [43,44,60,61]. In this technique, a 3-D computer model is first constructed for each character or object to be in an animated film. These models are then manipulated either by changing some of their parameters or by applying simple linear operations such as scaling, translation, and rotation in order to create motion. Although this approach allows considerable flexibility in defining the objects or characters and their motions, it has achieved only limited success. Modeled animation systems are awaiting improvements that would enable them to be extendible and to learn as they work. Such systems would become more powerful and intelligent with each use.

2.3. Composing Pictures from Pieces of Other Pictures

Composing pictures from pieces of other pictures has long been used in applications such as animation, computer graphics, and forensics. As discussed in section 2.2, conventional animation uses characters or pieces of a character which are hand-drawn on clear plastic sheets of cellulose called cels. Several cels may be stacked together and photographed to create one frame of an animated film. A computer graphics system called *Whatsisface*, has been developed by Gillenson and Chandrasekaran [62], to draw human faces on a graphics display. The system has pre-stored line-drawings of average human faces and facial features. The user first horizontally and vertically stretches an average face to the desired proportions and then updates its average features by replacing them with more suitable features from the database. Special kits of facial features are also used in forensic applications in order to construct the likeness of a criminal from the

memory of a witness. *Photofit, Identikit, Magnaface, Videofit,* and *Minolta Montage Synthesizer* [63] are examples of some commercially available kits.

Photofit and the *Identikit* contain a range of facial features, including eyes, noses, mouths, chins, and hair sections, that have been abstracted from monochrome photographs. Each kit has a total of a round 560 facial features. An eyewitness selects suitable facial features from the kit and assembles them into a composite image of the criminal. Features in the *Photofit* kit are printed onto thin cards, which can be slotted together in a special frame in order to produce a composite face. Features in the *Identikit* are printed onto transparent acetate sheets and the face is created by superimposing relevant sheets.

Magnaface is a more recently developed kit which was designed to produce a realistic composite portrait in full color. Face construction begins with a featureless face, called the clone, which is attached to a magnetic board. The individual facial features incorporate metallic backings so they can be laid smoothly and securely onto the clone, allowing a complete face to be composed feature by feature. Finally, special color overlays are placed over the composite in order to produce colors. Minor amendments can then be made to the face with cosmetic pencils and coloring materials supplied with the kit. The use of color in *Magnaface* improves the overall quality of likeness of the composites. *Magnaface* enjoys a 10% overall advantage over *Photofit* and *Identikit* [63].

A fourth system, which is based on the *Photofit* system was introduced by BBC television, and is called *Videofit*. In *Videofit*, a *Photofit* composite image is electronically processed to remove boundary lines between facial features, and colors are added to the different parts of the face, such as the eyes, flesh, or hair. In addition, features or

accessories can be transferred between a pair of composites in order to demonstrate appearance with or without disguise. No objective assessments of *Videofit*'s effectiveness have been conducted, but criminals have been identified and apprehended with the aid of *Videofit* images.

Minolta Montage Synthesizer is a device that optically blends features of different images into one composite image. This device has three parts: an optical blender, a closed circuit television camera, and a television monitor. The blender has four ports; one for the base input image and the other three for secondary images. Certain features are selected from the secondary images and blended with the base image. This is achieved by filtering out parts of the base image while simultaneously reflecting parts of the secondary faces. The size and brightness of each secondary image can be adjusted to obtain a good blend. The composite image then passes to the television camera and the monitor.

2.4. A General Methodology for Knowledge-Based Image Coding

In this section, basic ideas of animation are incorporated into a general knowledge-based system in order to develop a knowledge-based image coding system. As in animation, any image can be broken into a set of disjoint region, each of which encompasses a small object or feature of the original image. For example, an image of a human face can be broken into regions containing physical features, such as the eye, mouth, or nose. These regions may be considered to be the primitives of such an image, where each primitive may have more than one state. For instance, the mouth may be closed, slightly opened, opened, or widely opened.

For each class of images, there is a universal set **S** of primitives of which the set $\mathbf{P_f}$ of primitives in any image **f** in the class, is a subset. If $\mathbf{P_f} = \{p_1^{(j_1)}, p_2^{(j_2)}, \ldots, p_{n-1}^{(j_{n-1})}, p_n^{(j_n)}\}$, where the superscript j_i indicates the state of the i^{th} primitive p_i, then the image **f** can be represented as the union of its primitives; i.e.,

$$\mathbf{f} = \bigcup_{i=1}^{n} p_i^{(j_i)} \tag{2.1}$$

If the universal set **S** is known and the subset $\mathbf{P_f}$ for an image **f** is identified from **S**, then the image **f** can be easily reconstructed from the primitives of set **S**, using some information about the sizes, orientations, and locations of the original primitives in **f**.

Since the type of primitives that might appear in a given class of images is finite, the set **S** is finite, and therefore can be represented by a database. This database is very similar to the code book used in vector quantization, which was discussed in chapter 1. The difference is that the code book contains either one-dimensional or two-dimensional vectors, which do not represent any meaningful messages. They are simply groups of adjacent pixels, which frequently occur in the image. The database, on the other hand, contains meaningful image segments that represent the viewer's perception of the outside world. This database can be used to code any image in the given class in a way similar to that used in vector quantization.

The coding process starts by using image analysis techniques to extract the primitives $\mathbf{P_f}$ of the input image. Each of the extracted primitives is normalized with respect to size and orientation, and is then matched to the database, which is assumed to

be available before coding. When the best match is found, its order in the database, along with the primitive's normalization factors and location in the original image, is coded and transmitted. If a good match for a primitive can not be found, the primitive itself is transmitted and the databases at both the transmitter and the receiver are updated to include the new primitive. In coding time-varying image sequences it is not necessary to extract and code all the primitives from each frame of the sequence. Instead, it is sufficient to extract and code only those primitives that have encountered significant changes, such as rotation, scaling, or deformation from one frame to the next. These primitives are then projected on the previous frame which is assumed to be stored at the receiver.

The receiver uses the transmitted information and a duplicate database to construct a faithful replica of the original image. The decoding process is similar to the idea used in computer animation. First, each primitive is decoded from the database using its order; it is then scaled and oriented using its normalization factors so that it assumes its original size and orientation. Finally, each primitive is projected onto its proper location in a base image, which is simple the previous frame in a time-varying image sequence, and any artifacts that might appear in the transition regions between primitives are removed with a smoothing filter.

It is obvious that image compression is achieved in knowledge-based image coding technique by transmitting a small number of parameters, i.e., the order of a primitive in the database, its original location in the image, and its normalization factors, instead of transmitting the primitive itself.

2.5. The Structure of the Knowledge-Based Image Codec

2.5.1. Knowledge-Based Image Coder

The general structure of the knowledge-based image coder is shown in the block diagram of Fig. 2.2. It consists of eight basic blocks; the primitive extractor, fact base, database, database updator, primitive matcher, primitive coder, the binary coder and the controller, which maintains the entire system.

The primitive extractor is a simple image analyzer which is used to break the image into its primitives, each of which forms a meaningful message. The primitive extractor may use a priori knowledge to locate and extract the primitives in the image. A priori knowledge, such as the type of image to be coded, the image formation model, the primitive location model, and the primitive motion model is stored in the fact base, which is accessible to the primitive extractor.

The primitive matcher searches the database in a sequential order to find the best match to an extracted primitive. If the best match is found, the primitive matcher passes the order of the best match to the primitive coder; however, if the best match is not found it passes the message (i.e., the primitive) itself to the primitive coder. The primitive coder produces two types of message based on its input. If the message itself is input to the primitive coder, the coder represents the primitive with a minimum number of bits by using a suitable coding strategy. If the order of a best-match primitive is input to the primitive coder, it codes the order of the primitive and its normalization factors. In either case, the primitive coder adds overhead bits to distinguish between the two types of messages.

The database is a collection of many image primitives that might appear in an image in a given application. Its contents are application dependent. For instance, for head and shoulders images, the database contains pictures of various eyes, noses, mouths, hair segments and ears (see section 4.9); for facsimile purposes, it would contain alpha-numerical characters in many fonts, and basic graphics primitives. In any case the database is assumed to be available before coding.

The coder's controller coordinates and synchronizes the operations of all the coder blocks. First, it determines the mode of operation based on the input image, then it tells the primitive extractor which facts from the fact base are to be used in extracting the primitives of the input image. It also manages the database through the database update routine, and decides how a primitive is coded by the primitive coder. Therefore, it interacts with the primitive matcher, and whenever an acceptable best match of a primitive is found in the database it informs the primitive coder to code the order of the best match in the database and the necessary normalization factors. However, if an acceptable best match is not found, the controller signals the primitive coder to code the primitive itself with an optimum coding strategy that produces the least number of bits. It then informs the database update routine to add the primitive to the database. Finally, the binary coder codes and transmits the information produced by the primitive coder.

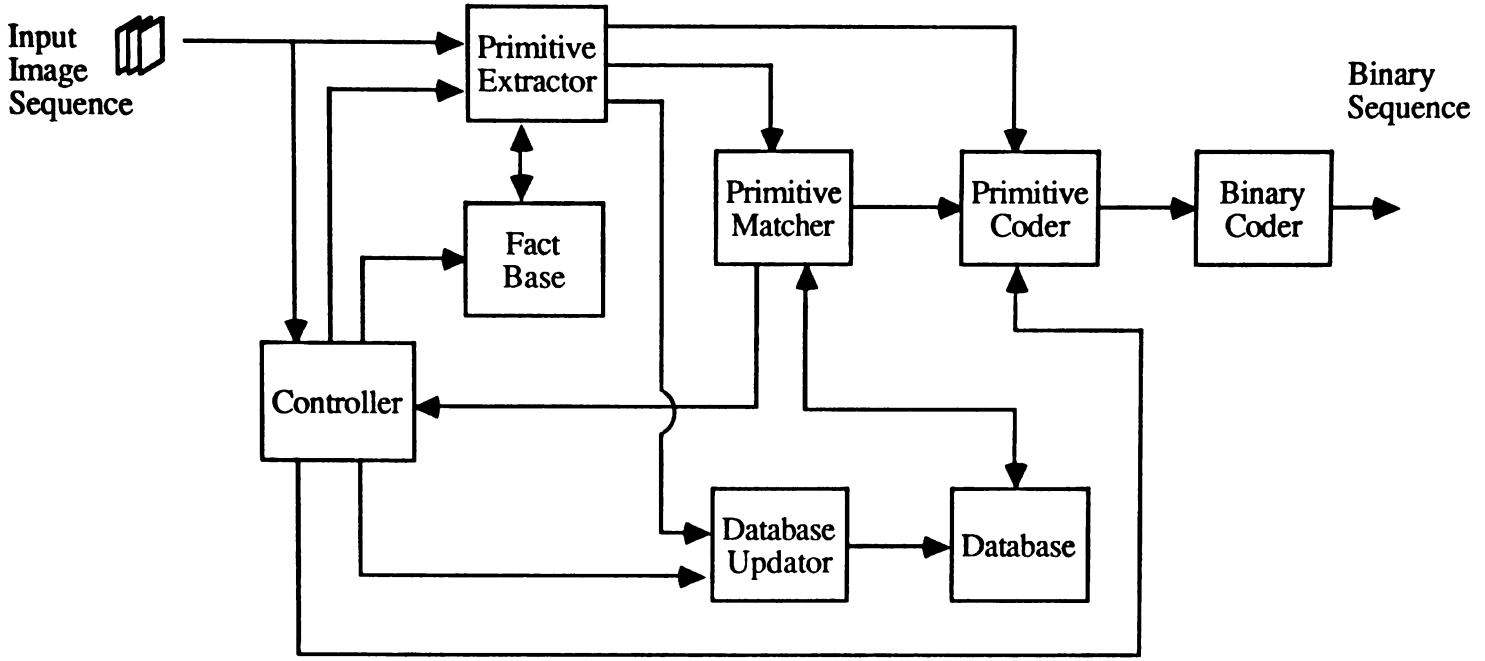


Fig. 2.2 Knowledge-Based Image Coder

2.5.2. Knowledge-Based Image Decoder

The knowledge-based image decoder operates in full synchronization with the coder, and its structure is similar to that of the coder. As depicted in Fig. 2.3, the decoder consists of nine blocks; the binary decoder, primitive decoder, primitive fetcher, primitive projector, controller, database, database updator, frame buffer, and post processor.

The binary decoder decodes the received binary message, and passes it to the primitive decoder, which in turn decides whether the primitive itself or the order of the primitive and its normalization factors was received. If the primitive itself was received, then it decodes the primitive and its location in the image and passes this information to the primitive projector; otherwise, it decodes the order of the primitive and its normalization factors, and passes the order to the primitive fetcher and the normalization factors to the primitive projector.

The primitive decoder also informs the controller concerning the type of received information, whereupon the controller signals the primitive fetcher and the primitive projector to perform the required actions. For instance, if the primitive itself was received, the controller signals the primitive fetcher to do nothing, and the primitive projector to take its input from the primitive decoder. On the other hand, if the primitive itself was not received, the controller signals the primitive fetcher to use the order of the primitive in the database to extract the primitive from the database, and then passes the fetched primitive to the primitive projector.

The primitive projector reverses the action of the primitive extractor of the coder. It takes a primitive and its normalization and location factors as inputs, and then scales

and directs the primitive to its proper location in the base image. The base image is stored in the frame buffer; it can be a typical image or the previous frame of a time-varying image sequence. Finally, the post processor uses an average filter around the boundary of each primitive to remove any artifacts that appear in the transition regions between the primitives.

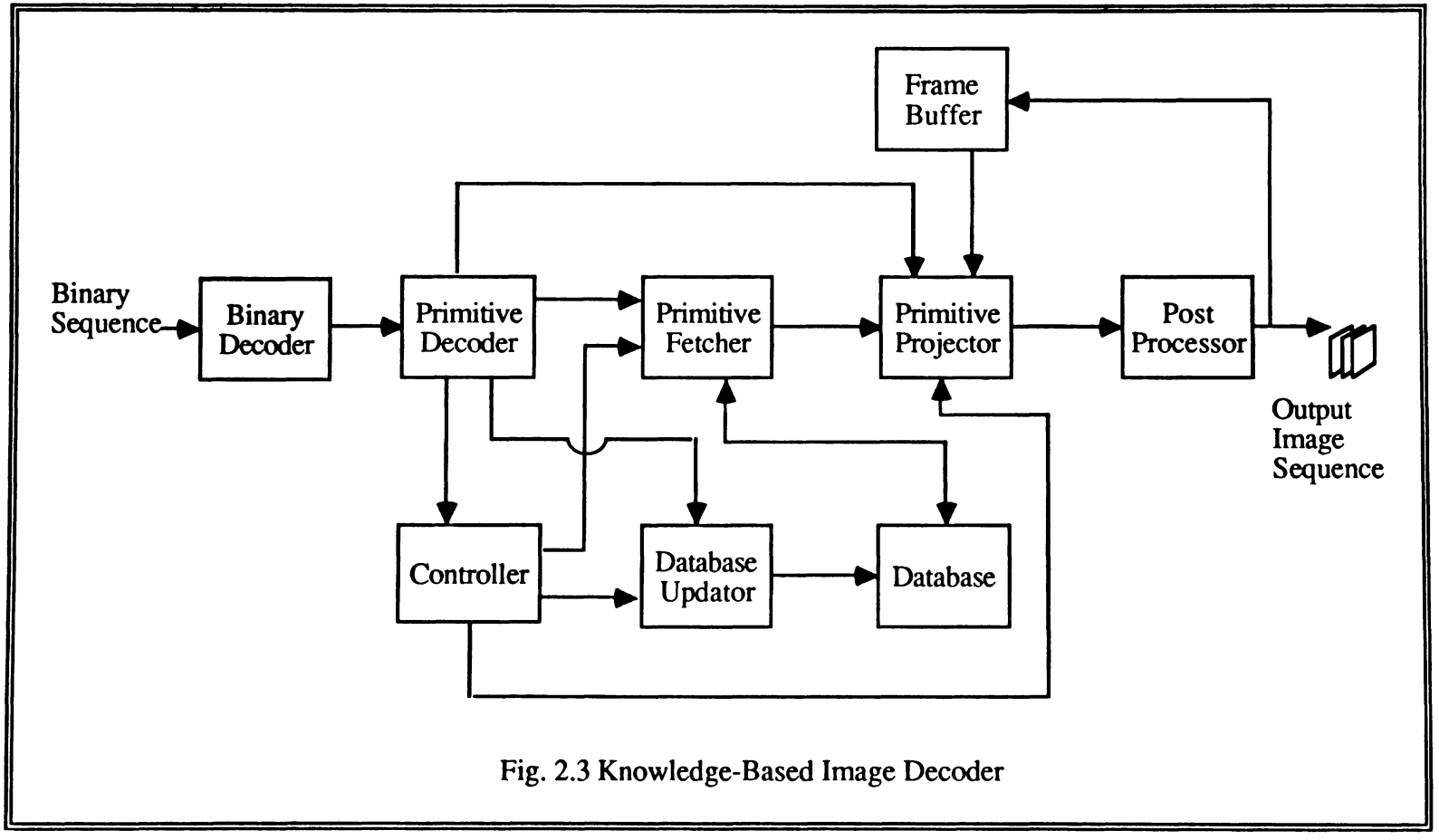


Fig. 2.3 Knowledge-Based Image Decoder

The database at the decoder is an exact duplicate of the one used by the coder. It is updated through the database update routine whenever the database at the transmitter is updated. Therefore, whenever a primitive itself is received, the controller informs the database updator to update the decoder's database with the received primitive.

2.5.3. General Requirement for Applying Knowledge-Based Image Coding Techniques

It is obvious that applying the knowledge-based image coding technique to a given class of images requires the following: (1) an accurate definition of the primitives of the images in that class, (2) a fast and accurate primitive extraction algorithm, (3) the construction of a representative database of reasonable size, and (4) a way to compare primitives.

In the next chapter a model-based algorithm is developed to extract the primitives of head and shoulders images, which are similar to those encountered in face-to-face telecommunications applications. This algorithm is then used in chapter 4 and 5, where the knowledge-based image coding technique is applied to still frames and time-varying image sequences, respectively.

CHAPTER 3

A MODEL-BASED ALGORITHM FOR FACIAL FEATURE EXTRACTION

3.1. Introduction

In spite of the obvious potential of face recognition to applications such as security and law-enforcement, limited success has been achieved. Most of the existing techniques are based on either verbal coding or geometrical coding of the image, followed by a sequential or matching algorithm. Goldstein et. al. [64] used verbal coding, which describes a picture verbally to develop an interactive face recognition system. Harmon et. al. [65-67] used geometrical coding in automatic identification of human face profiles. Each face profile in Goldstein's world was represented by a vector of seven automatically-extracted fiducial marks. These vectors were then used to compare the profile pictures. Sakai et. al. [68] and Bromley [69] developed algorithms for automatic location of the facial features in front view images. Sakai [68] used the vertical and the horizontal signatures of the pixels in a slit of predetermined dimensions to determine the location of the facial features. The slit moves around the picture in search of the location of the desired feature. This location must be consistent with the locations of other features. Whenever the algorithm detects an inconsistent location, the process is repeated in an iterative manner. Bromley [69] used a similar algorithm, except that the horizontal and vertical signatures of the global image, rather than of a slit, were used.

The location of facial features is necessary not only in face identification, but also in constructing composite images from parts of photographs. Gillenson and Chandrasekaran [62] developed a computer system called *Whatsisface*, which utilizes a database of human facial features to create facial images on a CRT. Wiederhold [70] developed a computer system to simulate the *Menolita Montage Synthesizer*, which blends features from different pictures to produce a composite for criminal identification. In both *Whatsisface* and the *Menolita Synthesizer*, the location of the feature to be extracted or projected is determined manually. Also, determining the locations of the facial features is very necessary for our knowledge-based image coding technique.

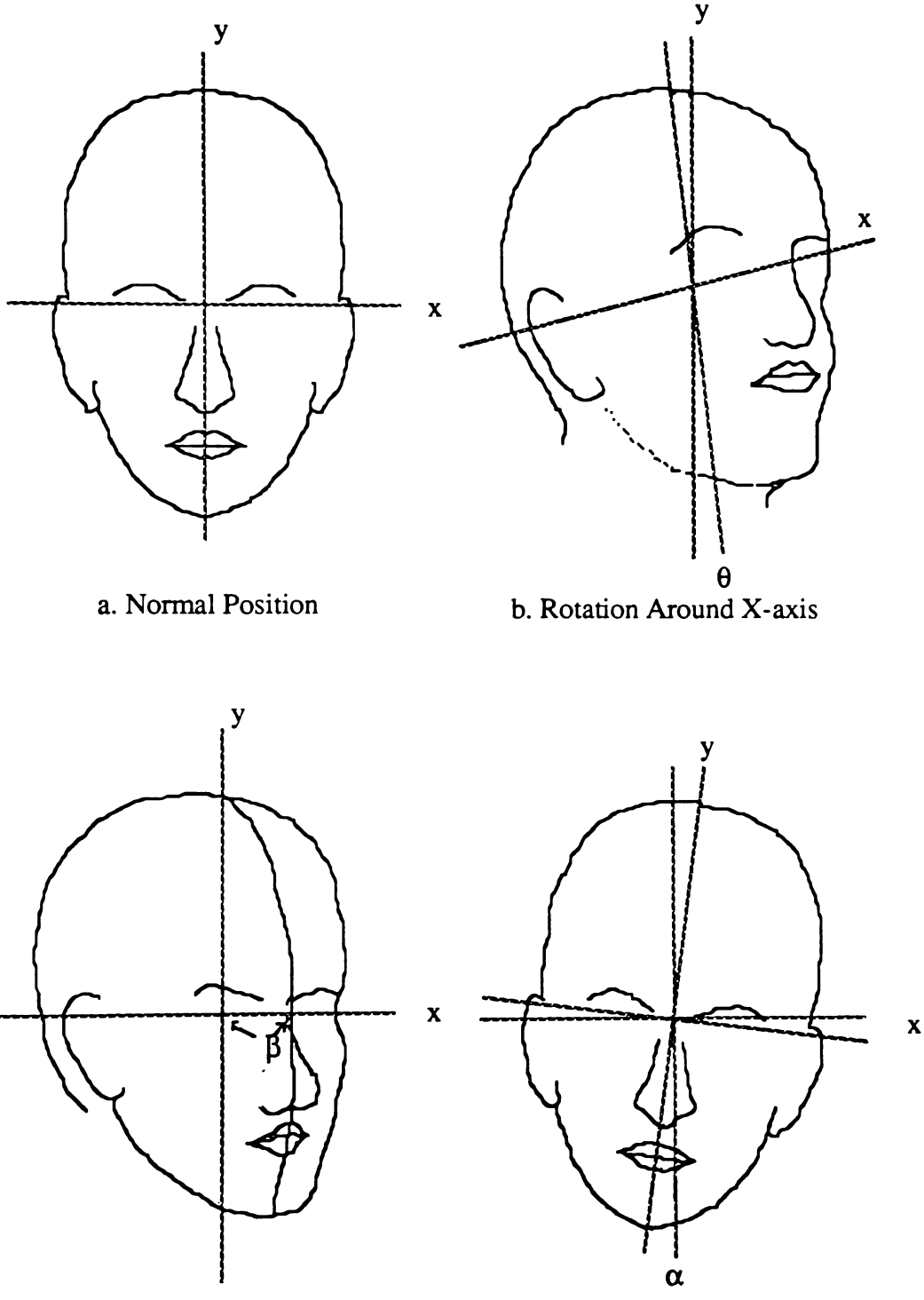
In this chapter, a model-based one-pass algorithm is developed which quickly and accurately locates the desired facial features. The art of drawing the human head is used to develop a model for the head in the photograph. Since the human head can be viewed as an ellipsoid, the head view in the photograph can be modeled by an ellipse. The algorithm estimates the parameters of the ellipse which best fits the head view in the photograph, and uses these parameters to estimate the locations of the facial features. It then adjusts the vertical coordinates of the mouth and nose by using the signature of the pixels in a vertical slit around the nose and mouth. In general, this model is good for all types of pictures; however, our discussion and implementation will be restricted to images of people with short hair. To extend the work to include people with long hair, one needs to model the face instead of the head.

3.2. A Model for Head and Shoulders Images

The human head is normally visualized by an artist in two ways, a cube or an egg [71,72]. The two approaches stress different prominent features of the human head that suit the drawing task. The cube shape stresses the angularity of the facial planes and the underlying bony structure of the skull. The egg shape stresses the curving quality of the skull and the fact that the facial features sit on a curve rather than on a flat surface. Neither model, however, is well suited to locating the facial features. A more mathematically appealing model is the ellipsoid. Therefore, the head of a human being can be thought of as an ellipsoid that sits on the top of the torso and is attached to it at the center by the neck.

The head movement can be classified mainly as turning right and left or nodding up and down. Both movements are supported by the anatomical structure of the neck. The first cervical vertebra is called the atlas. It supports the skull and allows it a large turning radius. The atlas, in turn, sits on the second cervical vertebra which is called the axis. The axis is a hinge joint that allows the head to nod up and down. The remaining five cervical vertebrae are flexible and extend the possibilities of motion for the head in nearly all directions. Therefore, the head movement can be viewed as rotations α, β, and θ around the x , y, and z axes, respectively, which pass through the center of the ellipsoid (see Fig. 3.1). The elongation and orientation of the projected ellipse are related to the head rotation angles α, β and θ. For example, in the front view position, where the angles α and β are both zero, the length of the minor axis is about two thirds the length of the major axis [71,72]. However, the ratio changes as the head moves up or down, and therefore is related to the value of angle α (assuming that the x-y plane is the image

plane). The symmetry of the head view in the original image is related to the angle β, and the tilt of the head, i.e., the orientation of the minor axis of the ellipse, is indicated by the angle θ.



a. Normal Position

b. Rotation Around X-axis

c. Rotation Around Y-axis

d. Rotation Around Z-axis

Fig. 3.1 Head Rotation Angles with Respect to the X, Y, and Z Axes

The human head can be uniquely characterized with a total of nine features [71]. These features are the brow ridge, the nose, the eye socket, the cheek bone, the mouth barrel, the chin box, the jaw corner, the ear, and the cheek bone arch. The locations of these features are normally well defined with respect to the head, which is approximately five eye-lengths wide at the mid-line [71,72] (see Fig. 3.2). Both eyes lie on the midway line and the distance between them is equal to almost one eye-length. The nose starts at the center of the face and descends to a point mid-way between the bridge of the nose and the base of the chin. The width of the nose at its base is also equal to approximately one eye-length. The mouth barrel starts at the base of the nose and extends two thirds of the distance down from the base of the nose to the chin. The sides of the barrel align with the centers of the eye sockets. This information is used to locate the facial features with respect to the center of the model ellipse. The locations are then adjusted to obtain more accurate results.

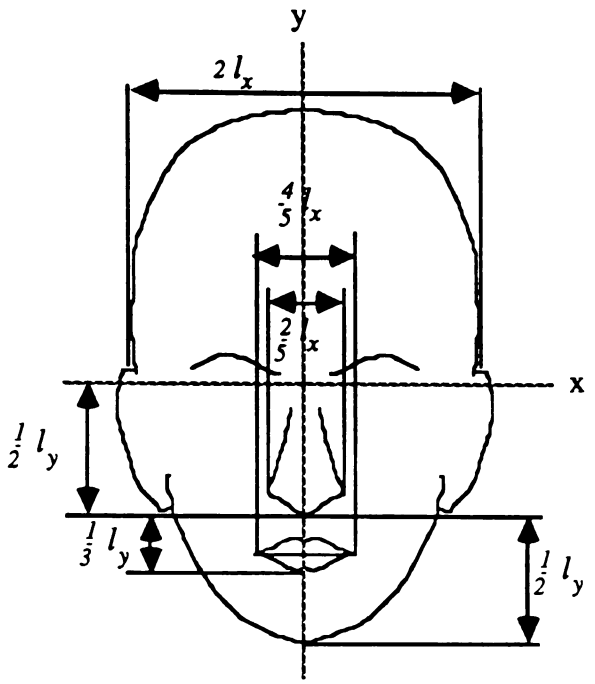


Fig. 3.2 Head Model for Head and Shoulders Images

Let (c_x, c_y) be the center of the ellipse that fits the head contour. Also let l_x, l_y and θ denote the minor and major semi-axes and the orientation of the ellipse, respectively. Using the previously described model of the head and the geometry of the ellipse, the locations of the eyes, nose, and mouth in terms of (c_x, c_y), θ, l_x, and l_y are given below.

Eyes:

If the length of each eye is e_l and the centers of the right and left eyes are (cre_x, cre_y) and (cle_x, cle_y) , respectively, then,

$$e_l = \frac{2l_x}{5} \tag{3.1a}$$

$$cre_x = c_x + e_l \cos\theta \tag{3.1b}$$

$$cre_y = c_y + e_l \sin\theta$$

$$cle_x = c_x - e_l \cos\theta \tag{3.1c}$$

$$cle_y = c_y - e_l \sin\theta$$

Nose:

If the center, length, and base width of the nose are denoted by (nc_x, nc_y), n_h and n_w, respectively, then,

$$nc_x = c_x - \frac{l_y}{4} \sin\theta \tag{3.2a}$$

$$nc_y = c_y + \frac{l_y}{4} \cos\theta$$

$$n_h = \frac{l_y}{2} \qquad (3.2b)$$

$$n_w = \frac{l_y}{5} \qquad (3.2c)$$

Mouth:

If the center, height and width of the mouth barrel are denoted by (mc_x, mc_y), m_h and m_w, respectively, then,

$$mc_x = c_x - \frac{2l_y}{3}\sin\theta \qquad (3.3a)$$

$$mc_y = c_y + \frac{2l_y}{3}\cos\theta$$

$$m_h = \frac{l_y}{3} \qquad (3.3b)$$

$$m_w = \frac{4l_y}{5} \qquad (3.3c)$$

Equations (1)-(3) are used in the facial feature extraction algorithm to estimate the location of the facial features.

The parameters of the ellipse are estimated by fitting the head contour points with the ellipse function,

$$f(x,y;a,b,c,d,e) = a\,x^2 + 2bxy + c\,y^2 + dx + ey - 1 = 0 \qquad (3.4)$$

It is easy to show that the center (c_x, c_y) of this ellipse is given by

$$c_x = \frac{be - cd}{2(ac - b^2)} \tag{3.5a}$$

$$c_y = \frac{bd - ae}{2(ac - b^2)} \tag{3.5b}$$

and the orientation θ is given by

$$\theta = \frac{1}{2}\tan^{-1}\left(\frac{b}{a - c}\right) \tag{3.5c}$$

The major and minor semi-axes, l_x, and l_y, are given by

$$l_x = \left(\frac{(c - a)(1 - ac_x^2 - bc_xc_y - cc_y^2 - dc_x - ec_y)}{(ac - a^2 - 2b^2)\sin^2\theta + (c^2 - ac + 2b^2)\cos^2\theta}\right)^{\frac{1}{2}} \tag{3.5d}$$

$$l_y = \left(\frac{(c - a)(1 - ac_x^2 - bc_xc_y - cc_y^2 - dc_x - ec_y)}{(ac - a^2 - 2b^2)\cos^2\theta + (c^2 - ac + 2b^2)\sin^2\theta}\right)^{\frac{1}{2}} \tag{3.5e}$$

A simple way to fit the head contour with the ellipse function is to minimize the fitting error, which is defined by

$$S = \sum_{i=1}^{n} (\Delta x_i)^2 + (\Delta y_i)^2 \qquad (3.6)$$

where Δx_i and Δy_i are the x and y deviations of the head contour point (x_i, y_i) from the ellipse contour. An iterative solution for this problem using the Lagrange multiplier method is shown in the Appendix. The algorithm starts by guessing the parameters of the ellipse. This initial guess can be some pre-defined values or the values of the parameters of the ellipse that fits five head contour points which are almost equidistant from each other. The algorithm then updates the initial guess by solving a set of linear equations for the deviation from the initial guess. The process continues until the deviation no significant improvement in the estimated parameters can be obtained with further iterations.

3.3. Facial Features Extraction Algorithm

In this section, an algorithm for extracting the facial features, based on the model developed in section 3.2, is described. The algorithm is summarized in the flowchart of Fig. 3.3. It consists of five consecutive stages. The first four stages are for the estimation of the parameters of the ellipse which best fits the head contour. These stages are edge detection, thinning, extraction, and curve fitting. An edge operator is used to enhance the edges. Then, the enhanced image is thresholded to separate significant edges from the background. These edges may be neither thin nor continuous. Although edge continuity is not necessary, it is desired, since it facilitates the extraction of the head contours.

Furthermore, thin edges result in better performance from the fitting algorithm. Hence, after thresholding the enhanced image, a thinning operator is applied to produce the thinned edges. The outer most head contours are extracted from the thinned image, and the parameters of the ellipse that best fits the contour points are estimated. From the parameters of the ellipse the location of the facial features are estimated.

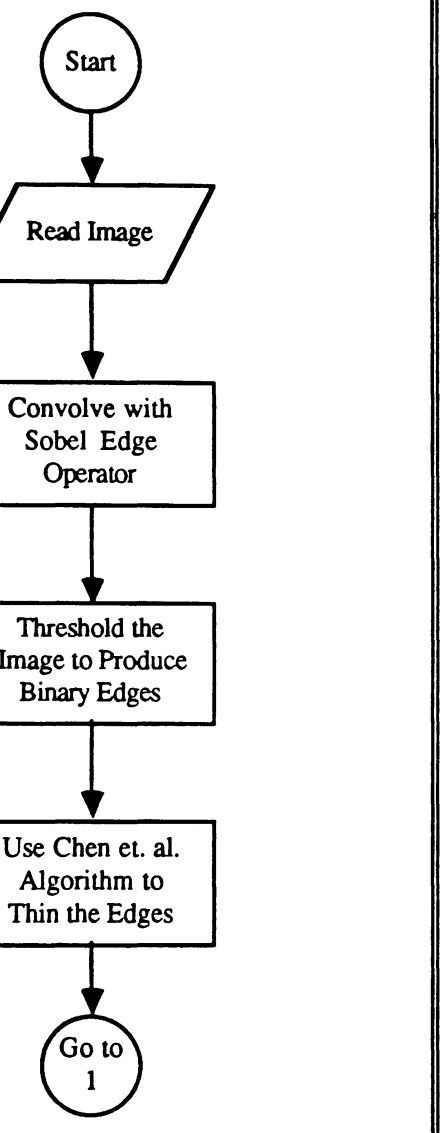


Fig. 3.3a Flowchart for Edge Detection

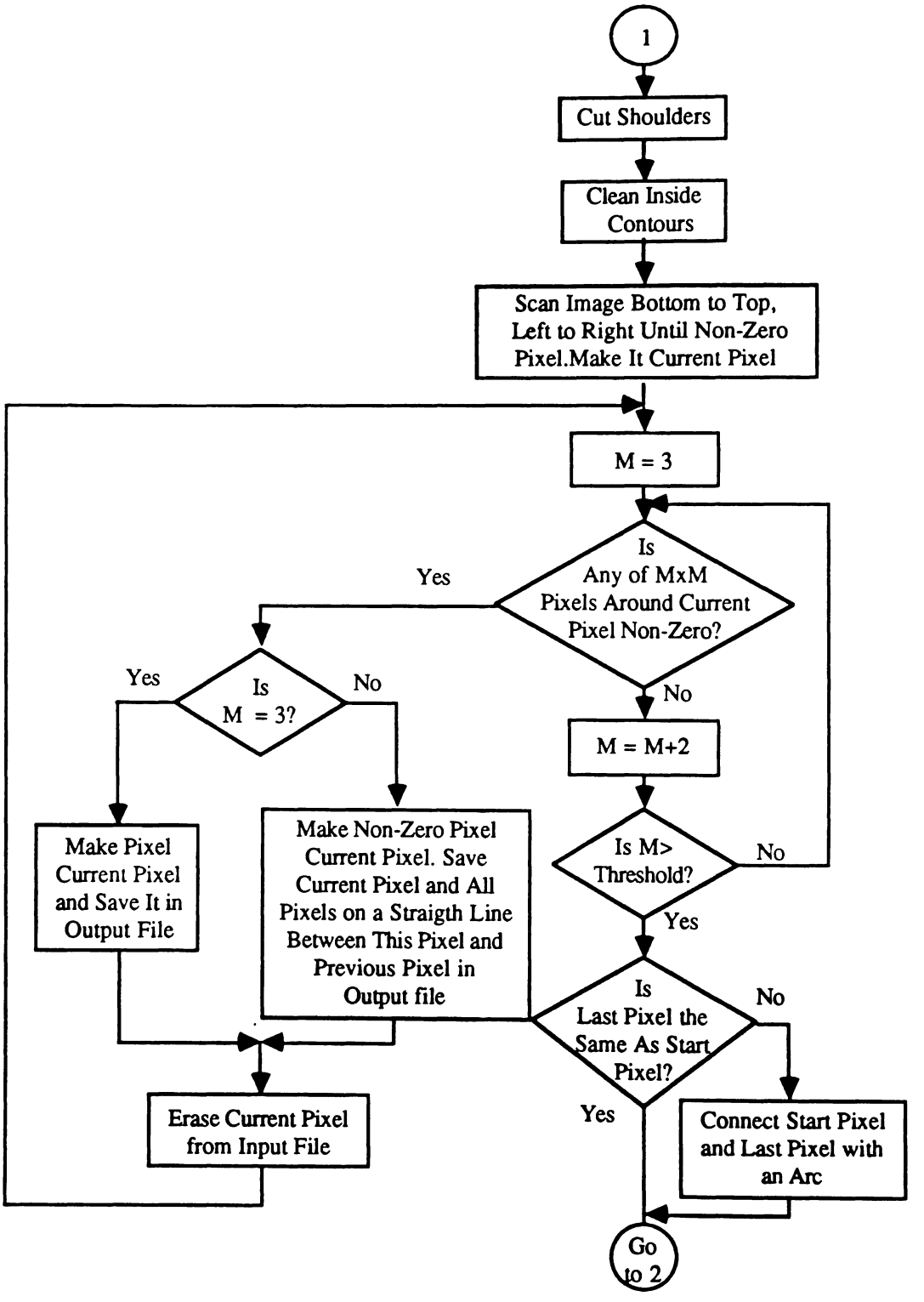


Fig. 3.3b Flowchart for Head Contour Extraction

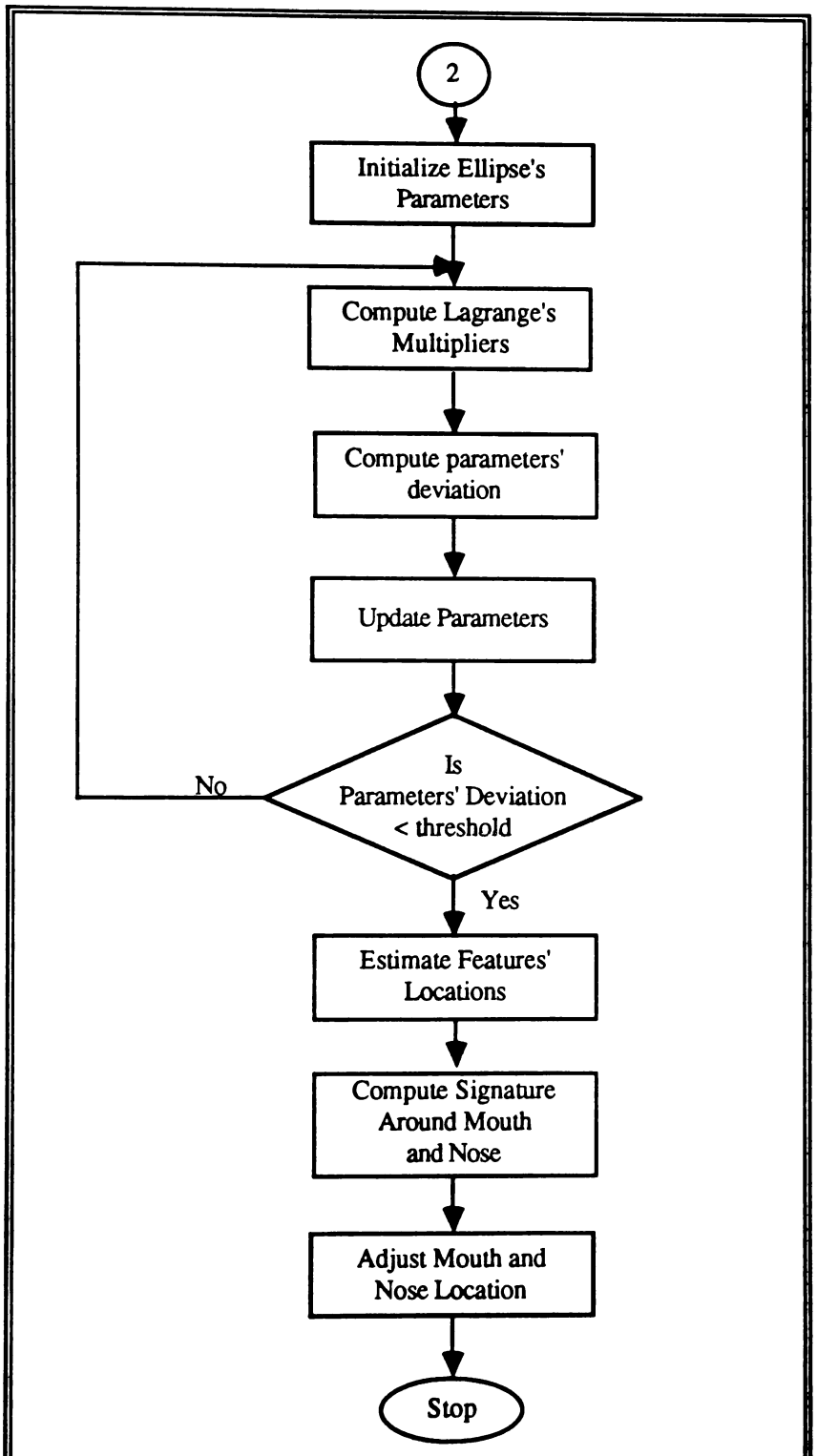


Fig. 3.3c Flowchart for Head Feature Location

The fifth stage is the final adjustment of the locations of the facial features. It is necessary because the location of the nose or the mouth with respect to the center of the ellipse varies slightly from one person to another. This stage begins with calculating the vertical signature of the pixels in a slit around the estimated nose or mouth location. These pixel values are taken from the original binary image before thinning. The base of the nose or the top and the bottom of the mouth are found from the signature by marking the start and end of the non-zero values in the slit.

The details of the above process can be summarized in the following algorithm:

Edge Detection:

(1) *Given a head and shoulders image, convolve the image with Sobel's [74] templates (see Fig. 3.4) to compute the magnitude of the image gradient. Template (a) in Fig. 3.4 approximates* $\frac{\partial f}{\partial y}$, *and template (b) approximates* $\frac{\partial f}{\partial x}$. *Hence, the magnitude of the gradient is*

$$G = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \qquad (3.7)$$

Then, threshold the gradient image globally to produce a binary edge image.

Edge Thinning:

(2) *Remove all pixels in the image that match one of the Chen et. al [75] templates (a)-(h), but leave pixels that match template (i) or (j) (see Fig. 3.5). Repeat this step until no further changes occur.*

Separating the Head from the Rest of the Body:

(3) Define the boundary which separates the head in the picture from the rest of the body. The boundary is positioned at ≈ 5 pixels above the shoulders (first non-zero pixel encountered vertically). Set all pixels in the rows below the boundary to zero values.

Outermost Head Contour Extraction:

(4) Scan the image horizontally from left to right and bottom to top, until a non-zero pixel x_0 is encountered.

(5) Inspect the pixels in an M x M window (initially 3 x 3) around the current non-zero pixel x_i. If another non-zero pixels x_{i+1} is found, go to step (6), otherwise, increase M and repeat this step as long as M is less than a certain threshold. Whenever M is greater than the threshold go to step (7).

(6) If M is 3 save the location of x_i and make the value of pixel x_i zero. If M is greater than 3, save the location of x_i and the locations of all pixels on a straight line between x_i and x_{i+1}, then make the pixel value of pixel x_i zero. Make pixel x_{i+1} the current pixel and go to step (5).

Approximating the Chin Contour:

(7) If the last non-zero pixel on the contour is not the same as the first, connect them with an arc or two line segments.

Estimating the Parameters of the Ellipse:

(8) *Initialize the parameters a, b, c, d, and e.*

(9) *Use the contour points to compute p_i in Eq. (A.6), and solve Eq. (A.4) for Δa, Δb, Δc, Δd, and Δe, (see the Appendix).*

(10) *Update the parameters a, b, c, d and e using the deltas from step (9). If the magnitude of the deltas is not small enough, go to step (9); otherwise go to step (11).*

Estimating the locations of the Facial Features:

(11) *Use the parameters from step (10) to estimate the locations of the facial features as given by Eq. (3.1)-(3.3).*

Adjusting the locations of the Facial Features:

(12) *Compute the vertical signature in a ($m_w x l_y$) slit centered at (c_x, $c_y + \frac{1}{2} l_y$); then choose the position of the last non-zero value in the slit as the base of the mouth.*

(13) *Search the signature for the valley of zero values that is closest to the center of the slit. Choose the starting point of this value as the base of the nose and the end point as the top of the mouth.*

(14) *Use the top and base of the mouth to compute the vertical coordinate of the center of the mouth. Use the base of the nose and the center of the ellipse to compute the vertical coordinate of the center of the nose.*

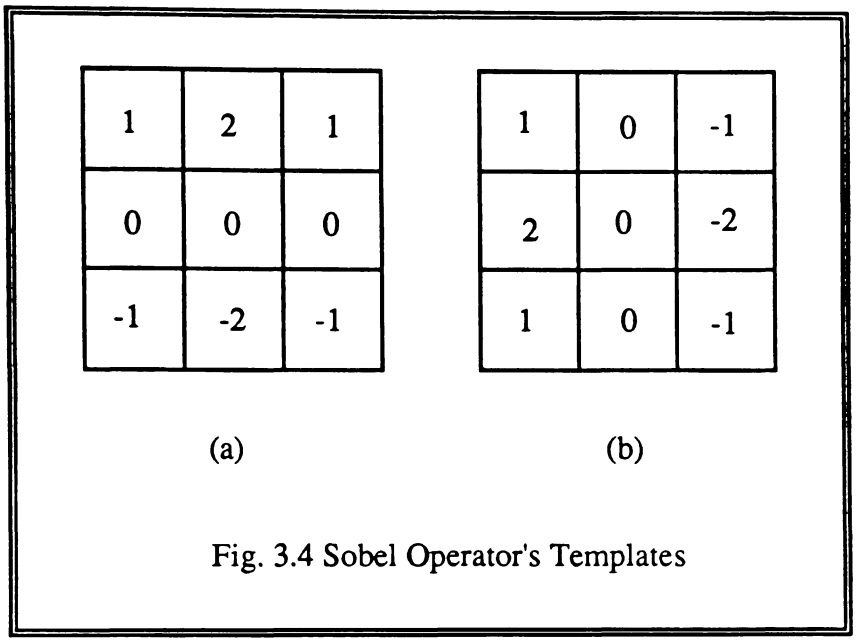


Fig. 3.4 Sobel Operator's Templates

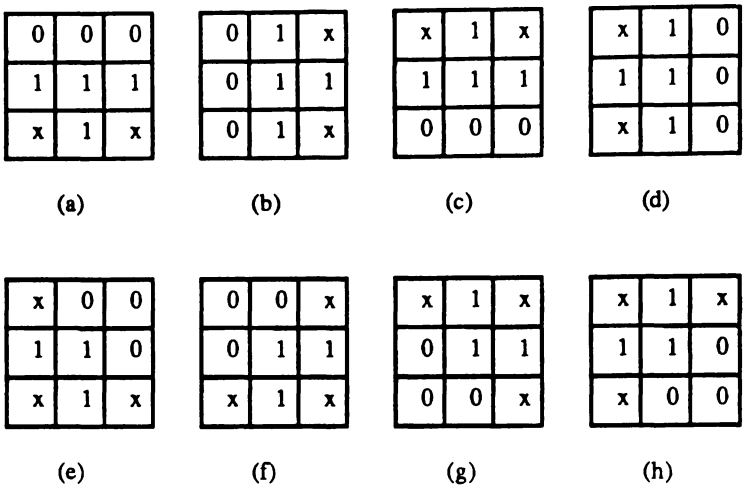


Fig. 3.5a Thinning Templates

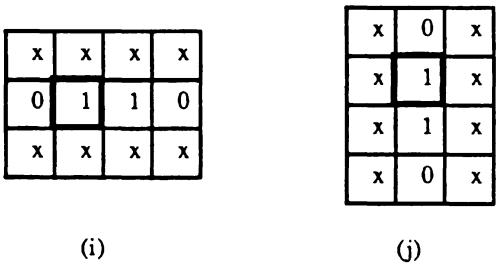


Fig. 3.5b Restoring Templates
(x's are don't cares)

Fig. 3.5 Chen et. al. Templates

3.4. Implementation and Simulation Results

The above facial features extraction algorithm was successfully implemented in the *C* language. The Sobel edge operator [74] was used to enhance the edges in the image. Sobel templates are shown in Fig. 3.4. Using the histogram of the enhanced

image, the value of the global threshold was determined. This value was used to separate the edges from the background and to convert the image into a binary image. The thinning algorithm of Chen et. al. [75] was used to produce thin edges. The Chen et. al. algorithm is an iterative parallel algorithm that has proven to be faster than other thinning algorithms due to its property of true parallelism. The templates of this algorithm are shown in Fig. 3.5.

Fig. 3.6 128x128 Input Image

When the facial feature extraction algorithm was applied to the image of Fig. 3.6, the images of Fig. 3.7 were obtained. Fig. 3.7a is the edge-enhanced image produced by the Sobel operator. The edges in this image were enhanced more than those obtained by other operators. For instance, they are stronger and thinner than those produced by Roberts and Davis operators [76]. Also, the Sobel operator outperformed Laplacian

operator, which produced a lot of undesired noise and inaccurate edges [78]. When the average Laplacian operator suggested by Sakai et. al. [68] was used, thick and inaccurate edges were obtained. This type of edge is undesirable because of the later thinning process. The intensity values of the pixels in the enhanced-edge image were re-scaled to gray levels (0-255). To produce a binary edge image, each intensity value was thresholded at a global threshold of 35, which was determined from the histogram of the enhanced image. The thresholded image is shown in Fig. 3.7b. Finally four iterations of the Chen et. al. thinning algorithm produced the thinned image of Fig. 3.7c.

Fig. 3.7a The Edge Image of Fig. 3.6

Fig. 3.7b The Thresholded Image of Fig. 3.7a

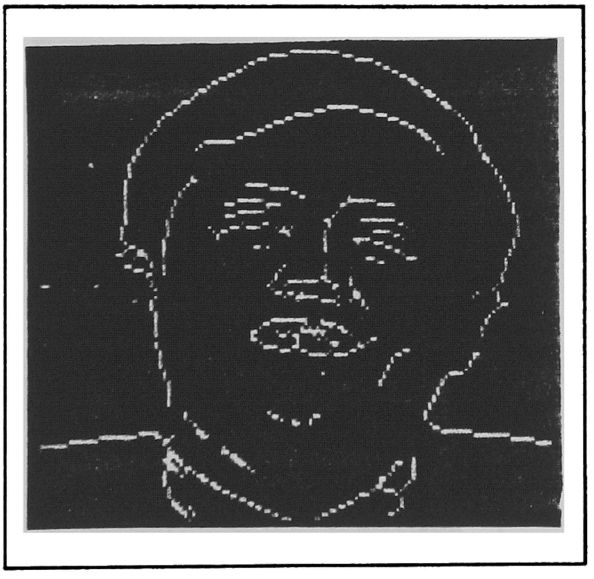


Fig. 3.7c The Thinned Image of Fig. 3.7b

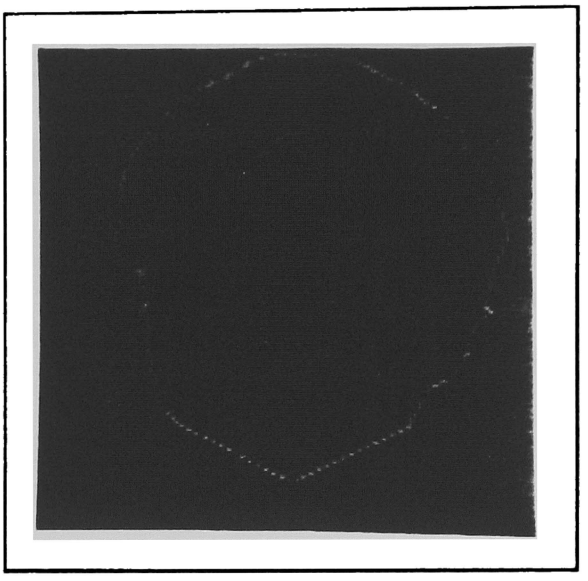


Fig. 3.7d The Outer Most Head Contour of Fig. 3.7c

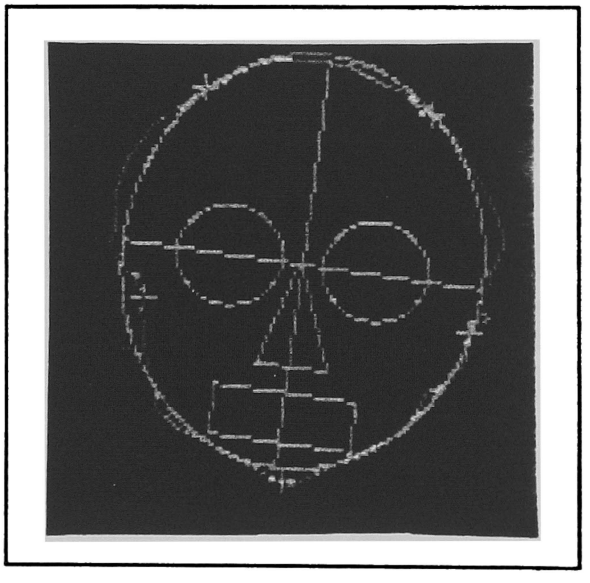


Fig. 3.7e The Estimated Ellipse for the Head Contour in Fig. 3.7c

Since the inside contours of the face are of no significance to the contour extraction process, they were first deleted from the image to avoid confusion with the head contours. In spite of the discontinuity of the edges, the previously described contour tracking algorithm has successfully extracted the outer most edges of the head, as shown in Fig. 3.7d. The curve-fitting algorithm used the contours of Fig. 3.7d to produce the ellipse of Fig. 3.7e. The estimated ellipse was superimposed over the head contour for comparison. The major inaccuracy of the estimation resulted from the fact that the chin part of the contour was missing (Fig. 3.7d), and it had to be approximated by two straight line segments. The curve-fitting algorithm took only four iterations to obtain an accurate estimate of the ellipse parameters.

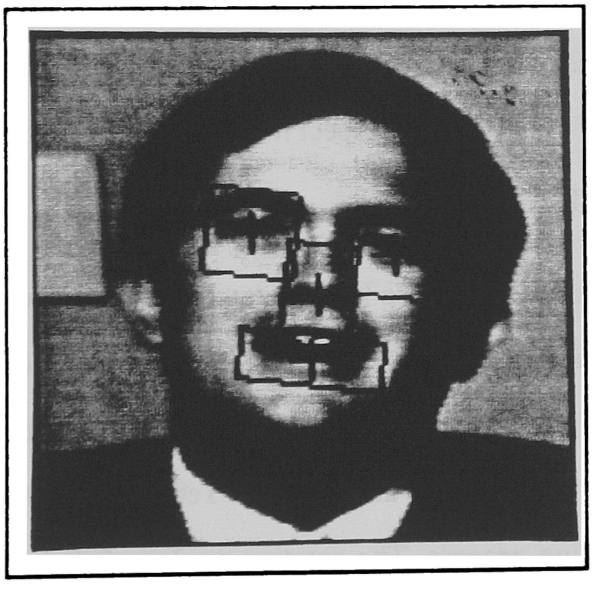


Fig. 3.8 The Estimated Locations of the Four Major Facial Features of Fig. 3.6

Next, the ellipse parameters were used to estimate the locations of the major features of the face, and the locations indicated by crosses on the image of Fig. 3.8 were obtained. Notice the shift in the mouth and lower part of the nose. This shift occurs for two reasons. The first is the approximation of the lower part of the head by two straight lines. If a curved segment had been used for this approximation, a better approximation of the vertical coordinates of the mouth and the nose would have been obtained. The second reason is that the length of the lower half of the face relative to the upper half varies significantly from one person to another. Hence, the vertical coordinates of the mouth and nose were adjusted using the vertical signature of the pixels in a slit centered at the estimated center of the nose, of length equal to the length of the semi-major axis of the ellipse, and of width equal to the width of the mouth. The adjusted locations are shown in Fig. 3.9. Obviously, the exact locations of the facial features were found without matching. Notice that it is not necessary to adjust the location of the eyes, since the length and locations of the eyes with respect to the length of the minor axis of the face is almost the same for all people.

This algorithm has also been tested on images taken from a sequence of pictures of a speaker. The motion in this sequence was not restricted to two-dimensional motion, but consisted of natural and normal motion. The speaker naturally rotates his head to both sides and nodes up and down as he speaks. Ten pictures of different rotation angles were used in the testing process. The rotation angles of those images varied from 0.27° to 7.36°. The picture in Fig. 3.6 represents the maximum rotation of 7.36°. In all cases, the algorithm produced accurate estimates of the three major facial features.

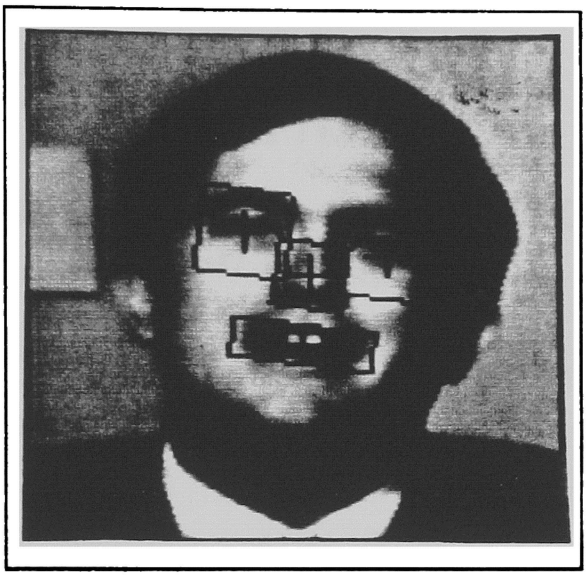


Fig. 3.9 The Adjusted Locations of the Four Major Facial Features of Fig. 3.6

CHAPTER 4

KNOWLEDGE-BASED IMAGE CODING TECHNIQUE FOR STILL PICTURES

4.1. Mug-Shot Images as an Application

In forensic applications, thousands of head and shoulders images of previously apprehended criminals are collected in mug-shot files. In spite of recent advances in image compression techniques and electronic storage technology, the tremendous amount of data forming these files is a major obstacle in the way of realizing a fully computerized archiving system with storage and transmission capabilities. However, since users are often more concerned with the general view of a picture than with its details, these files can now be efficiently coded by using the new knowledge-based image coding technique.

As mentioned in chapter 2, applying the knowledge-based image coding technique to a given class of images requires: (1) an accurate definition of the primitives of the images in the given class, (2) a fast and accurate primitive extraction algorithm, (3) the construction of a representative database of reasonable size, and (4) a way to compare primitives. Before discussing the details of these four requirements for the purpose of coding head and shoulders images similar to mug-shot and teleconference images, the process of human face recognition will be reviewed.

4.2. Recognition of Faces

People are assumed to process human faces by a system that is unique to face recognition, and different from all other systems for perceiving and recognizing non-facial objects [77]. This assumption is supported by the fact that people who suffer from prosopagnosia lose their ability to recognize previously familiar faces, but retain their ability to recognize non-facial objects. Also, studies have shown that faces are more easily remembered than other objects when presented in an upright orientation, but are more difficult to remember when inverted [77]. This points to a face system that is more sensitive to normally-encountered orientation than systems for processing other objects. Alternatively, faces are simply processed to a high degree automatically but without flexibility [77] .

Human face processing can be modeled by a system with three stages: encoding, storage, and retrieval. In the encoding stage, the facial information is encoded by the human visual system and sent to the memory, where the storage stage begins. The retrieval stage can be divided into two substages, recall and recognition. These stages are briefly discussed in the next four subsections, however, more details can be found in [63,77].

4.2.1. Human Face Encoding

When a facial image is presented to a subject, two types of facial information are encoded through the subject's vision system. The first is the wholistic or configurational structure and the second is the visual features of the face. However, no verbal description

of the face is encoded. It is assumed by most scholars that both types of information are encoded sequentially in a top to bottom order. Although both types of information are important in face perception, experimental studies have indicated that the wholistic information is better encoded than the visual features. The face is simply seen as a whole, with some features given more attention than others, which in turn may enhance their particular encoding. For instance, it is noticed that famous people are better recognized on the basis of their inner facial features such as the eyes, nose, and mouth, and strangers who are seen once are recognized equally as well from both inner and outer features. It is also known that the more we interact with people, the more attention we pay to their facial expressive areas over their non-expressive ones.

It is also a commonplace observation that faces from an unfamiliar race are more difficult to remember than own-race faces. One possible reason for this is that individuals from one race may pay attention to particular, discriminating facial features that are usually not so suitable for discriminating between faces of a different race. Hence, members of one race may fail to encode appropriate facial features of members of another race. Formal training in identifying members of another race may improve one's other-race face memory, at least temporarily, but it is difficult to produce any improvement in own-race face memory.

Among the many factors that may affect the encoding of facial features are the amount of time in which the face is seen, the type of face, and the mode in which the face is encountered. Studies have shown that increasing the viewing duration improves subsequent recognition memory, that unique faces are more easily encoded than typical

ones, and visual information are better encoded from live faces than from video recording or photographs.

4.2.2. Human Face Storage

Once a face has been successfully encoded, the trace of the face is stored in the neural nets of the brain. The neural nets consists of interconnected nodes that are capable of exciting or inhibiting each other in a parallel manner. The information is stored in the net as the weights of the connections between nodes. In the learning stage of the net, these weights are adjusted in an iterative manner until they represent the desired information. This memory is content-accessible, which means that an item is retrieved from the memory using the memory contents. This happens when the incoming input signal excites the suitable node, which in turn excites related nodes, inhibits irrelevant nodes, and so on. The face trace in memory is durable, or robust, and relatively unaffected by other faces that are bound to be seen during the intervening interval. Some faces may be relatively immune to distortion either from the passage of time or from retroactive interference. The more typical the face is, the most susceptible it is to distortion.

4.2.3. Human Face Recall

As previously mentioned, face retrieval is divided recall and recognition, both of which are of prime interest for forensic applications. The recall process involves conveying a facial image from the memory of one person to the memory of another

person with sufficient clarity that both persons can be said to share the same knowledge. Such a task can be achieved easily with information that is originally encoded as words, but the smell of a rose or the taste of a meal can not be so easily described. Similarly, the details of someone's face are difficult to convey when the face is present, and even more difficult when the face is absent.

Although verbal descriptions of faces are not suitable for recalling faces from memory, they are commonly used for such a purpose. Techniques such as *Photofit* and *Identikit* have been used for forensic purposes [63] to assist in the recall of criminal faces. Both techniques involve constructing composite images faces from a large collection of facial features. *Photofit* and *Identikit* enable eyewitness to convey at best only an approximate image of the original face.

4.2.4. Human Face Recognition

Recognizing faces is much easier than recalling them; Recognition generally involves less cognitive effort, and is exercised more often than the recall process. The current model for face recognition involves three major units. The first unit is the recognition unit, which fires in response to a familiar face, but recruits a new recognition unit when confronted with a novel face. The recognition unit passes its output to the second unit for analysis of the face. The result of this analysis is passed to the third unit, which is concerned with knowledge about the person concerned. Related information is then retrieved and the recognition process of the face completed

Three types of face recognition *tests* are used during criminal investigation:

(1) The witness searches mug-shot photographs in the hopes of seeing the culprit.

(2) A suspect is placed among a number of other people (usually similar to the suspect gross physical characteristics) for what is termed corporal identification (the lineup).

(3) The suspect alone is surreptitiously shown to the witness, in what is called the *showup*.

In the real world, a face seen at a test is unlikely to be identical to the one seen during an initial encounter, due to changes in hair style, expression, orientation, and so on. Factors such as familiarity with faces, unconscious transference (the phenomenon of wrongly categorizing familiar faces), context effect, and the pose of the face are very important in the recognition process of faces.

4.2.5. Knowledge-Based Image Coding Technique for Head and Shoulders Images and Face Recognition

Research has been conducted to determine which facial features in a photograph are more salient than others. In other words, research has attempted to answer the question of which facial features draw one's greatest attention. The result of such research is the basis for predicting face recognizability after a head and shoulders image is coded and decoded using the knowledge-based technique. These studies may be grouped into four categories [78]:

(1) Studies which have relied upon subjective reports or on verbal descriptions. In such studies, the subjects are asked what cues they employ in recognizing faces, or they are asked to describe a face to see which features they mention.

(2) Experiments which have examined either the ability of subjects to detect changes in facial cues, or the effect of such changes on recognition of faces. This approach is called the face *distortion* approach. The *face fragmentation approach*, is similar; only a part of the face is presented for recognition trials, and any feature that enables the subject to identify the target is said to be salient.

(3) Investigation of a subject's eye movements when examining faces. In this approach, the investigator carefully observes the eye movements of the viewer as he examines a face; if his gaze consistently falls on certain areas of the face, it might be reasonable to infer that those areas are the most salient and provide the most information.

(4) Studies which examine the properties underlying subjects' assessments of the similarity of faces. Since subjects cluster pictures into groups based on some underlying properties of the pictures, an analysis of such properties conveys important information on the saliency of most facial features.

Although the above four categories of study produce a variety of results which are highly dependent on the conditions under which the studies are conducted [77], a few general conclusions can be drawn. These conclusions are highly significant to the knowledge-based image coder. They can be summarized as follows.

(1) Inner facial features such as eyes, mouth, and nose are more salient than outer facial features such as forehead, ears, cheeks and jaw lines.

(2) The general shape of the head, including hair style, is the most salient feature, and that the eyes are the second most

(3) Central features such as eyes and nose are more salient than the mouth and ears.

In conclusion, we may order the facial features according to their degree of saliency as: hair > eyes > nose > mouth ≈ chin > ears. Based on this conclusion we may assume that replacing the ears, mouth and nose in a head and shoulders image with similar features from the database as is done in the knowledge-based image coding has little effect on face recognizability in coded/decoded images. However, replacing features such as the hair and eyes would have a significant effect on face recognizability in coded/decoded images. This effect, however, can be reduced with careful selection of the entries of the database, such that the recognizability of the decoded faces is maintained.

4.3. Primitives of Head and Shoulders Images

The primitives of head and shoulders images are well-defined; they correspond to image segments containing facial features of the human face. Therefore, an image segment which contains an eye, nose, mouth, ear, chin, mustache, eyebrow, or hair section could be considered a primitive of a head and shoulders image. A given primitive may have more than one state. For instance, the mouth may be closed, slightly opened, opened, or widely opened.

As seen in the previous chapter, facial features can be easily located in a given input image, based on a simple model in which the shape of the human head in a photograph is approximated by an ellipse. The geometry of the ellipse is combined with

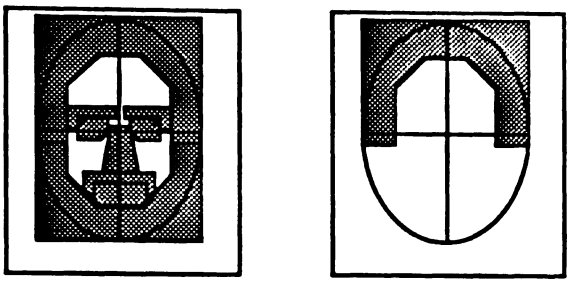
knowledge from the art of drawing the human head to derive simple mathematical equations to quickly and accurately locating the facial features (see Eq. 3.1-3.3). The orientation of the ellipse and the calculated positions of the facial features are used in conjunction with a set of filters, such as those depicted in Fig. 4.1, to extract the facial features.

4.4. Filter Set for Extracting the Facial Features

The facial features can be extracted from the input image by using a set of filters similar to those used by Wiederhold [70]. Some possible shapes for these filters are shown in Fig. 4.1. Each filter can be thought of as a mask of ones and zeros. The pixels corresponding to the shaded areas in Fig. 4.1 have the value of 1, and those outside the shaded areas have the value of 0. When a filter is multiplied by the input image, the values of the image pixels that correspond to the shaded area in the filter do not change, since they are multiplied by 1. However, the values of the other pixels become zero. This is the basis for primitive extraction from the input image.

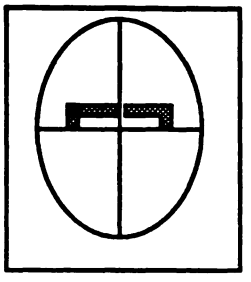
In order to ensure that a given filter will accurately encompass a desired feature of the input image, the filter's center of operation, size, and orientation are determined by applying the facial feature location algorithm of chapter 3 to the input image. Before applying the algorithm, the locations of the corners of each filter with respect to a normalized ellipse are first determined from Eq. 3.1a, 3.2b, 3.2c, 3.3b and 3.3c; the center of the normalized ellipse is at the origin and the length of its major and minor axes are 1 and $\frac{2}{3}$, respectively. The center of the ellipse is then shifted to the center of the face in the input image, and the lengths of the major and minor axes and the coordinates of the

filter corners are determined by applying the facial feature location algorithm to the input image. Finally, the corner coordinates are rotated, such that the orientation of the ellipse corresponds with the orientation of the head in the input image, the filters accurately encompass the proper facial features, and the primitives of the input image can be accurately extracted.

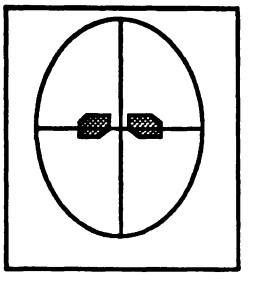


a. All Filters

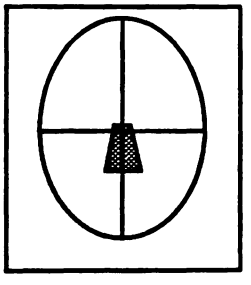
b. Hair Filter



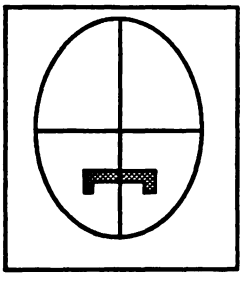
c. Eyebrows Filter



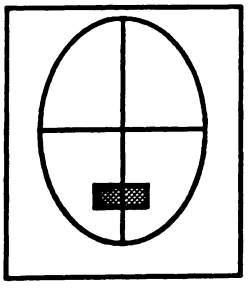
d. Eyes Filter



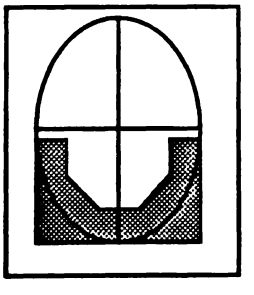
e. Nose Filter



f. Mustache Filter



g. Mouth Filter



h. Chin Filter

Fig. 4.1 Filter Set for Facial Feature Extraction

4.5. Primitive Matching

Primitive matching is the process of finding the best match of an extracted primitive to the primitives that have been pre-stored in the database. It is a major obstacle to realizing the knowledge-based image coding technique, since it is a more subjective than mathematical process. However, mathematical error measures such as the mean-square-error (*mse*) are usually adequate for this purpose.

If $f(x,y)$ denotes a pixel in a primitive in the database and $\hat{f}(x,y)$ denotes the corresponding pixel in a primitive extracted from the input image, then the *mse* is defined as

$$mse = \frac{1}{n} \sum_{(x,y)\epsilon I}^{n} (f(x,y) - \hat{f}(x,y))^2 \qquad (4.1)$$

where n is number of pixels in the primitive, and I is the (x,y) coordinates set of the pixels in the primitive. Since there may be a slight misalignment when comparing the extracted primitive with a primitives from the database, the primitives are displaced relative to each other one or two pixels in the vertical and horizontal directions until they are aligned. The extracted primitive is compared to each primitive in its group (e.g. noses) in the database. The primitive from the database which produces the least mean-square-error is then taken as the best match, provided that this mean-square-error is less than a specified threshold.

Another method of primitive matching utilizes a dedicated neural network. Stonham [79] developed a self-adapting machine called *Wisard* as a general purpose pattern recognition system. He also described its application to human face recognition. The central component of *Wisard* is a neural network that adapts to a particular training set during a learning process. The network is then able to recognize patterns that are very similar to those of the training set. Therefore, such a network could be used to match an image primitive of a given type to a similar primitive in the database. Figure 4.2 depicts the structure of the neural network use in *Wisard*. Note that each input is weighted before it is combined with the other inputs, and the sum is thresholded. The values of these weights are determined through the learning process, in which the network is presented with known inputs from a training set, and the weights are iteratively adjusted until the desired output is obtained for all the members of the training set. The *Wisard* machine is of parallel nature and is easily fabricated using currently available VLSI technology.

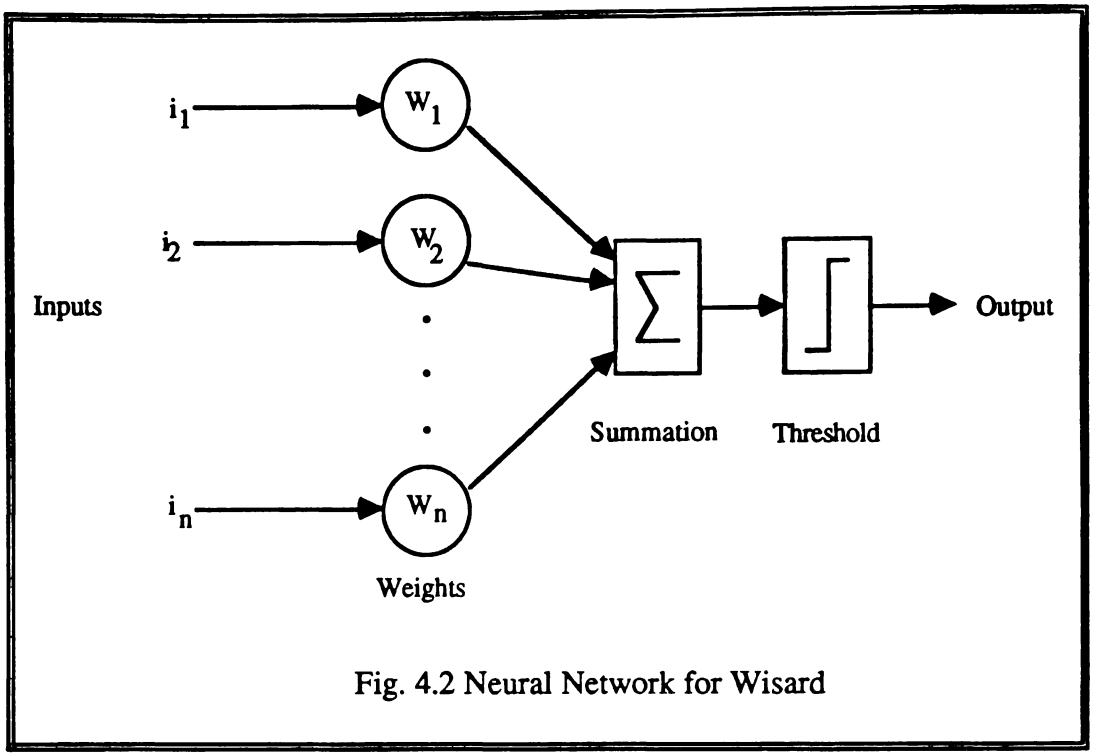


Fig. 4.2 Neural Network for Wisard

In both matching methods, it is assumed that the extracted primitives and the database primitives have the same number of pixels. As will be seen in section 4.7, the database primitives are normalized to a specific size before they are stored in the database. Therefore, before any primitives can be compared, a database primitive must be re-scaled, such that its number of pixels is the same as the number of pixels in the extracted primitive. Since database primitives are stored in a digital format, re-scaling them requires interpolation between pixels. Two methods of interpolation are used. The first is the simple linear interpolation; the second involves fitting a two-dimensional surface function to the pixels in the re-scaled primitive. The missing pixels are then calculated from the equation of the surface. Linear interpolation is simpler and requires less computation than surface fitting, but is less accurate than surface fitting. Interpolation accuracy is not crucial to the feature matching process, but, it is very essential in the feature reconstruction process.

4.6. Primitive Projection

Matting is a technique for combining two different images such that the resulting image looks like a photograph of two stacked transparences. Each pixel in each image is assigned a degree of transparency or opaqueness before the images are combined. Suppose image *A* is to be added to base image *B*, and image *B* will become the new output image. If a pixel in image *A* is defined as transparent, then the corresponding pixel in image *B* is unchanged. The more opaque the pixel in *A* is, the more it is added to picture *B*, until, when *A* is totally opaque, the pixel in *B* is completely replaced with the value of the pixel in *A*.

This matting process is used to project a decoded primitive onto the base image (initially, a prototype image, see next section). In this case, the primitive to be projected is assigned a full degree of opaqueness, while the base image is assigned a full degree of transparency, so that when the primitive is added to the base image, the primitive will completely replace the corresponding facial feature of the base image.

4.7. Prototype Image

A prototype image is thought of as the best example of a set or category of a sub-class of images. This image represents attributes of the sub-class better than any other image in the same sub-class. It also has the highest probability of being recognized as an entity that has been seen when other members of the sub-class are shown to a subject, even if the prototype image has not actually been shown. A prototype image can be

composed from features that have been extracted from a set of images in the given sub-class. The average model, attribute frequency model, and interval storage hypothesis are three different methods that can be used for selecting the most representative features to be extracted [80].

In the average model, features are selected such that they represent the average value for each dimension of the feature. For example, the average Euro-American nose might lean towards the narrow feature values, while the average Afro-American nose might lean towards the broad feature values. In both these sub-classes, the prototype image would be assigned its corresponding average nose. The attribute frequency model is analogous to the average model, except that modal values replace average values for each dimension of feature variation. In the interval storage hypothesis, features are selected on the basis of their effectiveness in making a subject envision other features of the same type. For example, any displayed feature represents to subjects an interval or range along one of the feature's dimension. The feature which produces the largest range is selected for the composition of the prototype image. Experimental results have shown that facial prototypes are best formed utilizing the attribute frequency model [80] .

4.8. Post-Processing

The above matting process does not completely blend the primitive and the base image. The average intensity of the pixels inside the primitive might be different from that of the pixels in the base image, which causes the primitive to be visually distinguishable from the rest of the image. This effect can be reduced by re-scaling the intensity value of the pixels in the primitive, such that the average intensity value is compatible with that in

the base image. Unfortunately, scaling does not completely eliminate the artifacts around the boundaries of the primitive. Therefore, it is also necessary to smooth the transition region between the primitive and the base image.

Since the boundaries of each primitive are well defined, a simple averaging filter can be designed to smooth the transition region between the primitive and the rest of the image. The average filter should be at least a 3 by 3 moving window that traces the boundary of the primitive and replaces the value of each pixel on the boundary with the average of the values of the pixels in the 3 by 3 window around the boundary pixel. Smoothing one single boundary pixel is not enough to eliminate artifacts. Therefore, for a better smoothing result, adjacent pixels that are interior and exterior to the primitive should also be smoothed. To assure a smooth transition region, two interior pixels and two other exterior pixels are smoothed in addition to the boundary pixel.

4.9. The Database

The database that represents the universal set **S** (see chapter 2) for head and shoulders images can be easily constructed from a collection of primitives extracted from prototype photographs. These photographs must be selected to be highly-representative of the class of head and shoulders images. A representative database of reasonable size would contain about the same number of primitives as the *Identikit* and *Photofit* kits, each of which contain around 150 hair sections, 60 ears, 100 eyes and eyebrows, 50 noses, 100 mouths, and 70 chins [63].

An example database structure for head and shoulders images is depicted in Figure 4.3. The primitives are normalized with respect to size and orientation, and grouped in the database under their respective primitive groups, e.g., hair, eyes, nose. Each group should include primitives which represent the various shapes and states that may be assumed by its corresponding facial feature. For instance, the eye group should contain picture segments containing eyes of different people. These eyes should be in various states, e.g., closed, slightly opened, and fully opened. The database should also contain a group of prototype images (see section 4.6) to be used as base images in the reconstruction process.

Database for Head and Shoulders Images											
Face	Hair	Eyebrows		Eyes		Nose	Mouth	Ears		Chin	Mustache
		Left	*Right*	*Left*	*Right*			*Left*	*Right*		
FA1	HA1	LEB1	REB1	LE1	RE1	NO1	MO1	LER1	RER1	CH1	MU1
FA2	HA2	LEB2	REB2	LE2	RE2	NO2	MO2	LER2	RER2	CH2	MU2
.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.
FAn	HAn	LEBn	REBn	LEn	REn	NOn	MOn	LERn	RERn	CHn	MUn

Fig. 4.3 Structure of the Database for Head and Shoulders Images

Such a database structure of the database facilitates searching the database for the best match of a primitive. Since the type of primitive extracted from the input image is

known beforehand, it is not necessary to compare the extracted primitive to all the primitives in the database, but is enough to compare the primitive with those in its corresponding group. Since each group contains a relatively small number of primitives, the time required to search the database is reduced.

For a very large database, the search time can be reduced further by constructing a sub-database, which is referred to as a cache database. In a cache database only the most frequently used primitives are stored. This idea is similar to that of cache memory, which is used to speed up data fetches from the main memory in most modern computers. Therefore, the primitives in each group in the main database are structured into fixed-size pages. One page from each group fits completely into the cache. The contents of the cache database are updated whenever the number of inquiries for primitives not in the cache exceeds a certain pre-defined threshold. The updating process is achieved by replacing the contents of the cache database with the most frequently used page in the main database, over the most recent interval of time. The same updating procedure is followed at the receiver. Therefore, both coder and decoder keep up with the statistics of the database inquiries, and no overhead information is transmitted for the purpose of updating the database at the decoder.

The coder and decoder must keep up with long-term statistics as well. These statistics are essential in updating the main database. Recall that a copy of an extracted primitive is added to the main database whenever the coder can not find a best match for in the main database. The updating procedure, in this case, is achieved by adding the new primitive to the end of the database, if enough storage space is available; otherwise, the new primitive has to replace another primitive that is already in the database.

Determination of the database primitive to be replaced is based on the frequency of use of each database primitive during the recent history of the coder. Therefore, both coder and decoder have to keep an up-to-date history on the use of each primitive in the database. Then, the least-used primitive is replaced with the new primitive.

Singular-value-decomposition [9] can be used to construct a compact database. Since facial primitives of the same type look alike, it is possible to store the primitives of each group in terms of a single primitive from that group. First, a prototype primitive is selected to represent each group. Then, the singular-value-decomposition of the selected primitive is calculated. The basis vectors of this primitive are used to transform the other primitives in the group, and only the differences between the transformation coefficients of these primitives and those of the prototype primitive are stored in the database.

Let the matrix **F** represent the prototype primitive of a certain group in the database. The singular-value-decomposition of **F** is given by

$$\mathbf{A}^{1/2} = \mathbf{U}^{T}\,\mathbf{F}\mathbf{V} \tag{4.2a}$$

$$\mathbf{F} = \mathbf{U}\mathbf{A}^{1/2}\mathbf{V}^{T} \tag{4.2b}$$

Where **U** and **V** are unitary matrices containing eigenvectors of the matrices $\mathbf{F}\mathbf{F}^{T}$ and $\mathbf{F}^{T}\mathbf{F}$, respectively, and **A** is a diagonal matrix whose terms are the corresponding eigenvalues. The singular-value-decomposition may be regarded as a symbolic

representation of a primitive in terms of a set of basis plane luminance patterns $\mathbf{u}_j\mathbf{v}_j^T$ according to the following relation

$$\mathbf{F} = \sum_{j=1}^{N} \lambda^{1/2}(j)\ \mathbf{u}_j\mathbf{v}_j^T \tag{4.3}$$

where $\lambda(j)$ is the j^{th} diagonal element of $\mathbf{A}$ and $\mathbf{u}_j$ and $\mathbf{v}_j$ are column eigenvectors of $\mathbf{U}$ and $\mathbf{V}$, respectively.

If $\mathbf{F}'$ represent a general primitive from the same group as the prototype primitive $\mathbf{F}$, then $\mathbf{F}'$ can be decomposed into the matrix $\mathbf{F}$ and an error matrix ε. The components of the matrix ε are very small. $\mathbf{F}'$ can be expanded in terms of the unitary matrices $\mathbf{U}$ and $\mathbf{V}$ of the matrix $\mathbf{F}$. Therefore,

$$\mathbf{F} = \mathbf{F}' - \varepsilon \tag{4.4a}$$

$$\mathbf{A}^{1/2} = \mathbf{U}^T(\mathbf{F}' - \varepsilon)\mathbf{V} \tag{4.4b}$$

$$\mathbf{A}^{1/2} = \mathbf{U}^T\mathbf{F}'\mathbf{V} - \mathbf{U}^T\varepsilon\mathbf{V} \tag{4.4c}$$

$$\mathbf{A}^{1/2} + \mathbf{U}^T\varepsilon\mathbf{V} = \mathbf{U}^T\mathbf{F}'\mathbf{V} \tag{4.4d}$$

$\mathbf{A}$ is the diagonal matrix of the eigenvalues of the prototype primitive $\mathbf{F}$. As in the case of unitary transforms in general, the energy in the error matrix ε will be packed into few

coefficients. These coefficients can be selected and stored in the database instead of the original values of the primitive $\mathbf{F}'$. Hence, a compression ratio on the order of 10:1 can be achieved in representing the primitives in the database, which means reduction in the required storage space.

To retrieve a primitive from a compact database, the original form of the primitive must be re-calculated from its error matrix. The process is simple, since

$$\mathbf{U}^T \mathbf{F}'\mathbf{V} = \mathbf{A}^{1/2} + \mathbf{U}^T \varepsilon \mathbf{V} \tag{4.5a}$$

$$\mathbf{U}(\mathbf{U}^T \mathbf{F}'\mathbf{V})\mathbf{V}^T = \mathbf{U}(\mathbf{A}^{1/2} + \mathbf{U}^T \varepsilon \mathbf{V})\mathbf{V}^T \tag{4.5b}$$

$$\mathbf{U}\mathbf{U}^T \mathbf{F}'\mathbf{V}\mathbf{V}^T = \mathbf{U}\mathbf{A}^{1/2}\mathbf{V}^T + \mathbf{U}\mathbf{U}^T \varepsilon \mathbf{V}\mathbf{V}^T \tag{4.5c}$$

then

$$\mathbf{F}' = \mathbf{U}\mathbf{A}^{1/2}\mathbf{V}^T + \mathbf{U}\zeta\mathbf{V}^T \tag{4.5d}$$

$$\mathbf{F}' = \mathbf{F} + \mathbf{U}\zeta\mathbf{V}^T \tag{4.5e}$$

where $\mathbf{A}^{1/2}$ is the singular-value-decomposition of the prototype primitive $\mathbf{F}$, and $\zeta = \mathbf{U}^T \varepsilon \mathbf{V}$ is the singular-value-decomposition of the error matrix. Therefore, a primitive can be retrieved from the database by calculating the inverse of the singular-value-decomposition of its error matrix and adding the result to the prototype primitive.

Other unitary transforms, such as the discrete cosine, Fourier, and Hadamard transforms [13], can be used instead of singular-value-decomposition. However, these

transforms would not achieve the same compactness that is achieved with singular-value-decomposition. This is because each of these transforms is optimal for a specific class of images, but would perform sub-optimally for any other class. For instance, the cosine transform is the Karhunen-Loève transform for a first-order Markov images as the adjacent element correlation approaches unity [81]. Hence it is the optimal transform for that class only. The singular-value-decomposition of an image, on the other hand, is optimal for diagonalizing the image and achieving high compression. When the resultant eigenvectors are used with images that are extremely close to the original image, the performance is very close to optimal.

4.10. Coding and Decoding a Head and Shoulders Image

The coding process of a still head and shoulders image is summarized in the flowchart of Fig. 4.4. The process starts with reading the input image, which is assumed to be a head and shoulders image. This image is matched to the database to find the index of a prototype image from the database that can serve as a base image at the receiver. The index of the prototype image is transmitted. The edges of the input image are then detected and thinned, and an ellipse is fitted to the outer-most contour of the head in the image (as was explained in chapter 3). The parameters of the best fit ellipse along with the set of filters shown in Fig. 4.1 are used to extract the primitives of the image, and the parameters of the ellipse are transmitted. Each primitive is then matched to the database and the index of the best match is transmitted. If an acceptable match is not found in the database, the primitive is first transmitted then added to the database. This process is repeated for all the primitives in the image.

When the primitive itself is to be transmitted, it is necessary to reduce the total number of bits to be transmitted. A standard coding technique such as transform or predictive coding is used to achieve the necessary compression ratio. Alternatively. singular-value-decomposition can be used in a process analogous to that used in constructing a compact database. The primitive to be transmitted is first expanded in terms of the basis vectors of a prototype primitive from the database (see Eq. 4.4d). The prototype primitive must be of the same type as the primitive to be transmitted. Then, the difference between the singular-value-decomposition of that primitive and the prototype primitive is calculated. Finally, only those coefficients of high energy in the difference image are quantized and transmitted; low energy coefficients are discarded.

At the receiver, the primitive is re-constructed by calculating the inverse transform of the received coefficients, after adding to it the diagonal matrix of the prototype primitive (see Eq. 4.5d). This matrix and the basis vectors are already available, since the receiver possesses a duplicate database. The discarded coefficients are assumed to be zero.

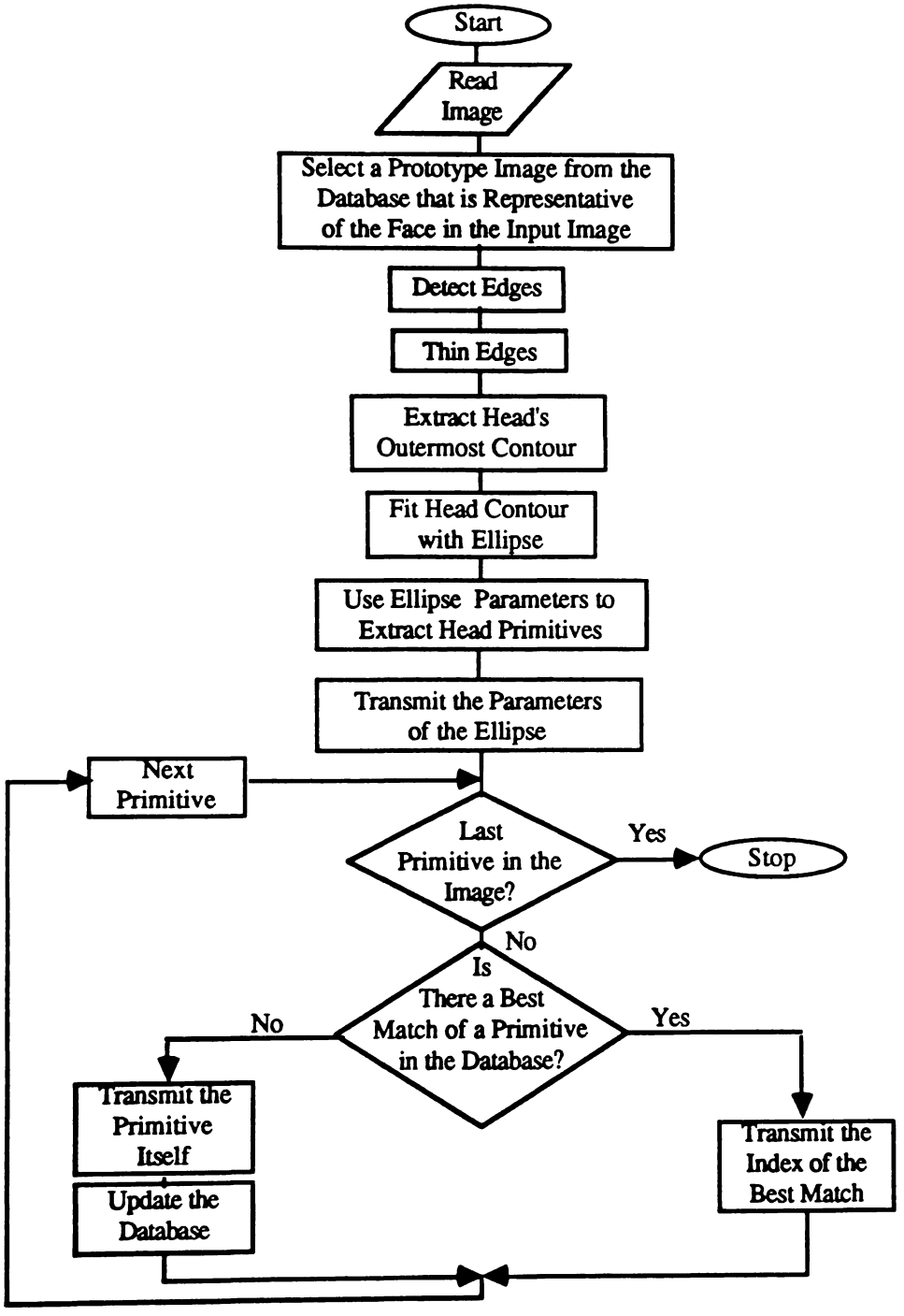


Fig. 4.4 Knowledge-Based Coding Process for Still Pictures

The decoding process is summarized in the flowchart of Fig. 4.5. It starts with fetching a prototype image from the database. This prototype image is modified for use as a base image on which the primitives will be projected; the lengths of the major and minor axes and the orientation angle of the best fit ellipse are used to scale and rotate the head in the prototype image, such that its size and orientation match the size and orientation of the head in the input head and shoulders image.

Next, the primitives of the transmitted image are decoded for projection onto the base image. The decoder decides whether a received message is an index for a primitive in the database, or a primitive itself. If it is an index, then the corresponding primitive is fetched from the database and re-scaled; the length of the major axis of the ellipse is used to re-scale the primitive. Otherwise, the primitive itself is decoded, as described previously, by taking the inverse singular-value-decomposition of the received coefficients, and the resulting primitive is added to the database. In either case, the intensity values of the pixels within the primitive are re-scaled so that the average intensity value inside the primitive is compatible with the average intensity value of the entire base image. Finally, the primitive is rotated, directed to the proper location, and projected onto the base image using the previously described matting process. The orientation angle of the best fit ellipse is used in rotating the primitive; its location is calculated with Eq. 3.1-3.3. This decoding process is repeated for all the primitives in the image, and after all the primitives are projected onto the base image, the image is post-processed with a 3 by 3 smoothing filter as described previously.

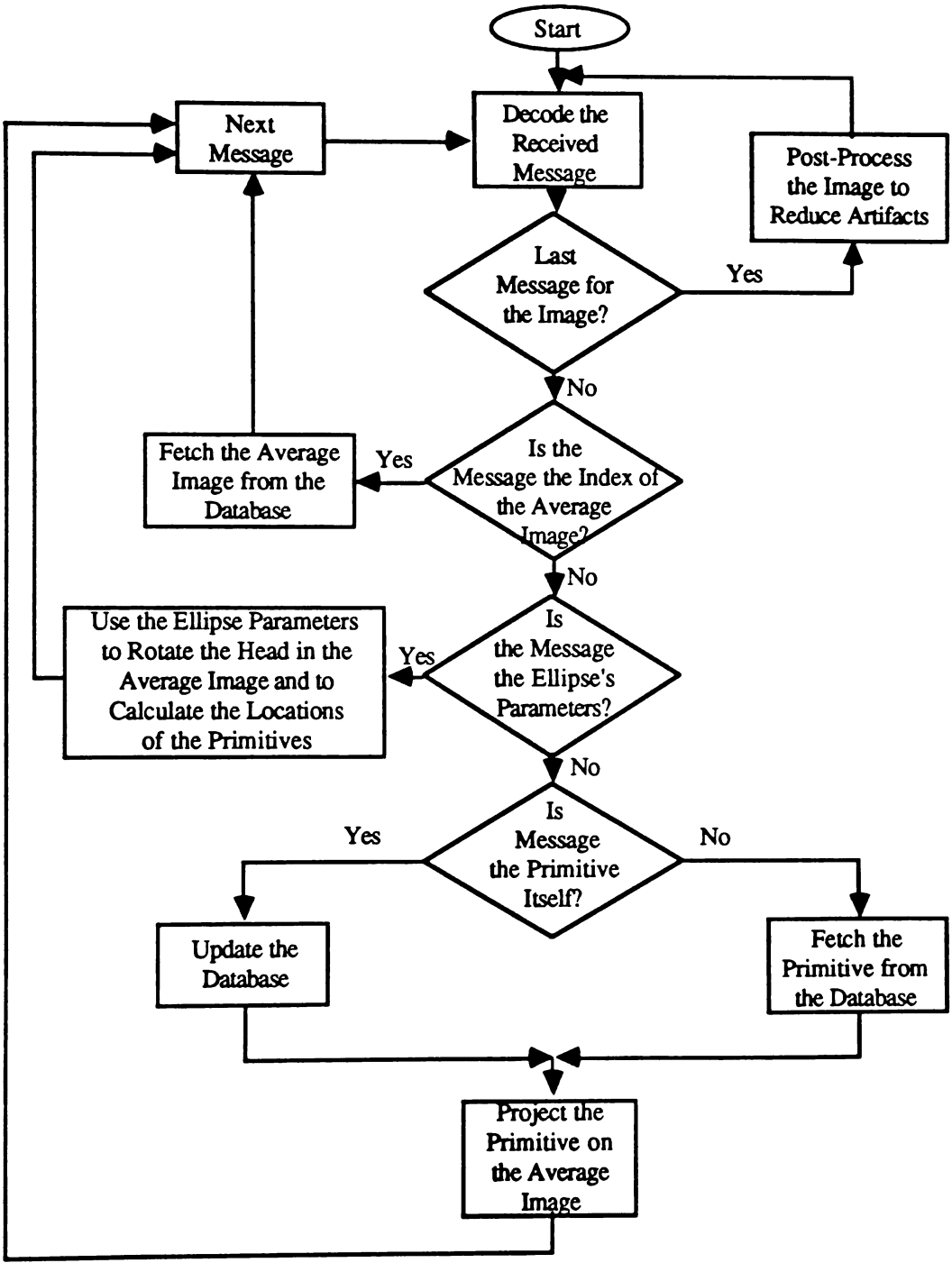


Fig. 4.5 Knowledge-Based Decoding Process for Still Pictures

CHAPTER 5

KNOWLEDGE-BASED IMAGE CODING TECHNIQUE FOR TIME VARYING IMAGE SEQUENCES

5.1. Video Teleconferencing and Picture Phone Images as Applications

Video teleconferencing and Picture Phone are two major applications of the new knowledge-based image coding technique. Transmission of a time-varying image sequence for such applications requires a very large bandwidth and is very costly. For instance, consider transmission of a video signal at 30 frames per second, with an image size of 512x512 pixels, and each pixel quantized to one of 256 gray levels. A transmission rate of over 64 Mbits/s is required. This bandwidth requirement limits the transmission of such a signal to satellite and fiber optics channels, which are not economical and are not always available to small business firms. Transmitting the signal over a low-bit-rate channel (e.g., less than 64 Kbits/s, as in local area networks) is more economical and convenient, but real-time transmission of such a video signal is almost impossible. In fact, it would take more than half a minute to transmit a single frame through such a channel. For real-time transmission through a 64 Kbits/s channel, the signal has to be compressed over 1000 times before transmission.

A simple way to achieve such compression is by employing first generation image coding techniques (see chapter 1) after reducing the frame rate from 30 frames/sec to 15

frames/sec, and after reducing the image resolution via sub-sampling and coarse quantization. Unfortunately, frame rate reduction causes motion jerkiness and resolution reduction produces significantly degraded and visually unpleasant images. However, certain facts about the images in teleconference and Picture Phone applications can be exploited in the knowledge-based image coding technique to avoid motion jerkiness and still produce good image quality at the desired compression ratio. These include the fact that teleconference and picture phone images are normally head and shoulders images, which are normally viewed from a distance, and the fact that the viewer is more interested in the general appearance of the objects in the image than in the finer details.

In this chapter the knowledge-based image coding technique will be applied to head and shoulders images in order to achieve the required compression ratio. The feature extraction, matching, and projection processes which were discussed in chapter 4 will be used for this purpose.

5.2. Coding and Decoding a Time-Varying Head and Shoulders Image Sequence

Head and shoulders images are typical in both teleconference and Picture Phone applications. As in the case of still (i.e., non-time-varying) head and shoulder images, the hair sections, ears, eyes, eyebrows, nose, mustache, mouth, and chin are considered to be the primitives of these images. Filters designed to extract these primitives are shown in Fig. 4.1. Since the upper part of the each ear is often covered with hair, the ears and the hair are joined into one compound primitive. Also, the eyes and the eyebrows are joined to form one compound primitive. This compound primitive is more appropriate than two

separate primitives because the motion of the eyebrows is highly synchronized with the motion of the eyes. Similarly, the mustache and mouth are joined to form one compound primitive, rather than two separate primitives.

In general, the motion in a time-varying head and shoulders image sequence is limited to the movement of the head, eyes, mouth and jaws, and hands (if hands appear in the picture). The motions of these features tend to be repetitive; i.e, as the person in the image speaks, his eyes and mouth open and close, his head moves up and down or to the sides, and his hands move in synchronization with his speech. Therefore, their corresponding primitives can be assigned a limited number of states that are easily collected into a database of image primitives. In fact, some of these primitives assume only one state, since they remain virtually static throughout the entire interval of transmission. The nose is an example of a one-state primitive. The apparent shape of the nose, however, changes from one frame to the next, depending on the global motion of the head; i.e., as the head moves up and down or from one side to the other, different nose shapes appear in the image.

Definition of the head motion is simple; it is assumed to be limited to rotation around an axis that is perpendicular to the image plane (see chapter 3). Therefore, each frame in the image can be derived from the previous frame by simple rotation around this axis of the head, and subsequent updating of any primitives that have encountered high deformation from one frame to the next. It should be noted that although the eyes and the mouth are the primitives that usually deform the most as the person speaks, their deformation does not necessarily occur between every two frames in the image sequence. Some primitives in some frames might remain stable, without any motion or deformation.

These primitives are detected during the coding process and ignored. The mean square error (Eq. 4.1) between the primitive in the present frame and the same primitive in the previous frame is used as a measure of the deformation.

The coding process for a time-varying head and shoulders image sequence is similar to the coding process for still pictures of the same type. The process for a time-varying images sequence is depicted in the flowchart of Fig. 5.1. A database similar to the one described in section 4.7, but with fewer types of primitives, is used in the coding process. This database and its construction are discussed in section 5.3. The coding process itself occurs in two stages. In the first stage, the image primitives are located and extracted, and in the second, they are coded and transmitted.

As in the case of still pictures, to locate the primitives, it is first necessary to fit the outermost contour of the head with an ellipse Therefore, the edges in the image are detected with an edge detector, thinned, and finally fitted with an ellipse function. The parameters of the ellipse, which best fits the outermost head contour, are used to locate the facial features of the input image. They are transmitted so that they can later be used at the receiver to determine the locations of the primitives in the output image.

The filter set shown in Fig. 4.1 is then used to extract the primitives from the input image. The degree of deformation of each extracted primitive is determined by computing the *mse* , as defined by Eq. 4.1, between this primitive and the corresponding primitive in the previous image. If the *mse* error is below a specific threshold, then the primitive is assumed to have the same state as in the previous frame, and hence it is not necessary to replace or change the primitive in the previous frame. In this case, the primitive is ignored and a flag is transmitted to the receiver to indicate no attempt is

needed to modify the primitive. On the other hand, if the *mse* is above the specified threshold, then the primitive in the previous frame, $\mathbf{F}_{i-1}$, has to be updated at the receiver, such that it represents the primitive in the present frame, $\mathbf{F}_i$. Therefore, the primitive in the present frame, $\mathbf{F}_i$, is matched to the database to find a best match that can replace the corresponding primitive in the previous frame, $\mathbf{F}_{i-1}$. If a best match is found, its index in the database is transmitted. If not, the primitive itself is coded using the singular-value-decomposition of a representative primitive of the same type, as described in section 4.8. This 2-stage coding process is repeated for each frame in the image sequence.

The decoding process for a time-varying image sequence, as depicted in Fig. 5.2, differ slightly from the decoding process for still pictures (see chapter 4). This process starts by decoding the parameters of the ellipse that best fits the outermost head contours in the present frame. These parameters are then used to rotate the head in the previous frame and adjust its location, such that it corresponds with the head in the present frame. The previous frame is assumed to be available at the receiver and is stored in a special buffer for this purpose. The ellipse parameters are also used to calculate the positions of the primitives. Then the transmitted primitives are decoded either from the database or by calculating the inverse singular-value decomposition, and then projected onto the previous frame in the same manner described for still pictures in section 4.5. Finally, the image is post-processed using an averaging filter, as described in section 4.6.

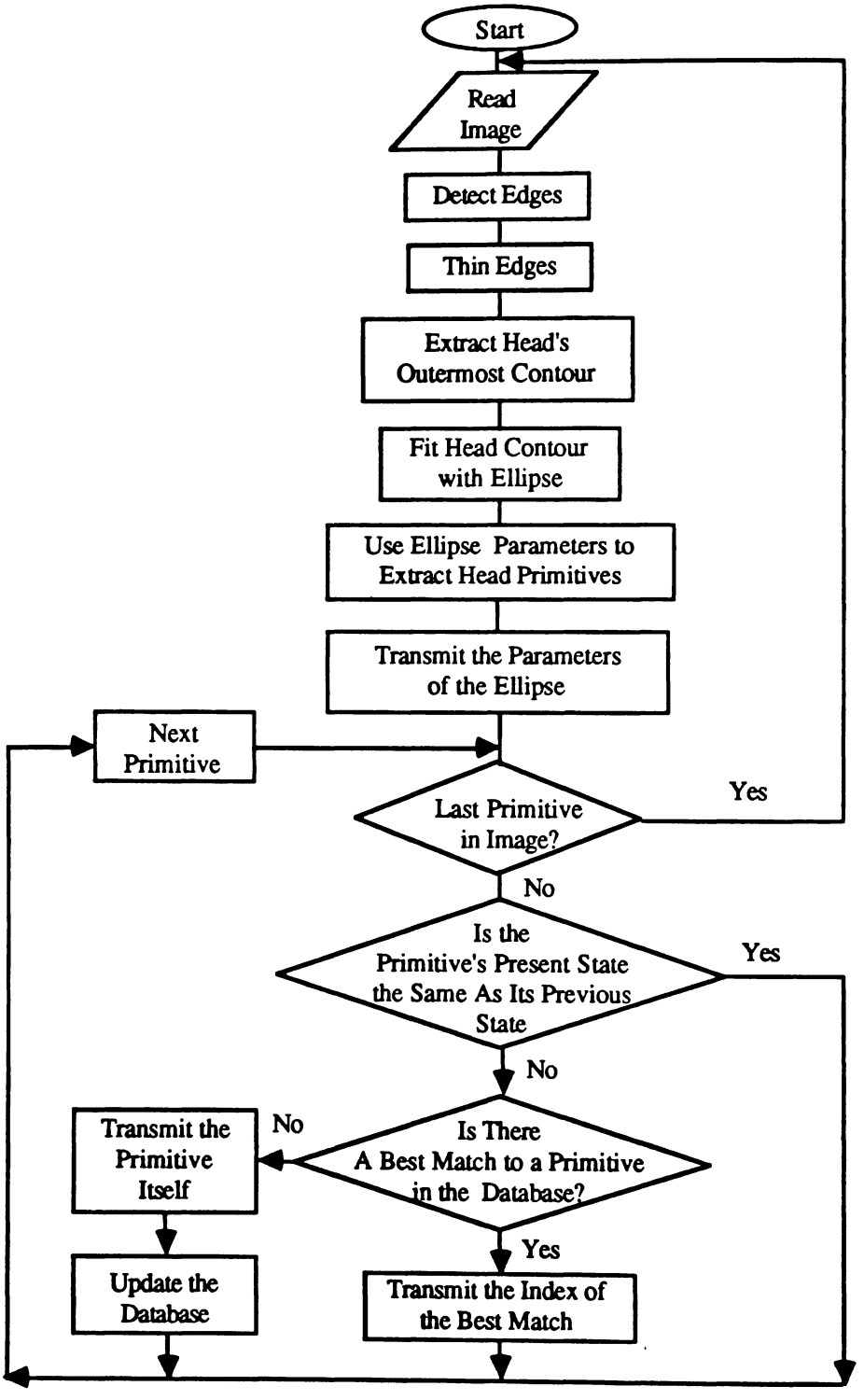


Fig. 5.1 Knowledge-Based Coding Process for a Time-Varying Image Sequence

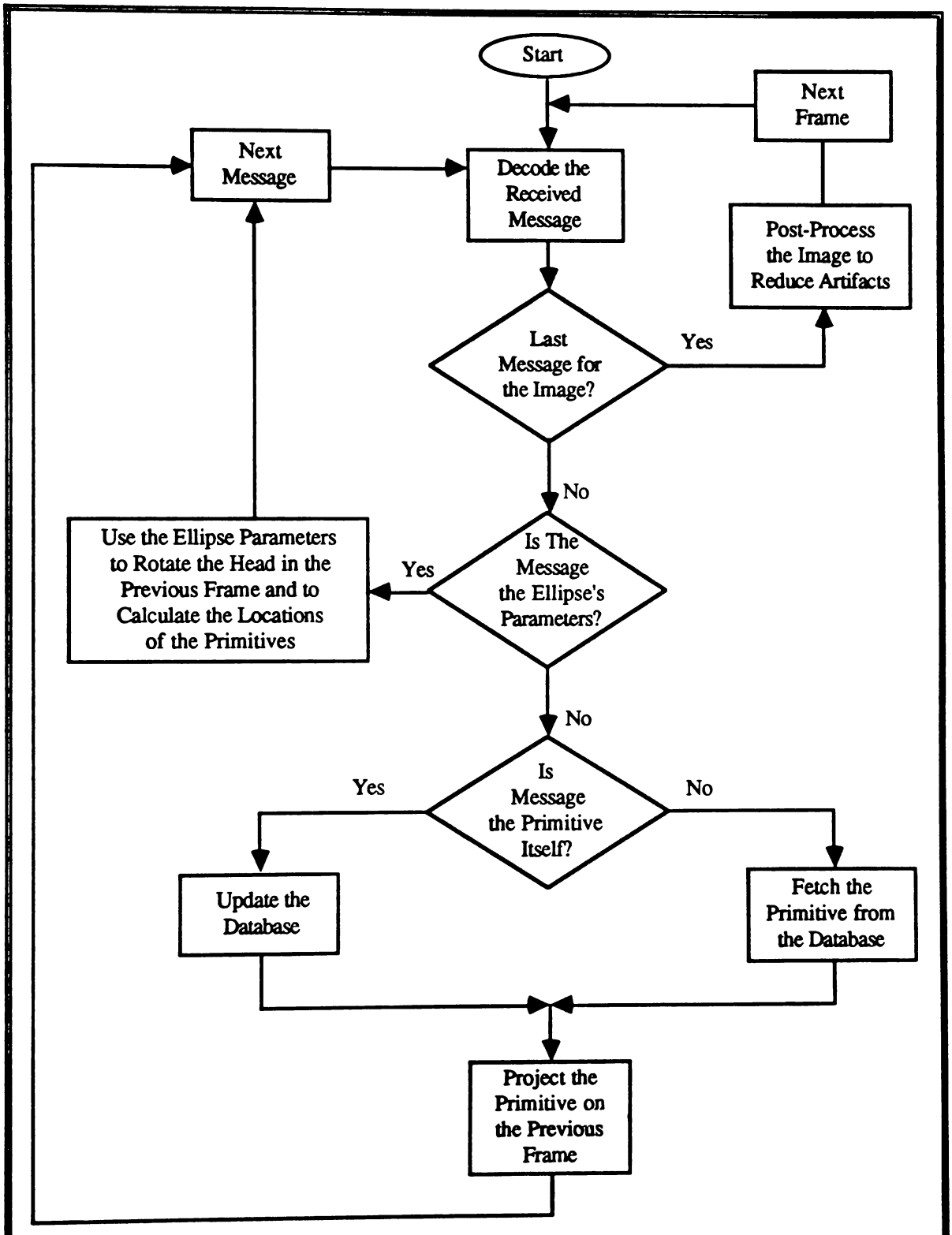


Fig. 5.2 Knowledge-Based Decoding Process for a Time-Varying Image Sequence

5.3. On-Line Database Construction

The structure of a database for a time-varying head and shoulders image sequence is similar to that which was described in the previous chapter for still pictures. The major difference between the two types is that they have different numbers and types of primitives. Figure 5.3 depicts the structure of a suitable database for time-varying head and shoulders image sequence Note, only three types of primitives are stored in the database, namely, the eyes and eyebrows, nose, and mouth and mustache. Primitives such as average images, ears, and hair sections need not be stored in the database. Since the first frame in the image sequence is assumed to be available at the receiver (it is transmitted at the beginning of the transmission), it replaces the average frame in the still pictures case. Moreover, the ears encounter almost no locomotion, but move with the rest of the head, therefore, they need not to be updated from the database. The ears positions and shapes are determined as the head in the previous frame is rotated to match the orientation of the head in the present frame.

Database for Time-Varying Head and Shoulders Image Sequence			
Eyes and Eyebrows		Nose	Mouth and Mustache
Left	*Right*		
LE&B1	RE&B1	NO1	MO1
LE&B2	RE&B2	NO2	MO2
.	.	.	.
.	.	.	.
LE&Bn	RE&Bn	NOn	MOn

Fig. 5.3 Structure of the Database for Time-Varying Head and Shoulders Image Sequence

A full database is not required to be available before starting transmission. In fact, the system, starts with an empty database that is filled during the transmission of the first few frames. Since, in the beginning, the database is empty, the system has to fill the database with every extracted primitive and transmit the primitive itself. Recall, the system adds a new primitive to the database whenever that primitive has no best match in the database, which is the case with an empty database. After the system fills the database, the stored primitives are re-used in the normal operation mode, for the rest of the transmission period.

The initial performance of the system while the database is being constructed reduces to the same level as that of traditional image coding techniques. However, the average performance of the system over the entire transmission period is not seriously affected. The achievable compression ratio, in the first few frames, drops from 1000:1 to

10:1. However, when this reduction is averaged over the entire transmission period, the compression ratio is still around 1000:1.

5.4. Accumulation of Errors and Frame Refreshment

Since the basic idea of the knowledge-based system is to transmit images that are perceived by the viewer as the originals, but are not actually not the originals, there will be error accumulation. After a period of time, this error will cause an intolerable image degradation. However, the error can be reduced by refreshing the frames periodically, e.g., a complete frame is transmitted every three seconds without compression. This complete frame replaces the base image, which is the image to be updated and modified to obtain the next frames of the sequence.

The frame-updating process can be achieved gradually. Instead of transmitting all the pixels in the update frame at the same time, for this purpose, the pixels from a three lines of the frame are transmitted as 8-bit PCM in each frame. These three lines are evenly spaced in the image raster and are moved up each frame-time so that the entire frame is updated after a few seconds. If the image contains 512 lines, then the entire frame would be updated in six seconds. Thus, a stationary portion of the picture acquires the quality obtainable with 8-bit-PCM less than six seconds after it became stationary.

5.5. Buffer Requirements and Control

It should be noticed that the output bit-rate of the knowledge-based image codec changes according to the contents of the database and the motion in the input image. For

instance, only few bits are transmitted when a best match of a primitive is found in the database; but an excessive amount of bits are transmitted for a primitive when no best match is found. Also, a primitive in the image is not transmitted if its state does not change from one frame to another. In order to transmit such a variable output bit-rate over a constant bit-rate channel the coder's output has to be held temporary in a first in-first-out buffer that can accept input at a nonuniform rate, but produces output to the channel at a uniform rate. A similar buffer must also be used at the receiver to convert the uniform channel rate to the original variable rate, which is necessary for decoding the received messages.

The use of a buffer to regulate the transmission rate introduces a delay in transmission. This delay is proportional to the size of the buffer, and normally there is a trade off between the size of the buffer and its performance in regulating the transmission rate. For instance, a large buffer introduces a significant delay that may affect real-time transmission of the signal, but effectively regulates the transmission rate. On the other hand, a small buffer introduces an insignificant delay, but can introduce problems associated with buffer overflow. Choosing a suitable buffer size depends on the statistics of the input image and the channel bit-rate. A large buffer must be used if the statistics of the signal change severely. A small buffer size can be used with high bit-rate channels when the signal statistics are relatively constant or when the channel bit-rate is significantly greater than the coder bit-rate [71].

A buffer control mechanism must also be employed to prevent any possible buffer overflow or underflow. Buffer overflow occurs only when an excessive amount of bits is

to be transmitted. Therefore, the system has to keep track with how full the transmitter's buffer is, in order to detect the buffer overflow before it occurs.

An excessive amount of bits is transmitted in the knowledge-based image codec only when a primitive must be transmitted as is. A very straightforward technique to reduce the data rate, in this case, is to sub-sample the primitive and to reduce number of quantization levels for each sample, e.g., to send only 75 or 50 percent of the samples in the primitive and to represent each sample with 5 bits instead of 8 bits. But, since coding a primitive, in this case, is achieved using the principle of transform coding based on the singular-value-decomposition of a prototype primitive, the data rate can be reduced by coarse quantization and transmission of fewer coefficients. This could guarantee no buffer overflow. If buffer overflow is severe and it can not be prevented by coefficient reduction and coarse quantization, the index of the closest primitive from the database is transmitted instead of the coefficients of the transformed primitive.

The above methods are guaranteed to prevent buffer overflow, but they cause image degradation when the source coder is overloaded. The effects of sub-sampling on picture quality are only visible in areas of high detail where large pel-to-pel transitions occur. Also, reducing number of coefficients to be transmitted and using coarser quantizer degrades high frequencies in a primitive. This has a blurring effect on the primitive quality. This dissertation does not contain a detailed study of buffer considerations.

5.6. Parallel/Pipeline Implementation of the Knowledge-Based Teleconference Image Codec

The knowledge-based image code can be implemented in a combined parallel/pipeline structure as shown in Fig. 5.4. Such a structure reduces the processing time of each frame, such that a real time implementation of the coder becomes feasible. The nature of the coding processes is very suitable for this purpose. For instance, the facial features extraction algorithm lends itself to a pipeline implementation, since as discussed earlier, it has five consecutive stages: a pipeline of five stages, each corresponding to a single stage of the algorithm, can be used. The first stage of the pipeline is the edge detection unit, in which the image is operated on by an edge detection algorithm. The second stage is the edge thinning unit, which, itself, is a parallel implementation of the Chen et. al. [74] thinning algorithm. The parallel nature of this algorithm allows all the pixels in the image to be thinned simultaneously, which tremendously speeds up the coding process. The third stage is the contour extraction unit, in which the head's outermost contour is extracted and passed to the fourth stage for ellipse fitting. Since the fourth stage is an iterative stage, it is divided into sub-stages to form a sub-pipeline for high speed iterations. When speed is critical, the number of iterations should be fixed, such that each iteration can be performed in one stage of the sub-pipeline. In our simulations, it has been determined that four iterations are adequate for computing an accurate estimate of the ellipse parameters; hence, the sub-pipeline is composed of four stages. The last stage in the main pipeline is the facial feature location unit, which is used to compute an estimates of the locations of the major facial features. These locations are adjusted as it was explained in chapter 3 .

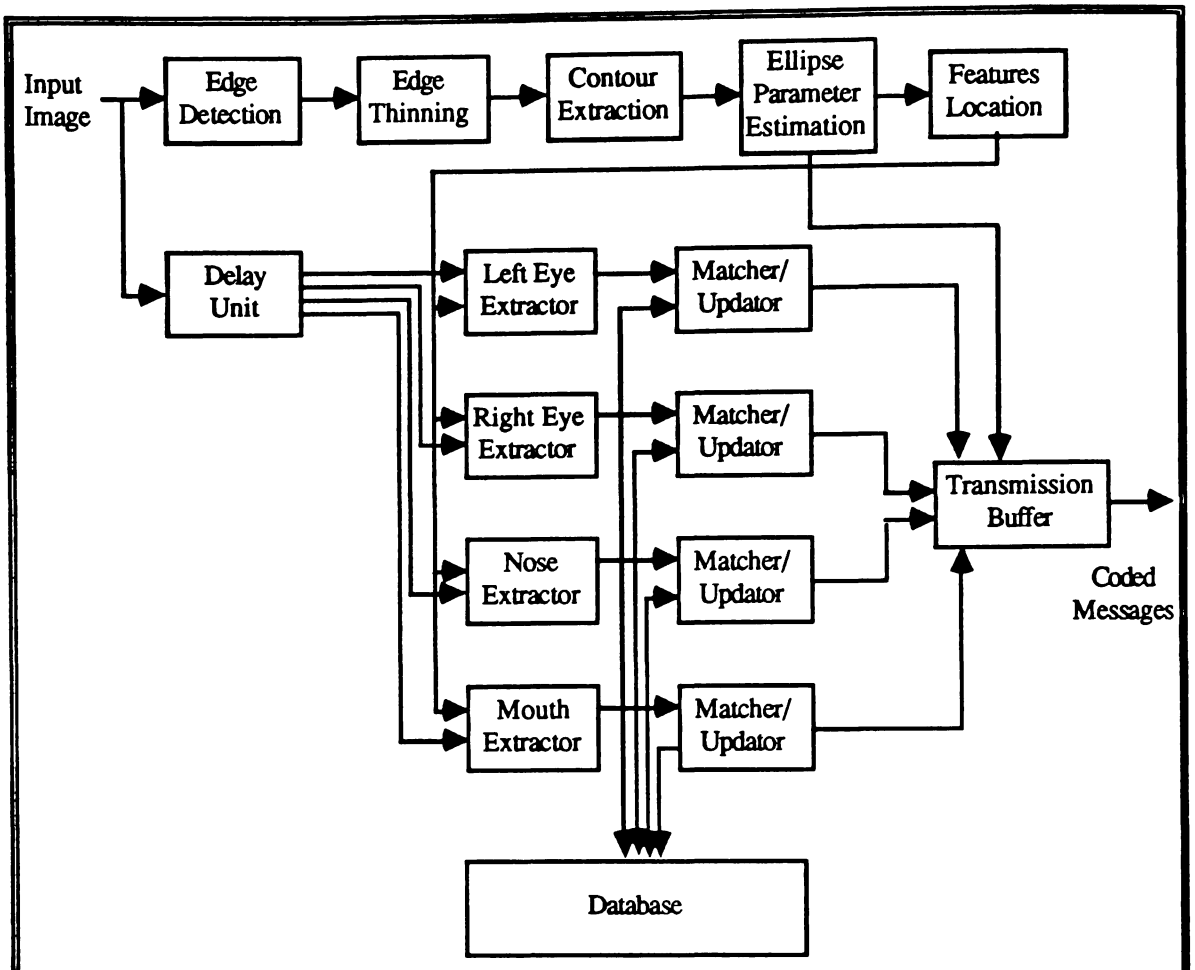


Fig. 5.4 Parallel and Pipeline Implementation of the Knowledge-Based Coder

The coding process starts after the facial features are located, so a delay unit is needed. This unit buffers the input to the coding stage until all the facial features have been located. The coding stage is also implemented in a parallel structure; i.e, all facial features are extracted and matched to the database simultaneously. Therefore, four feature extraction units followed by four feature matching and updating units are used to code the

primitives. The output of the coding stage is passed to a transmission buffer which is a queue that regulates the transmission rate through the channel.

A similar parallel/pipeline structure for the decoder is shown in Fig. 5.5. This structure is simple and has the advantage of reducing the required processing time. First the ellipse parameters decoding unit decodes the ellipse parameters and simultaneously passes them to the features location and previous frame preparation units. In the feature location unit, the locations of all features are calculated. The previous frame preparation unit manipulates the head in the previous frame to form a base image on which the decoded features are to be projected. Later messages are delayed until the locations of all features have been calculated and the previous frame has been prepared. At this time, all primitives are decoded simultaneously, using four parallel units. These units are used to decode the left eye, right eye, nose and mouth. Each of these decoding units is followed by a projection unit to project the corresponding primitive onto its location in the base image. It should be noted that although the decoding and projection steps are each performed in parallel, they are part of an overall pipeline that terminates with the post processing unit, which is used to eliminate the artifacts from the picture. The parallel/pipeline structure of the decoder obviously reduces the required processing time.

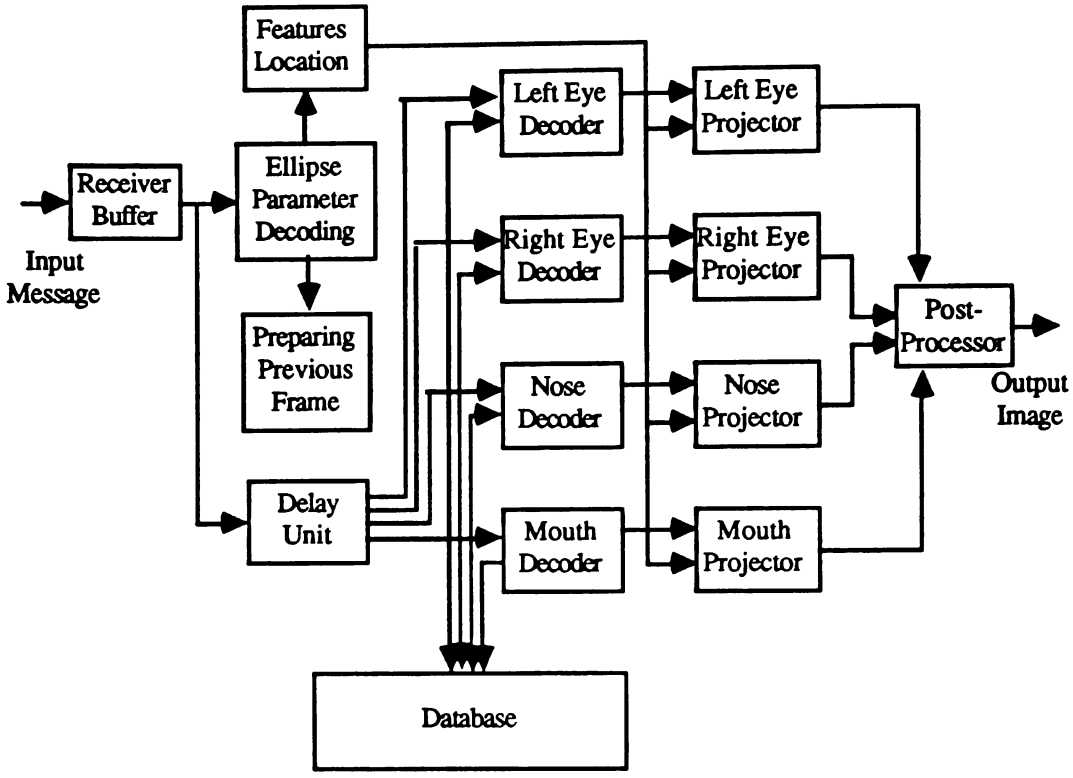


Fig. 5.5 Parallel and Pipeline Implementation of the Knowledge-Based Decoder

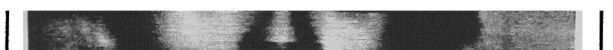
CHAPTER 6

IMPLEMENTATION, SIMULATION, AND PERFORMANCE ANALYSIS OF THE KNOWLEDGE-BASED IMAGE CODEC

6.1. Analysis of the Input Head and Shoulders Image Sequence

Fig. 6.1 Frame 11 in the Head and Shoulders Image Sequence

The input image sequence used in this research was created at North Carolina State University using a *Picture Element Limited Video Sequence Processor* system. Its frame rate is 30 full frames per second. The sequence contains 40 frames of a head and shoulders scene of a speaker talking and moving his head. Each frame is 128x128 pixels, and each pixel is quantized to 8 bits. Hence, each pixel has an intensity value in the range



between 0 and 255. Two frames of this sequence are shown in Figures 6.1 and 6.2. From these figures, it is clear that the images of this sequence are of limited spatial resolution, and that they contain noise.

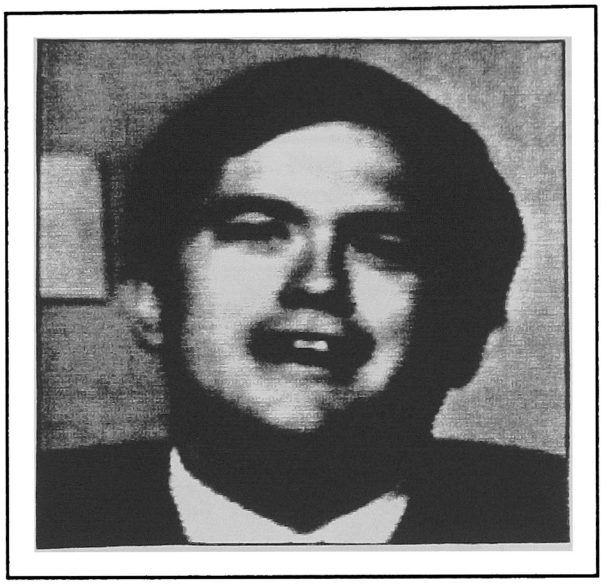


Fig. 6.2 Frame 20 in the Head and Shoulders Image Sequence

6.1.1. Noise Analysis

The noise in the images is due to the recording environment, which includes camera jitter, lighting conditions, and quantization precession. The probability density function of this noise has been estimated by considering areas of the image which have no motion. The upper left hand corners of the images in Figures 6.1 and 6.2 are examples of such areas. Any frame-to-frame change in these areas is considered to be noise. Therefore, a difference frame has been calculated between every pair of consecutive frames in the sequence. The histogram of all the non-zero pixels in a 16x16-

pixel block extracted from the upper left hand corner of each difference frame is shown in Fig. 6.3. This histogram has been normalized, and can thus be considered to be the probability density function of the noise in the image. It is apparent from Fig. 6.3 that the noise is Gaussian, with almost zero mean and a large variance. In fact, the mean and variance of the noise have been estimated to be 0.0483 and 2.3148, respectively.

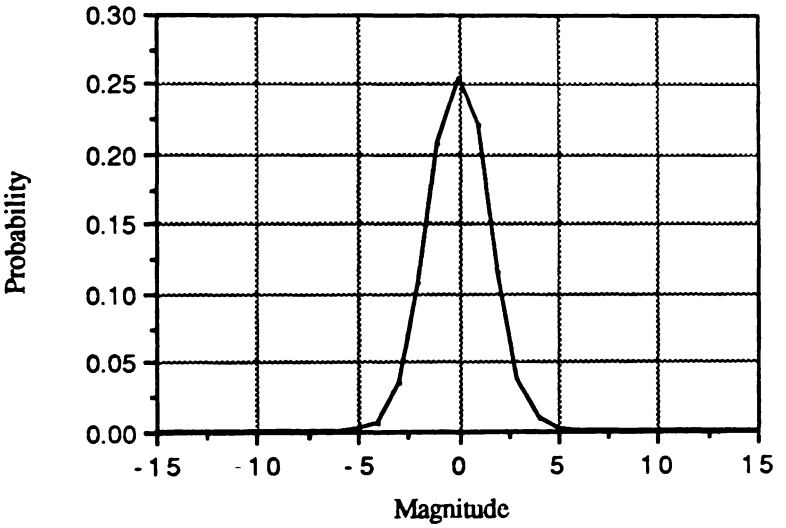


Fig. 6.3 Noise Probability Density Function

6.1.2. Motion Analysis of Full Frames of the Image Sequence

The motion in the image sequence has been calculated by counting the number of pixels which have changed in value by an amount that is greater than that which is accountable for by noise. That is, a pixel value change greater than approximately twice the standard deviation of the noise is considered to be a change due to motion. Therefore,

for each frame in the image sequence, pixels with a change in magnitude of more than 3 were counted and the result plotted in Fig. 6.4. Changes in magnitude of less than 3 were ignored, since they were assumed to be due to noise rather than to motion. Although some of these small changes must be due to motion, they are assumed to be imperceptible to the average viewer, since the human vision system tends to average small changes between frames. This phenomenon is characterized by the temporal modulation transfer function of the human eye, which behaves as a band-pass filter [82].

Fig.6.4 Motion in the Head and Shoulders Image Sequence

It can be noted from Fig. 6.4 that the motion in the first eight frames is very small, but starts to increase at the ninth frame and becomes vigorous in the tenth to the fifteenth frames. At the sixteenth frame, the motion begins to gradually decline and reach a minimum at the thirty second frame, whereupon it rises once again, in a cyclic fashion.

In general, the motion in each frame of the image sequence does not exceed 50%, and the average motion for the entire image sequence is less than 25%. This motion is considered to be higher than that of a typical head and shoulders image sequence, but this is due to the small size (128x128 pixels) of the images used for the study; a more typical image size is 512x512 pixels. As the image size increases, larger stationary areas appear, so a smaller percentage of motion occurs in the overall image.

6.1.3. Motion Analysis of Sub-Areas of the Frames of the Image Sequence

The motion in image sub-areas, such as those containing the nose and one eye, the mouth, or the chin were also studied. First, each frame of the image sequence was divided into 32x32-pixel blocks, and the number of significant changes between the pixels of these blocks and those of the previous corresponding blocks was counted. For the purpose of this study, a significant pixel value change is greater than approximately twice the standard deviation of the noise and is assumed to be due to motion rather than to noise, as described in section 6.1.2. Figure. 6.5 shows the motion in the nose and left eye area, and Fig. 6.6 shows the motion in the nose and right eye area. A comparison of these two figures with Fig. 6.4, shows that there is a high correlation between the motion in the eye/nose areas and the motion in the entire frame. Figure 6.7 represents the motion in the mouth area. Unlike the motion in the eyes/nose areas, Fig. 6.7 does not reflect reflect high correspondence between the motion of the mouth and that of the entire frame. Figure 6.8 shows the motion in the chin area. Note that the number of significantly changed pixels in this area is much smaller than that in the eye/nose and mouth areas.

This change is due only to the motion of the jaws and the rotation of the head, as opposed to the relatively more vigorous motion of the eyes and mouth opening and closing. Therefore, the motion of the chin is very small in comparison to the motion of the eyes and mouth.

It has been calculated from the motion analysis of the full-frames and sub-areas of the frames in the image sequence that more than 70% of the motion in each frame occurs around the head area, while the remaining 30% occurs in the neck and shoulders areas. The distribution of the head area motion over the entire image sequence has been estimated to be 24.7% in the nose/eye areas, 11.7% in the mouth area, 6.0% in the chin area, and 27.6% distributed over the rest of the face. Therefore, careful rotation of the head in a frame and accurate updating of its major primitives from a database would recover most of the motion in the frame and produce the subsequent frame.

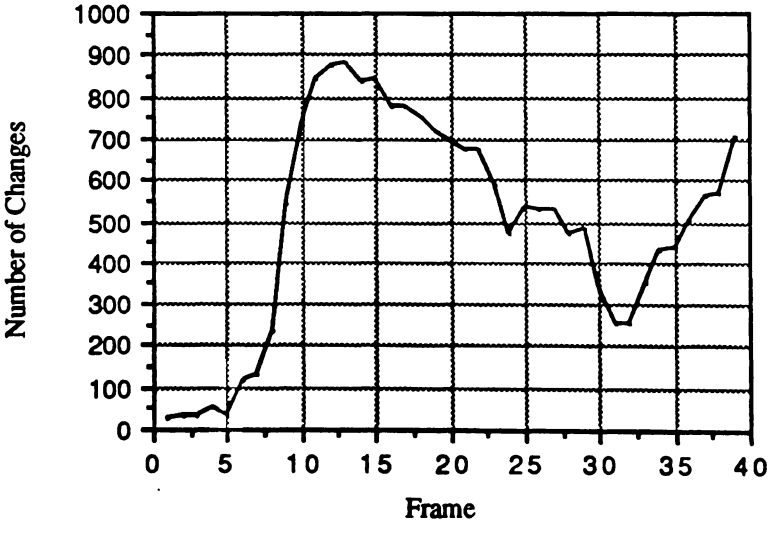


Fig. 6.5 Motion in the Nose and Left Eye Area

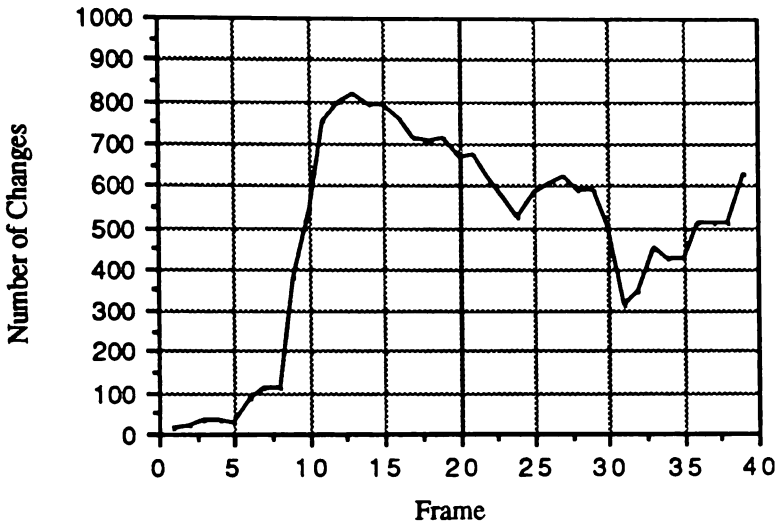


Fig. 6.6 Motion in the Nose and Right Eye Area

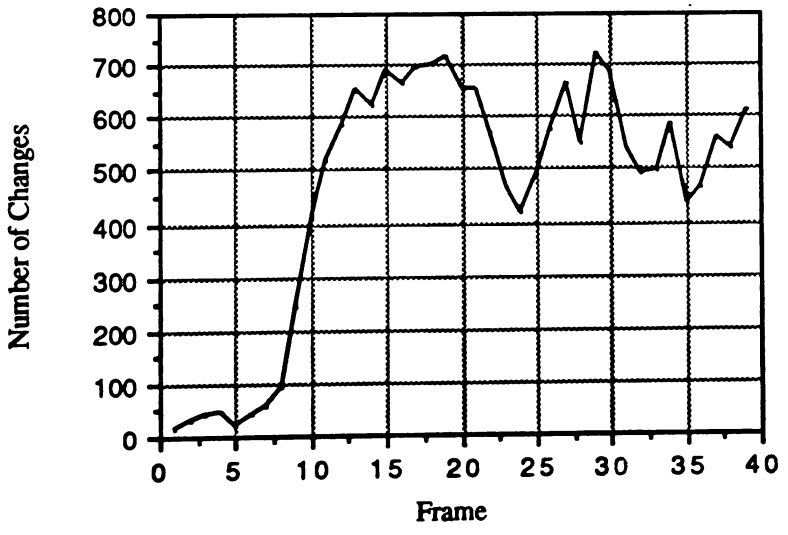


Fig. 6.7 Motion in the Mouth Area

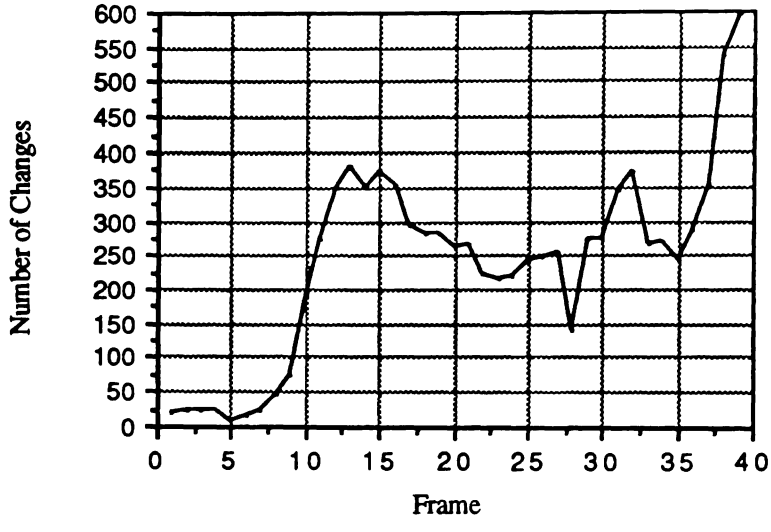


Fig. 6.8 Motion in the Chin Area

6.2. Implementation and Simulation Results of the Knowledge-Based Image Codec

The knowledge-based image codec which is described in chapters 2, 4, and 5 has been successfully implemented in the C language, and a simulation run has been made using the time-varying image sequence of section 6.1. In this simulation, the facial feature extraction algorithm of chapter 3 has been applied to each of the input images to locate its facial features. The application of this algorithm to the image of Fig. 6.1 is described in section 6.2.1. The simulation results for coding and decoding both still images and time-varying image sequences are presented in sections 6.2.2 and 6.2.3, respectively.

6.2.1. Facial Feature Location and Extraction

When the Sobel edge operator [74] is used to enhance the edges in the input image of Fig. 6.1, the edge image of Fig. 6.9 is obtained. The intensity values of the pixels in this edge image are normalized to be between 0 and 255. Figure 6.10 shows the histogram distribution of the pixels in the edge image. This histogram indicates that a global threshold between 16 and 64 can be used to separate the edges from the background of the image. A threshold of 35 has been experimentally selected and applied to the image of Fig. 6.9 to produce the image of Fig. 6.11. From the resultant thresholded image of Fig. 6.11, it is clear that the use of a single global threshold for edge detection produces undesirable edges that are neither thin nor continuous. Therefore, an attempt has been made to use optimal thresholding by defining local thresholds based on properties of the local area around a given pixel. Unfortunately, local thresholding produces many unnecessary, local edges that are of no interest to the knowledge-based technique. A different attempt has been made to improve the quality of the edges by using an average Laplacian edge operator. However, very thick and inaccurate edges are obtained. The inaccuracy is due to the noise in the image, which has a great effect on the locations of the zero crossings of the second derivative. These zero crossings indicate the locations of the edges in the image.

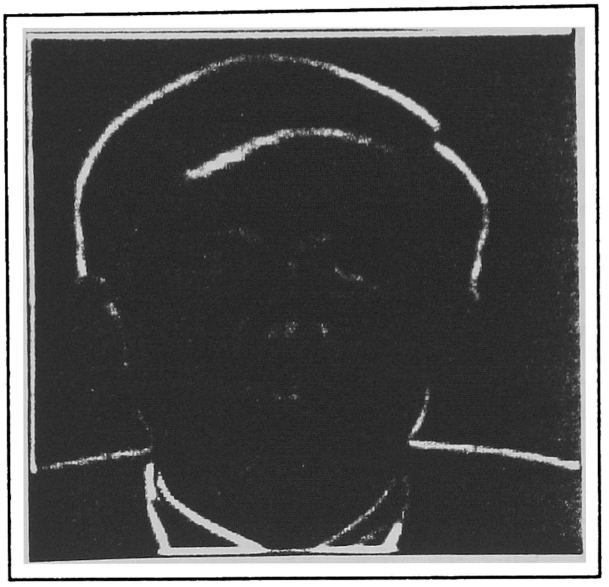


Fig. 6.9 The Edge Image of the Image in Fig. 6.1

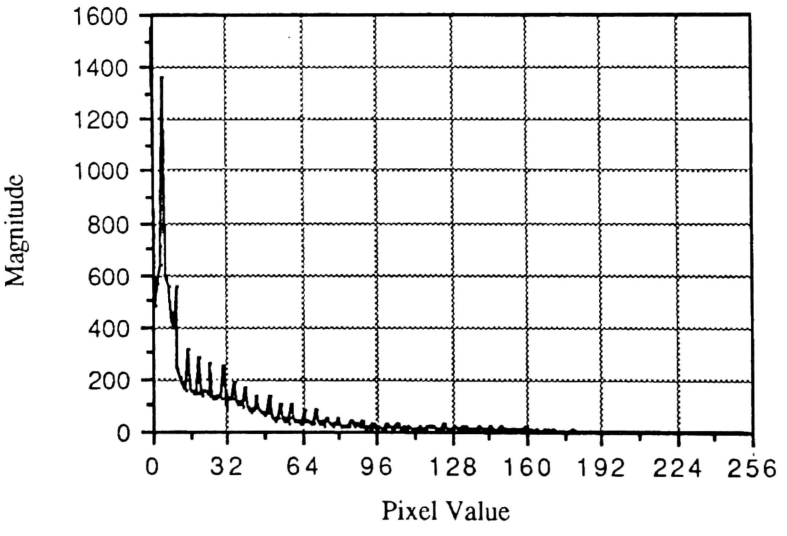


Fig. 6.10 The Histogram of the Edge Image of Fig. 6.9

Fig. 6.11 The Thresholded Edges of the Edge Image of Fig. 6.9

Since the global threshold method which has produced the image of Fig. 6.11 is the best of the three methods tried, the simulation has continued with the image of Fig. 6.11. The edges of the image have been thinned using the Chen et. al thinning algorithm, which can be implemented in a parallel architecture. The thinned image is shown in Fig. 6.12. The outermost contour of the head, as it appears in the image of Fig. 6.12, has been extracted and is shown in Fig. 6.13. For this purpose, it is necessary to separate the head contour from the rest of the image by cutting off the contours of the shoulders and by eliminating the internal facial edges, which may confuse the contour extraction algorithm. The cutting off of the shoulders contours results in cutting off of the chin contour also. However, this does not seriously affect the quality of the subsequent ellipse fitting, since the chin contour is not clearly detected in the original image of Fig. 6.12 anyway. Nevertheless, the chin has been approximated by two different methods in order to compensate for any possible error. In the first method, two straight line segments

Since the global threshold method which has produced the image of Fig. 6.11 is

making a V shape are used to replace the chin, and in the second method, an arc is used to approximate the chin contour. Both methods have been implemented; the arc approximation produces a slightly better-fitting ellipse than the straight line segments.

Fig. 6.12 The Edge Image of Fig. 6.11 After Thinning

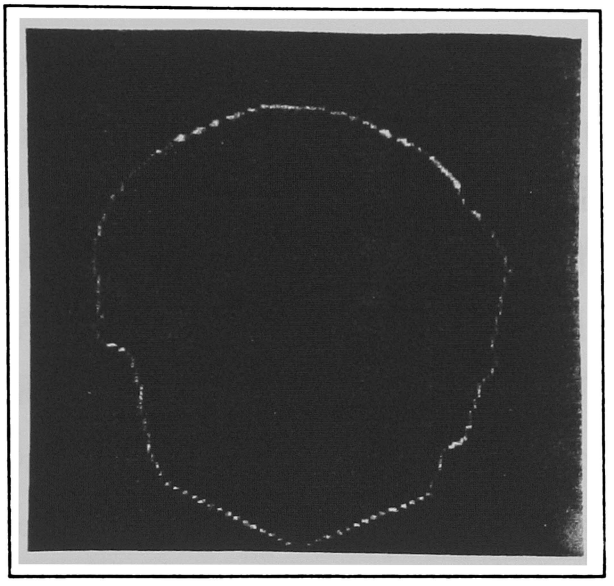


Fig. 6.13 The Outermost Contour of the Head in Fig. 6.1

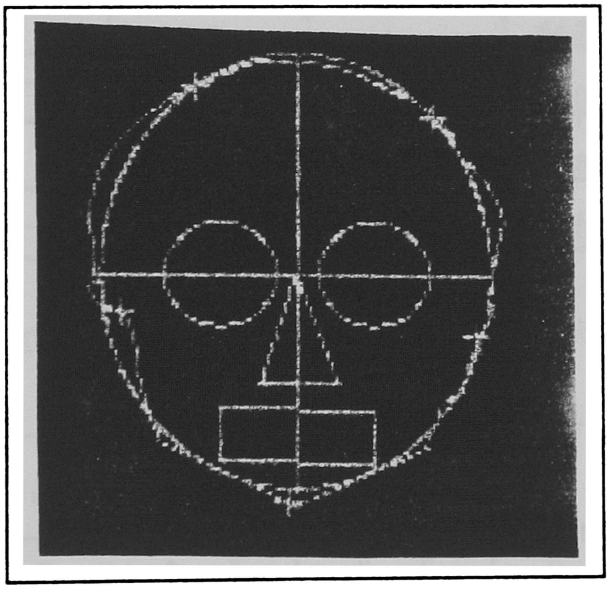


Fig. 6.14 The Best Fit Ellipse for the Head Contour of Fig. 6.13

The iterative curve fitting algorithm which is described in chapter 3 and appendix A has been used to fit an ellipse to the outermost head contour of Fig. 6.13. Although the process is iterative, only four iterations are necessary to produce excellent results. Fig. 6.14 shows the best-fit ellipse superimposed on the head contour for comparison. The goodness of the fit is readily apparent from the figure. The convergence of the algorithm is shown in Fig. 6.15 for fitting two head contours and one synthesized ellipse. From this figure it is clear that the convergence error becomes very small after the first two iterations, which means that the algorithm produces good estimates of the ellipse parameters after only two iterations, and slightly enhances these estimates with the following iterations. The estimated parameters are summarized in Table 6.1.

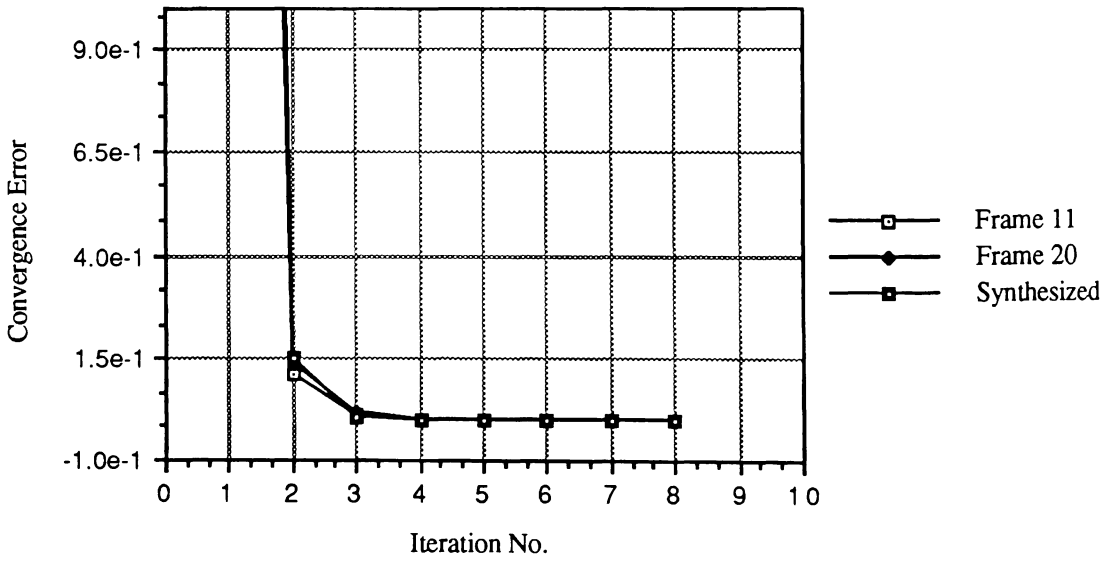


Fig. 6.15 Convergence Error vs. Number of Iterations

Comparable results can be expected from the closed-form curve-fitting algorithm developed by Groshong et. al.[83]. Groshong's algorithm has the advantage of being faster because it requires a smaller number of computations than the iterative method. Table 6.2 summaries the required computations for both the closed-form and the iterative algorithms, assuming N contour points are available.

		Synthesized Ellipse	
Frame 11	**Frame 20**	**Original**	**Estimated**
0.004752	0.128538	0.5	0.501453
58.565	68.579	65	64.77
57.114	56.297	30	29.083
45.611	44.469	20	19.848
53.528	53.576	45	45.012

Table 6.1 The Parameters of the Fitted Ellipses

	Closed Form Curve Fitting	Iterative Curve Fitting
Number of Scalar Additions	36N	30N
Number of Scalar Multiplications	39N	84N
Number of Matrix Additions	Two 3x3	0
Number of Matrix Multiplications	Ten 3x3 and two 3x3 by 3x1	One 5x5 by 5x1
Number of Matrix Inversions	Two 3x3	One 5x5
Number of Eigenvalues & Eigenvectors Determined	Two 3x3	0

Table 6.2 Comparison Between Closed-Form and Iterative Curve Fitting

The locations of the facial features with respect to the best-fit ellipse have been calculated using Equations (3.1)-(3.3). These locations are marked with crosses on Fig. 6.16. Note the shift in the locations of both the mouth and the nose. As explained in chapter 3, this shift is due mainly to the fact that the ratio between the vertical lengths of the upper and lower halves of the human head varies significantly from one person to another. Therefore, it is necessary to adjust these locations, as in Fig. 6.17, by using the vertical signature of the pixels in a slit around the nose and mouth (see chapter 3).

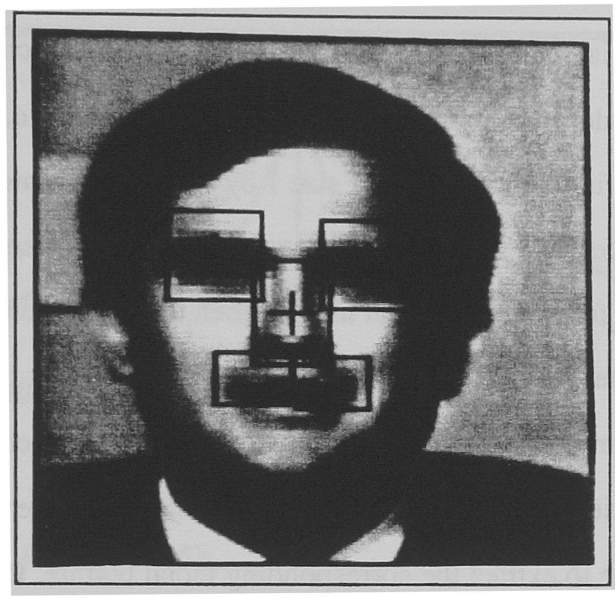


Fig. 6.16 Locations of the Estimated Facial Features

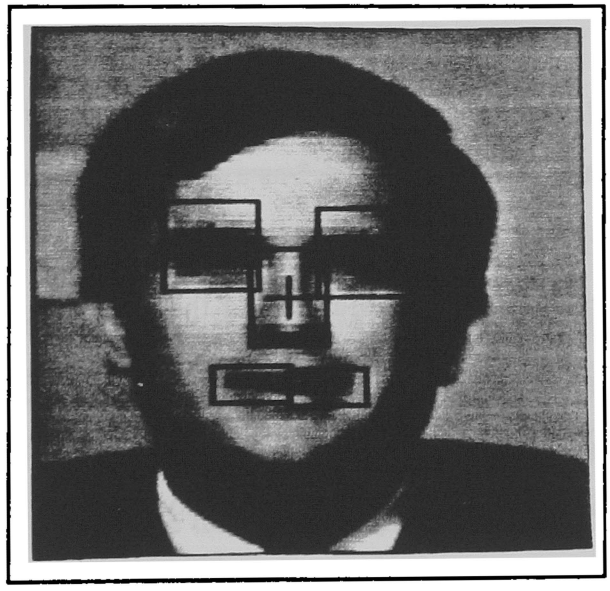


Fig. 6.17 The Adjusted Locations of the Facial Features

For facial feature extraction, the general shape of each filter in the filter set of Fig. 4.1 has been defined and stored in its own separate file. The size of each filter is normalized to fit within a square of size 1 by 1, which is centered at the origin. Before the extraction process, however, the filter's center is shifted to coincide with the center of the corresponding feature in the input image. Then the filter itself is re-scaled and re-oriented according to the size and orientation of the head in the input image. The center, lengths of the major and minor axes, and the orientation of the estimated ellipse are used as shifting, scaling, and orienting factors, as discussed in chapter 4. The filter is then multiplied with the input image to extract the corresponding facial feature. Recall that the internal pixels of the filter are of value 1, and the outer pixels are of value 0, which enables feature extraction by simple multiplication of the filter with the input image.

6.2.2. Coding and Decoding Still Pictures

Consider the image of Fig. 6.2 as a still frame picture. Coding this picture using the knowledge-based technique requires following the techniques of section 6.2.1 to locate and extract the facial features, and then matching the extracted features to a database. The results of the facial feature location and extraction are presented in Figures 3.7 and 3.8 of chapter 3. The extracted primitives must now be matched to a database, and either the indices of the best-match primitives or the primitives themselves must be transmitted.

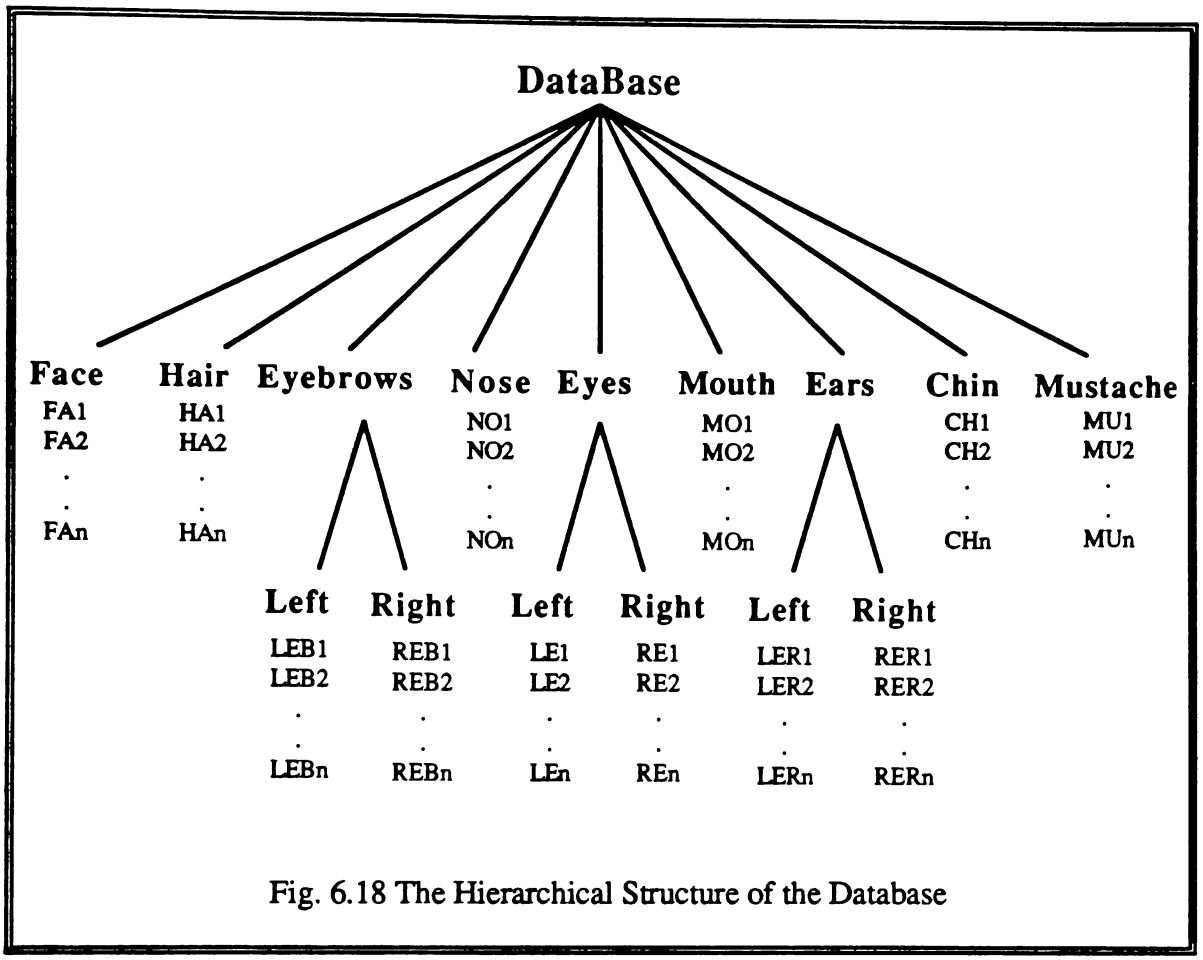


Fig. 6.18 The Hierarchical Structure of the Database

For this purpose, an example database is used. This database has the same characteristics as the database proposed in chapter 4. The primitives are stored in a hierarchical directory structure that emulates the structure of the database. As depicted in Fig. 6.18, each feature type in the database is represented by a directory that contains feature files. The image segments for use in this example database have been manually extracted from the image sequence discussed in section 6.1. Each image segment that represents a feature is stored in a feature file under the directory for its feature type. Each directory group contains only 10 features, so it is not necessary to store these features in the compact form which was discussed in chapter 4.

The simple mean-square-error (*mse*) is used to compare the features extracted from an input image to those of the database. Although the *mse* is not always highly correlated with the subjective image quality, satisfactory results have been obtained in comparing an extracted primitive to the primitives in the example database and selecting the best match. This might not always be true, specially with a real database that contains a large number of primitives that are subjectively different from each other. In such a case, other matching processes, such as those discussed in section 4.5 can be used to achieve the necessary results.

The example database used in this study does not contain a group of prototype images, which are used as base images onto which decoded primitives are projected. Therefore, the image of Fig. 6.19 has been used to construct the prototype image of Fig. 6.20. For this purpose, the head in the image of Fig. 6.19 is first blurred with an 11 by 11 average filter, and then rotated and shifted, so that its center and orientation coincide with those of the input image of Fig. 6.2. The image is blurred such that its details will be lost when the output image is constructed by projecting decoded primitives onto the base image.

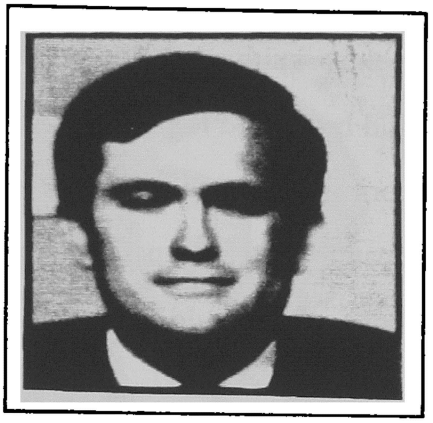


Fig. 6.19 An Image for Constructing a Prototype Image

Fig. 6.20 A Prototype Image Constructed from the Image of Fig. 6.19

In addition to transmitting the indices of the best-match primitives or the primitives themselves, the parameters of the ellipse that best-fits the head contour in the original image must also be transmitted. These are used at the receiver to calculate the location and orientation of each primitive for the output image. Alternatively, the absolute location of each primitive could be transmitted, which is likely to be better than transmitting the ellipse parameters, since a single parameter of the ellipse received in

error, would corrupt all the primitive locations, while an error in one of the transmitted absolute locations, would be limited to only its corresponding primitive. That is, error propagation from one primitive to another would be eliminated. The disadvantage of transmitting absolute locations is that 18 parameters must be transmitted instead of the 5 parameters of the ellipse; this would imply a 44.4% reduction in the achieved compression ratio.

The indices of the selected primitives have been used to retrieve primitives from the database. Size and intensity re-scaling (see chapter 4) of the retrieved primitives are not necessary in this case, because the primitives in the database have all been previously extracted from the same image sequence. However, rotation of each primitive is still necessary, so that each primitive is aligned with the orientation of the head. Each primitive has been projected onto the previously created base image in its proper location, as calculated from the ellipse parameters. Finally, the boundaries of the primitives have been smoothed with a smoothing filter that replaces the value of each pixel on the boundary with the average value of the pixels in a 3 by 3 window around the boundary pixel, and the output image of Fig. 6.21 has been obtained.

Fig. 6.21 Output Image with 1092:1 Compression

Although the *mse* between the image in Fig. 6.21 and the original image of Fig. 6.2 is very high (121.37), the subjective quality of the image is very good. Some background details, have been lost in the blurring process used to create the base image. Such a loss must be expected in the knowledge-based coding technique, since the technique is concerned mainly with the general view of the image rather than its fine details. The blurred transition region between the neck and the head is also due to the averaging process used in preparing the base image. However, this effect is expected to be eliminated with the use of a prototype image as described in chapter 4. The boundaries of the primitives in the image are not completely smoothed because only one boundary pixel was averaged. A better result could be obtained by averaging 5 boundary pixels, as will be shown in coding the time-varying image sequence in section 6.2.3. In spite of the image degradation and the fact that the original primitives of the image have been replaced by similar ones from the database, face recognizability is still preserved and the person in the image can still be perceived as the original person.

background details, have been lost in the blurring process used to create the base image.

It should be noted that only the five parameters of the ellipse and the nine database indices corresponding to the nine facial features extracted from the input image are needed to construct the image of Fig. 6.21. If 16 bits are allocated to the orientation parameter of the ellipse, and 8 bits to each of the four other parameters and the nine indices, then a total of 120 bits are required to transmit the image of Fig. 6.2 and obtain the image of Fig. 6.21. Since each pixel in the input image is represented by 8 bits, and the size of the image is 128 by 128 pixels, a compression ratio of 1092:1 is achievable using this technique. However, the size scaling factor has to be transmitted as well, which requires 8 bits for each primitive. Hence 72 extra bits would be required (for 9 primitives), and the compression ratio drops to 683:1.

With full implementation of the knowledge-based coder, this compression ratio is expected to be even less, because when a best match can not be found in the database, the original primitive itself is transmitted. Consider, for instance, coding a head and shoulders image of size NxM pixels, with each pixel represented by b bits. If a best match of a primitive of size IxJ can not be found in the database, then all the pixels in the original primitive must be transmitted. Therefore bIJ additional bits are required for that primitive. With the use of singular-value-decomposition (see chapter 4), however, this number can be reduced to $0.1bIJ$, assuming 10:1 compression ratio from singular-value-decomposition. Since the size scaling factor for that primitive is no longer required, there would be a saving of 8 bits. If the best match of n primitives, each of size IJ, could not be found in the database, then $(0.1bIJn - 8n)$ extra bits are required. Therefore, the compression ratio would drop to

$$Compression\ Ratio = \frac{bNM}{192+0.1n(bIJ-160)} : 1 \qquad (6.1)$$

With careful design of the database, the average number of primitives not found in the database should be restricted to be between 0 and 1. If the average size of a primitive is 10x10 pixels then the performance of the coder should be around 512:1, which is much higher than ratios achievable by the earlier image coding techniques (maximum of 100:1 by second generation techniques [6]). Note that the achieved compression ratio is highly dependent on the size of the input image. For instance, if the input image is 512x512 pixels, and its average primitive size is 40x40 pixels, then the compression ratio with the knowledge-based image coding technique is 1440:1.

6.2.3. Coding and Decoding Time-Varying Image Sequences

The knowledge-based image coder for time-varying image sequences has been successfully implemented and simulated using the head and shoulders image sequence described in section 6.1. First, the four major primitives of the input image are located as shown in Fig. 6.17. They are then extracted. The extracted primitives are matched to an example database with the hierarchical structure depicted in Fig. 6.18; the mean square error is used for comparing the extracted primitives with those in the database.

The database contains ten eyes, five noses, and six mouths, which have been previously extracted from the image sequence described in section 6.1. These primitives have been selected by examining the image sequence subjectively and assigning a code value between 0 and 5 to each one, based on its state. For example, a tightly closed eye

is assigned code 0, while a widely opened eye is assigned code 5. Then a representative primitive for each code under each type of primitive is selected and included in the database. The assigned codes for the states of all the primitives in the forty frames of the image sequence are summarized in Table 6.3.

Since the example database contains primitives extracted from the input image sequence, a best match is always found in the database for an input image from the image sequence. In this simulation it has never been necessary to transmit the primitive itself, and the indices of the best match have always been transmitted. This will also be the case with a real implementation of the coder, except during the construction of the on-line database, and when switching to a new scene. It is obvious from the codes in Table 6.3 that the construction of the on-line database will not occur in only a few consecutive frames, but will be distributed over a large number of frames (approximately 15). This has the advantage that the performance reduction will be distributed among a larger number of frames, so transmission buffer congestion is avoided.

Frame	Eyes	Nose	Mouth
0	3	1	1
1	3	1	1
2	3	1	1
3	3	1	1
4	3	1	1
5	3	1	1
6	2	1	1
7	1	1	1
8	0	1	1
9	1	2	1
10	2	2	1
11	3	2	2
12	4	3	3
13	4	3	3
14	4	4	4
15	4	4	4
16	4	4	4
17	4	4	5
18	4	5	5
19	4	5	5
20	4	5	5
21	3	5	5
22	3	5	5
23	3	5	5
24	3	5	4
25	3	5	4
26	3	5	3
27	3	5	1
28	3	4	0
29	3	4	1
30	3	4	2
31	3	4	3
32	3	3	4
33	3	3	3
34	3	3	2
35	3	3	2
36	3	2	1
37	3	2	0
38	3	2	0
39	3	2	1

Eyes and Mouth: 0 = closed tightly, 1 = closed, 2 = partially opened, 3 = opened, 4 = very opened, 5 = widely opened

Nose: 0 = down, 1 = normal, 2 = partially up, 3 = high, 4 = very high, 5 = erected

Table 6.3 Status of the Eyes, Nose, and Mouth in Each Frame of the Image Sequence

At the receiver end, the head in the frame previous to the current frame of the input sequence is rotated such that its orientation corresponds to the head orientation in the current frame. For this purpose, the transmitted parameters of the best-fit ellipse are used. Since the rotation is performed on a discrete grid, a few pixels scattered over the head area are missing after the rotation. Hence, these pixels are assigned zero values, which makes them appear as pepper noise in the image. The rotation also causes background areas next to the outermost contour of the head to be revealed. Each pixel in such areas is also assigned a zero value, which causes a thick black border to appear on the side of the head a way from the direction of the head's motion. Both problems are overcome through the use of simple linear interpolation to interpolate the values of the missing pixels from their neighboring pixels. This interpolation completely solves the pepper noise, and replaces the black area around the head with a blurred area that is much less objectionable (see Fig. 6.23).

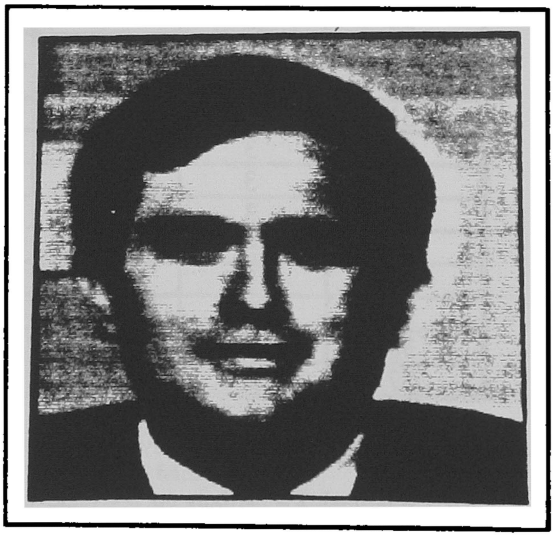


Fig. 6.22 Previous Frame

less objectionable (see Fig. 6.23).

Next, the indices of the selected primitives are used to retrieve duplicate primitives from the database. These primitives are then projected onto their correct locations, which have been calculated using the parameters of the ellipse, as in Equations. 3.1-3.3, on the previous frame, which is used as a base image. The boundaries of the primitives are smoothed using a 3x3 average filter, as with still frames. However, since averaging one pixel along the borders of each primitive in still frames, has not produced excellent results, the boundaries of each primitive are considered 5 pixels wide. Therefore, in addition to each actual boundary pixel, two interior pixels and two exterior pixels are smoothed, and the image of Fig. 6.23 is obtained. Note the difference between the quality of this image and the image in Fig. 6.21.

Fig. 6.23 The Output Image with 1638:1 Compression

Although the mean-square-error between the decoded image of Fig 6.23 and the original image of Fig. 6.1 is very high (95.37), the subjective quality of the image in Fig. 6.23 is very good. It is also obvious that although the original primitives of the image has been replaced with other primitives form the database, face recognizability is still preserved. This observation is expected since face recognizability is influenced more by the external facial features, especially the hair and the head configuration, than by the internal features such as the mouth and the nose. The head configuration is preserved in all the images of the sequence, and the primitives are extracted from the same image sequence (on-line database).

Note that only 80 bits are required to code the image of Fig. 6.23. 16 bits are allocated for the orientation of the ellipse and 8 bits for each of the other parameters. The size scaling factors are not necessary in this case, because the primitives are taken from the on-line database, which contains primitives extracted from the same image sequence as the input, hence, an incredibly high compression ratio of 1638:1 has been achieved for a single frame. However, as in the case of still pictures, this ratio would actually be lower, because when a primitive is not found in the database that primitive itself must be transmitted, with only a 10:1 compression ratio.

It is clear from Table 6.3 that six different primitives, one for each possible state, are required to represent each primitive type in the database. When coding the input image sequence, each of these primitives must be initially transmitted using the principle of singular-value-decomposition, as discussed in section 5.3. If the image size is NxM pixels with b bits/pixel, and the maximum primitive size is IJ, then $0.1bIJ$ bits are required to transmit each of these primitives assuming a $10:1$ compression ratio. If the

image sequence contains K frames, then $\frac{2.4bIJ}{K}$ additional bits are required to transmit each frame in the given sequence. Therefore, the achievable compression ratio is

$$Compression\ Ratio = \lim_{L \to K} \frac{bNM}{80 + \frac{2.4bIJ}{L}} : 1 \qquad (6.2)$$

Note that as K increases, the compression ratio increases, and as K becomes very large the compression ratio approaches $\frac{bNM}{80} : 1$. For instance, if the image size is 128x128 pixels, each pixel is represented by 8 bits, the maximum primitive size is 10x10, and the length of the image sequence is 10 minutes, then the compression ratio would be 1636.2:1, which indicates an insignificant reduction in the compression ratio. This reduction would be significant, however, as the number of frames, K, decrease. For instance, the achievable compression ratio in transmitting a sequence of one second in length would be 910.2:1, which is still extremely high compared to the achievable compression ratio with any of the traditional or second generation image coding techniques.

6.2.4. Investigation of Mis-Projection Error

In order to investigate the viewer tolerance for mis-projecting a primitive onto the base image, intentional errors have been introduced in the locations of some of the projected primitives, and the resultant pictures have been evaluated subjectively. In Fig. 6.24, for instance, the left eye has been shifted 2 pixels down and 3 pixels to the right, and the mouth has been shifted 2 pixels up and 3 pixels to the right. Obviously, these

introduced shifts are not easily detected by the viewer, especially in a viewing time of $\frac{1}{30}$ s, and especially if the boundaries of the primitives are completely smoothed. In Fig. 6.25, the nose has been shifted 4 pixels to the left and 2 pixels up, the mouth has been shifted 3 pixels down and 2 pixels to the right, and an acceptable image is still obtained. However, when the left eye of Fig. 6.25 is shifted 5 pixels to the left and 3 pixels up, the visually objectionable result of Fig. 6.26 is obtained.

Fig. 6.24 Eye Shifted (3,2) and Mouth Shifted (3,-2)

Fig. 6.25 Nose Shifted (-4,-2) and Mouth Shifted (3,3)

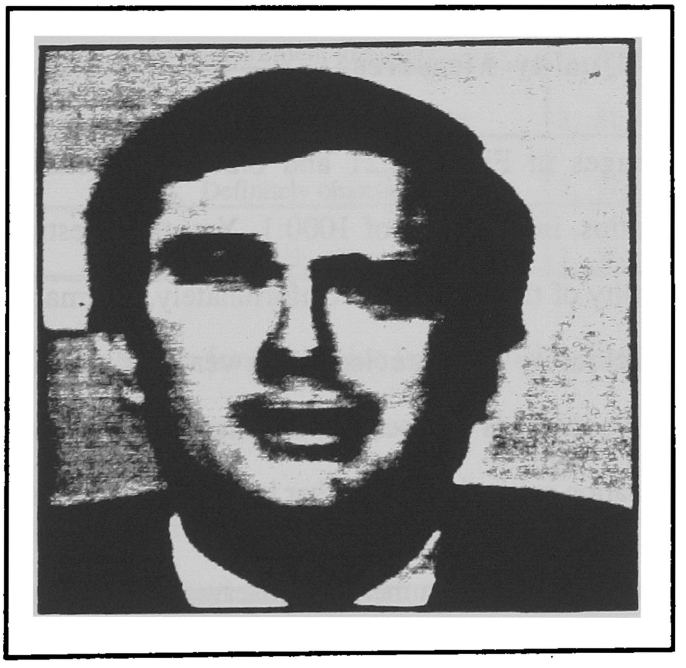


Fig. 6.26 Eye Shifted (-5,-3), Nose Shifted (-4,-2), and Mouth Shifted (3,3)

From these investigations it seems that mis-projection errors of small magnitude can be tolerated, but not of large magnitude. Large magnitude errors might occur due to a transmission error in the parameters of the ellipse, and would tend to produce unusual images with highly objectionable facial structures, such errors would propagate to all primitives, and could cause a primitive such as the eye to be projected onto the mouth area or vice versa. On the other hand, small magnitude errors might occur due to an error in estimating the locations of the primitives. Hence, they would be limited to a single primitive, and would probably affect face expressions rather than face recognizability. Codes for the purpose of error checking and correction could be transmitted along with the ellipse parameters.

6.2.5. Image Quality Measures

The images in Figure 6.21 and 6.23 were compressed at extremely high compression ratios, in the order of 1000:1. Yet, the question to be answered is "how good is the quality of these images". Unfortunately, no image quality measure, that can be used to accurately and precisely answer this question, is readily available. Traditionally, two methods of assessing the image quality have been used. The first is the subjective evaluation method and the other is the quantitative evaluation method [84].

In subjective evaluation method, observers are shown an image and are asked to judge its quality according to some pre-defined rating scale (column one and two in Table 6.4), and the mean opinion score (MOS) for all observers is taken as a measure of the

image quality. The MOS is defined to be simply the average rating of the observers. This method is called the absolute evaluation method. Subjective evaluation can also be done comparatively by asking the observer to compare the image to the average of a set of other images of the same type but with various degrees of degradations. This method is called the comparative evaluation method, and column 3 of Table 6.4 shows a scale that is commonly used for the comparison purpose. Alternatively, subjective evaluation is tedious and time consuming, and it is the final arbiter in determining picture quality.

Overall Goodness Scale	Impairment Scale	Group Goodness Scale
5. Excellent	1. Not perceptible	7. Best in group
4. Good	2. Just perceptible	6. Well above average
3. Fair	3. Definitely perceptible	5. Slightly above average
2. Poor	4. Impairment to picture	4. Average
1. Unsatisfactory	5. Somewhat objectionable	3. Slightly below average
	6. Definitely objectionable	2. Well below average
	7. Extremely objectionable	1. Worst

Table 6.4 Rating Scales for Subjective Image Quality Evaluation

A simpler method to assess the image quality quantitatively involves evaluating a mathematical function based upon some measurements of the reconstructed and the original images. If $F(j,k)$ denotes the original image and $\hat{F}(j,k)$ denotes the constructed

image, then the quantitative image quality measure may be expressed in the generalized form as

$$Q = \sum_{j=1}^{J} \sum_{k=1}^{K} (O\{F(j,k), \hat{F}(j,k)\}) \tag{6.3}$$

where $O\{.\}$ is some operator, such as the square of the difference between both images, the cross-correlation function, or a more complicated function that incorporates the properties of the human vision system. Usually, the Normalized Mean-Square Error (*NMSE*) is used for its simplicity and mathematical tractability.

$$NMSE = \frac{\sum_{j=1}^{J} \sum_{k=1}^{K} [G(j,k) - \hat{G}(j,k)]^2}{\sum_{j=1}^{J} \sum_{k=1}^{K} [G(j,k)]^2} \tag{6.4}$$

where $G(j,k)$ is the transformed image field as explained below. Alternatively, the square of the peak value of the image can be used as a normalization factor. The mean-square error measure, in this case, is called the Peak Mean-Square Error (*PMSE*) and is defined as

$$PMSE = \frac{\frac{1}{JK} \sum_{j=1}^{J} \sum_{k=1}^{K} [G(j,k) - \hat{G}(j,k)]^2}{A^2} \tag{6.5}$$

Where A is the peak of the image. The image $F(j,k)$ is transformed to $G(j,k)$ such that certain properties of the image are emphasized. For instance, the power law

$$G(j,k) = [\, F(j,k)]^{\nu} \tag{6.6}$$

may be used to compensate for the effect of the image display device. The logarithmic law can also be used to compensate for the effect of the early stages of the human vision system. Hence

$$G(j,k) = k_1 \ log_b [k_2 + k_3 \, F(j,k)] \tag{6.7}$$

where k_1, k_2, and k_3 are constants. The Laplacian operator can also be used to emphasize the quality of the edges in the image. In this case

$$G(j,k) = F(j+1,k) + F(j-1,k) + F(j,k+1) + F(j,k-1) - 4F(j,k) \tag{6.8}$$

A major drawback of all the above quantitative image quality measures is that the simulation results do not always reflect correlation with the subjective quality of the image [84]. Therefore, better image quality measures incorporates various characteristics of the human vision system. A recent model [85] uses a logarithmic type of nonlinearity based

on the global threshold effects. Then the error between the transformed original and reconstructed images is filtered by a spatio-temporal filter that emphasizes the mid-frequency range, both in the temporal and spatial domain. This error signal is then normalized with a measure that incorporates the masking action of the human visual system. This measure reflects local changes in the signal in a small neighborhood close to the picture element being processed. Finally, the resulting normalized error signal is averaged over an area, and taken as a measure of the image quality.

A more recent approach to the evaluation of the image quality is the use of the Picture Quality Scale (*PQS*) [86]. The *PQS* is computed using a vector of three error measurements. The first measurement is the random errors, the second is the autocorrelation errors and the blocking artifacts, and the third is the local errors along the contours. These three measurements are first normalized then transformed by the principal component analysis. The *PQS* is a linear combination of the principal components. The test results of *PQS* measure have indicated high correspondence with the Mean Opinion Score (*MOS*), but more simulation result is required to establish the usability of the *MOS* measure.

None of the image quality measures described in this section were used in this research.

6.3. Computation Requirements

An estimation of the computation requirements of the various stages of the knowledge-based image coder is summarized in Table 6.5. For this estimation, it is

assumed that the size of the image is NxN pixels, and the head contour has a maximum of M pixels. It is also assumed that the size of each primitive is IxJ pixels, and that each primitive group in the database contains L primitives. The total number of computations varies, based on number of primitives in the image. For instance, while there are four primitives in coding time-varying head and shoulders image sequences, there are eight primitives in coding still pictures of the same type. Therefore, the number of computations per primitive is given in the table for the feature matching, feature projection, and post-processing stages. In the feature projection stage, a one-pixel linear interpolation between the pixels of the scaled primitive is assumed (see chapter 4). The number of computations required for rotating the head in the previous frame has not been included in the number of computations required for the feature projection step. In fact, head rotation requires a maximum of $4NxN$ multiplications, $2NxN$ additions, and $2NxN$ calls for the standard mathematical functions such as sine, cosine and square root of 2.

	Edge Detect.	Edge Thin.	Curve Fit.	Feature Locat.	Feature Match.	Feature Projec.	Post Proce.
Number of Scalar Add/Sub	$17N^2$		30M	70	2K*I*J	2*I*J	80(I+J)
Number of Scalar Multip/Divis	$20N^2$		84M	130	K*I*J	4*I*J	2(I+J)
Logical Oper.		$94N^2$					
Std. Math Function Calls	$1N^2$			22		2*I*J	
Number of Matrix Multiplications			One 5x5 by 5x1				
Matrix Inversion			One 5x5				

Table 6.5 Computation Requirements for the Knowledge-Based Image Codec

Although branching and decision making have an effect on the speed of the algorithm, they are not included in the table because it is extremely difficult to assess their effect on the speed of the algorithm.

The knowledge-based image coding algorithm has been implemented in the C language on a VAX 785 computer. Execution of the algorithm requires 3 to 5 minutes, depending on the computer load at the time of the execution. No attempt has been made to estimate the required CPU time for coding one image, because the VAX is a time-shared machine.

The number of additions and multiplications, that would be necessary to code and decode a 256x256 head and shoulders image sequence with primitives of sizes 10x10, and a maximum of 1024 head contour points (4 times the image size) can be approximated from Table 6.5. If we assume every 50 logical operators are equivalent to one adder, and each standard mathematical function is implemented with approximately 3 additions and 33 multiplications, then the coder performs in the order of 45 million multiplications and 110 million additions per second. On the other hand, the decoder performs in the order of 22 thousand multiplications and 75 thousand additions per second.

It should be noted that the parallel /pipeline implementation of the knowledge-based image coder, which was discussed in section 5.4, have the same number of computations as summarized in Table 6.5, but since edge detection and curve fitting are computed through a pipeline, and since all the pixels in the image are thinned simultaneously, the time required to locate the features in the image would be approximately one fourth of the time required for a normal implementation. It is also obvious that the time required to extract, match, project, and post-process all the features of the image would be equivalent to the time required to extract, match, project, and post-process a single feature in the normal implementation. Therefore, the parallel/pipe line implementation would be at least five times faster than the normal implementation.

CHAPTER 7

CONCLUSIONS AND SUGGESTIONS

7.1. Conclusions

In this study the general structure of a new knowledge-based image coding technique is developed. This technique is capable of achieving an extremely high compression ratio, while producing images of very good quality by utilizing a priori knowledge about the class of input images. This technique allows various type of a priori knowledge to be incorporated in the coding process in order to reduce the amount of transmitted information. Hence, the more a priori knowledge is included in the system the higher the system's achievable compression ratio is.

The developed knowledge-based technique is then applied to the class of head and shoulders images of the type commonly found in mug-shot files and "face-to-face" telecommunications. The a priori knowledge in these two applications is explicitly included in the system in the form of image primitives stored in a database. Also, other a priori knowledge is implicitly used to define a model for the head and shoulders images that allows fast and accurate extraction of the image primitives. These extracted primitives form the messages to be transmitted.

The Knowledge-based system for coding head and shoulders images is successfully implemented, and compression ratios in the order of 1000:1 are achieved with both still frames and time-varying image sequences. The quality of the decoded

images is very good at an extremely high compression ratio. However, the decoded image quality in still pictures case depends on the quality of the database, so a highly representative database of the images in an application is required. This requirement, however, is not essential for coding time-varying image sequence, since an on-line database can be constructed using primitives extracted from the images of the input sequence. The new technique enables real-time transmission of video signals through local telephone networks at 64 Kbits/s. Also it enables coding mug-shot files for the purpose of automatic archiving system for forensic applications.

Also, an algorithm is developed in order to locate the major facial features in the input head and shoulders image. The algorithm is based on a model developed expressly for head and shoulders images. The algorithm has been successfully implemented. Results show that the major facial features can be accurately located in a photograph, in a simple and fast manner.

7.2. Suggestions for Further Study

Although the developed knowledge-based image coding techniques achieve extremely high compression ratios while producing images with very good quality, this technique is still in the infancy stages and further improvements are necessary. Therefore, the following topics are suggested for further research:

(1) Develop a method for automatically generating and updating the database. For this purpose, issues such as size, rate of growth, and management of the database need to be investigated.

(2) Extend the algorithm to include other classes of images. This requires an accurate definition of the primitives of the images in the given class, a fast and accurate primitive extraction algorithm, and the construction of a representative database of reasonable size.

(3) Improve the quality of the produced image by coding an error image, that is constructed from the difference between the original image and the constructed image.

(4) Incorporate more knowledge in the system in order to achieve higher compression ratio; e.g., knowledge about the motion model of the primitives.

(5) Store in the database three-dimensional models for the primitives, that allows three- dimensional manipulation of the primitives. Hence, a primitive can be manipulated through model-based animation, in order to obtain an accurate representation of the original primitive in the image.

(6) Extend the facial feature extraction algorithm to include profile and semi-profile views of the head. This can be done by utilizing the other rotation angles of the head (see chapter 3), which are highly related to the elongation and symmetry of the facial view.

(7) Devise an expert system that is able to produce the best image quality under any given application constraints. Therefore, it is necessary to evaluate existing image coding techniques to determine the type of images and bandwidth constraints for which an image coding technique performs best, and include the results of such an evaluation in the knowledge base of the system.

CHAPTER 8

LIST OF REFERENCES

[1] A. K. Jain, "Image Data Compression: A Review," *Proc. of IEEE*, vol. 69, No. 3, pp. 349-389, March 1981.

[2] A. N. Netravali and J. O. Limb, "Picture Coding: A Review," *Proc. of IEEE*, vol. 68, No. 3, pp. 366-406, March 1980.

[3] H. G. Musmann, P. Pirsch, and H. Grallert, "Advances in Picture Coding," *Proc. of IEEE*, vol. 73, No. 4, pp. 523-548, April 1985.

[4] Y. Yasuda, Y. Yamazaki, T. Kamae, and K. Kobayashi, "Advances in Fax," *Proc. of IEEE*, vol. 73, No. 4, pp. 707-730, April 1985.

[5] M. Kunt, A. Ikonomopoulos, and M. Kocher, "Second-Generation Image Coding," *Proc. of IEEE*, vol. 73, No. 4, pp. 549-574, April 1985.

[6] S. A. Rajala, M. R. Civanlar, and W. M. Lee, "A Second Generation Image Coding Techniques Using Human Visual System Based Segmentation," *Proc. of the ICASSP'87*, pp. 1362-1365, April 1987.

[7] A. N. Netravali and B. G Haskell, *Digital Pictures Representation and Compression*, Plenum Press, New York 1988.

[8] B. M. Oliver, J. R. Pierce, and C. E. Shannon, "The Philosophy of PCM," *Proc. of IRE*, vol. 36, pp. 1324-1331, October 1948.

[9] J. Max, "Quantization for Minimum Distortion," *IRE Trans. Infor.Theory*, vol. IT-6, pp. 7-12, March 1965.

[10] R. M Gray, "Vector Quantization," *IEEE ASSP Magazine*, pp. 4-29, April 1984.

[11] Y. Linde, A. Buzo, and R. M Gray, "An Algorithm for Vector Quantizer Design," *IEEE Trans. Commun.*, vol. COM-28, pp. 84-89, January 1980.

[12] W. H. Equitz, "Fast Algorithm for Vector Quantization Picture Coding," *M.S. Thesis*, MIT, June 1984.

[13] R. J. Clarke, *Transform Coding of Images*, Academic Press, London 1985.

[14] P. A. Wintz, "Transform Picture Coding," *Proc. of the IEEE*, vol. 60, July 1972.

[15] A. K. Jain, "A Fast Karhunen-Loève Transform For A Class of Random Processes", *IEEE Trans. Commun.*, vol. COM-24, pp. 1023-1029, 1976.

[16] H. C. Andrews and W. K. Pratt, "Fourier Transform Coding of Images," *Hawaii Int. Conf. on System Science*, pp. 677-679, January 1968.

[17] J. Makhoul, "A Fast Cosine Transform in One and Two Dimensions," *IEEE Trans. Acoustics, Speech and Signal Proc.*, vol. ASSP-28, No. 1, pp. 27-34, February 1980.

[18] W. K. Pratt, J. Kane and H. C. Andrews, "Hadmard Transform Image Coding," *Proc. IEEE*, vol. 57, No. 1, pp. 58-68, January 1969.

[19] J. W. Modestino, N. Farvardin and M. A. Ogrinc, "Performance of Block Cosine Image Coding with Adaptive Quantization," *IEEE Trans. Commun*, vol. COM-33, No. 3, pp. 210-217, March 1985.

[20] H. C. Andrews and C. L. Petterson, "Singular Value Decomposition (SVD) Image Coding," *IEEE Trans. Commun.*, vol. COM-24, 4, pp. 425-432. April 1976.

[21] J. W. Woods and S. D. O'Neil, "Subband Coding of Images," *IEEE Trans. Acoustics, Speech and Signal Proc.*, vol. ASSP-34, pp. 1278-1288, October 1986.

[22] J. B. O'Neal, Jr., "Delta Modulation Quantization Noise Analytical and Computer Simulation Results for Gaussian and TV Input Signals," *Bell Syst. Tech. J.*, vol. 45, pp. 117-142, 1966.

[23] J. B. O'Neal, Jr., "Predictive Quantization System (Differential Pulse Code Modulation) for the Transmission of Television Signals," *Bell Syst. Tech. J.*, vol. 45, pp. 689-721, May-June 1966.

[24] W. Zschunke, "DPCM Picture Coding With Adaptive Prediction," *IEEE Trans. Commun.*, vol. COM-25, No. 11, pp. 1295-1302, November 1977.

[25] A. N. Netravali and C. B. Runimstein, "Quantization of Picture Signals Using Spatial Masking," *Proc. of the IEEE*, vol. 65, pp. 536-548, April 1977.

[26] D. K. Sharma and A. N. Netravali, "Design of Quantizers for DPCM Coding of Picture Signals," *IEEE Trans. Commun.*, vol. COM-25, pp. 1267-1274, November 1977.

[27] F. W. Mounts, "A Video Encoding System With Conditional Picture Element Replenishment," *Bell Sys. Tech. J.*, vol. 48, No. 7, pp. 2545-2554, September 1969.

[28] J. A. Roese, W. K. Pratt, and G. S. Robinson, "Interframe Cosine Transform Image Coding," *IEEE Trans. Commun.*, vol COM-25, No. 11, pp. 1329-1339, November 1977.

[29] W. A. Pearlman and P. Jakatdar, "The effectiveness and Efficiency of Transform/DPCM Interframe Image Coding," *IEEE Trans. Commun.*, vol. COM-32, No. 7, pp. 832-838, July 1984.

[30] L. D. Davisson, "Data Compression Using Straight Line Interpolation," *IEEE Trans. Infor. Theory*, vol. IT-14, pp. 390-394, May 1968.

[31] R. B. Arps, "Binary Image Compression," *in Advances in Electronics and Electron Physics*, Suppl. 12, Image Transmission (W.K. Pratt, Ed.), Academic Press, New York, 1979.

[32] G. G. Langdon, Jr. and J. Rissanen, "Compression of Block-White Images with Arithmetic Coding," *IEEE Trans. Commun.*, vol. COM-29, No. 6, pp. 858-867, June 1981.

[33] J. C. Candy and R. H. Bosworth, "Methods for Designing Designing Differential Quantizers Based on Subjective Evaluations of Edge Busyness," *Bell Syst. Tech. J.*, vol. 51, pp. 1495-1516, Sept. 1972.

[34] A. N. Netravali, "On Quantizers for DPCM Coding of Picture Signal," *IEEE Trans. Inform. Theory*, vol. IT-23, no. 3, pp. 360-370, May 1977.

[35] J. D. Eggerton and M. D. Srinath, "A Visually Weighted Quantization Scheme for Image Bandwidth Compression at Low Data Rates," *IEEE Trans. Commun.*, vol. COM-34, No. 8, pp. 840-847, August 1986.

[36] W. F. Schreiber, C. F. Knapp, and N. D. Kay, "Synthetic Highs, An Experimental TV Bandwidth Reduction System," *J. SMPTE*, vol. 68, pp. 525-537, August 1959.

[37] J. K. Yan and D. J. Sakrison, "Encoding of Images Based on A Two-Component Source Model," *IEEE Trans. Commun.*, vol. COM-25, No. 11, pp. 1315-1322, November 1977.

[38] A. Ikonomopoulos, M. Kunt, " High Compression Image Coding Via Directional Filtering," *Signal Processing*, vol. 8, pp. 179-203, 1985.

[39] P. J. Burt and E. H. Adelson, "The Laplacian Pyramid As A Compact Image Code," *IEEE Trans. Commun.*, vol. COM-31, pp. 532-540, Apr. 1983.

[40] R. Wilson, H. E. Knutsson, and G. H. Granlund, "Anisotropic Nonstationary Image Estimation and Its Application: Part II-Predictive Image Coding," *IEEE Trans. Commun.*, vol. COM-31, pp. 398-406, March 1983.

[41] G. Zorpette, "Fractals: Not Just Another Pretty Picture," *IEEE Spectrum*, pp. 29-31, October 1988.

[42] M. F. Barnsley and A. D. Sloan, "A Better Way to Compress Images," *Byte Magazine*, pp. 215-223, January 1988.

[43] F. I. Parke, "A Model for Human Faces That Allows Speech Synchronized Animation," *J. Computers and Graphics*, vol. 1, No. 1, pp. 1-4, March 1975.

[44] F. I. Parke, "Parameterized Models for Facial Animation," *IEEE Computer Graphics and Applications*, vol. 12, pp. 61-67, November 1982.

[45] K. Aizawa, H. Harashima, and T. Saito, "Model-Based Synthetic Image Coding System- Construction of A 3-D Model of A Person's Face," *in Proc. Picture Coding Symposium (PCS-87)*, Royal Institute of Technology, Stockholm, pp. 50-51, 1987

[46] W. Geuen, "Principle Strategy of Model Based Source Coding," *in Proc. Picture Coding Symposium (PCS-87)*, Royal Institute of Technology, Stockholm, pp. 165-166, 1987.

[47] W. J. Welsh, "Model-Based Coding of Moving Images at Very Low Bit-Rates," *in Proc. Picture Coding Symposium (PCS-87)*, Royal Institute of Technology, Stockholm, pp. 47, 1987.

[48] R. Forchheimer, "The Motion Estimation Problem in Semantic Image Coding," *in Proc. Picture Coding Symposium (PCS-87)*, Royal Institute of Technology, Stockholm, pp. 171-172, 1987.

[49] R. Forchheimer and O. Fahlander, "Low Bit-Rate Coding Through Animation, *Picture Coding Symposium (PCS-83)*, Davis, pp. 113-114, 1983.

[50] R. J. Brachman and H. J. Levesque, editors, *Readings in Knowledge Representation*, M. Kaufmann Publishers, Los Altos 1985.

[51] N. Cercone and G. McCalla, editors, *The Knowledge Frontier: Essays in the Representation of Knowledge*, Springer-Verlag, New York 1987.

[52] J. Walters and N. R. Nielsen, *Crafting Knowledge-Based Systems: Expert Systems Made Easy Realistic*, Wiley, New York 1988.

[53] K. Parsaye and M. Chignell, *Expert Systems for Experts*, Wiley, New York 1988.

[54] P. H. Winston, *Artificial Intelligence*, second edition, Chapter 7, pp. 205-249, Addison-Wesley, 1984.

[55] P. H. Winston, *Artificial Intelligence*, second edition, Chapter 4, pp. 87-131, Addison-Wesley, 1984.

[56] N. Magnenat-Thalmann and D. Thalmann, *Computer Animation: Theory and Practice*, Springer-Verlag, Tokyo 1985.

[57] N. Burtnyk and M. Wein, "Computer-Generated Key-Frame Animation,"*J. of Society of Motion Picture and Television Engineers*, vol. 80, pp. 149-153, 1971.

[58] N. Burtnyk and M. Wein, "Interactive Skeleton Techniques for Enhancing Motion Dynamics in Key Frame Animation," *Comm. ACM*, vol. 19, pp. 564-569, October 1976.

[59] W. T. Reeves, "Inbetweening for Computer Animation Utilizing Moving Point Constraints," *Proc. SIGGRAPH'1983*, Computer Graphics, pp. 359-376, 1983.

[60] A. M. Barr, "Superquadrics and Angle-Preserving Transformations," *IEEE Computer Graphics and Applications*, vol. 1, pp. 310-321, January 1981.

[61] B. A. Barsky, *Computer Graphics and Geometric Modeling Using Beta-Splines*, Springer-Verlag, Tokyo 1985.

[62] M. L. Gillenson and B. Chandrasekaran, "A Heuristic Strategy for Developing Human Facial Images on a CRT," *Pattern Recognition*, vol. 7, pp. 187-196, 1975.

[63] G. M. Davies, "Face Recall Systems", in *perceiving and Remembering Faces*, Edited by G. Davies, H. Ellis, and J. Shepherd, pp. 226-250, Academic Press, London 1981.

[64] A. J. Goldstein, L. D. Harmon, and A. B. Lesk, "Identification of Human Faces," *Proceedings of the IEEE*, vol. 59, No. 5, pp. 748-760, May 1971.

[65] L. D. Harmon and W. F. Hunt, "Automatic Recognition of Human Face Profiles," *Computer Graphics and Image Processing*, vol. 6, pp. 135-136, 1977.

[66] L. D. Harmon, S. C. Kuo, P. F. Famig, and U. Raudkivi, "Identification of Human Face Profiles by Computer," *Pattern Recognition*, vol. 10, pp. 301-312, 1978.

[67] L. D. Harmon, M. K. Khan, R. Lasch, and P. F. Ramig, "Machine Identification of Human Faces," *Pattern Recognition*, vol. 13, No. 2, pp. 97-110, 1981.

[68] T. Sakai, M. Nagao and T. Kanade, "Computer Analysis and Classification of Photographs of Human Faces," *Proceedings of the First USA-Japan Computer Conference,* pp. 55-62, 1972.

[69] L. K. Bromley, "Computer-aided Processing Techniques for Usage In Real-time Image Evaluation," *Master's Thesis*, University of Houston, 1977.

[70] T. P. Wiederhold, "Facial Image Generation By Computer," *Master's Thesis*, University of Houston, 1976.

[71] B. Hogarth, *Drawing the Human Head*, 1st ed., Watson-Guptill, New York 1965.

[72] S. Simmons III and M. S. Winer, *The Creative Process*, Prentice-Hall, New Jersey, 1977.

[73] A. N. Netravali and B. G Haskell, *Digital Pictures Representation and Compression*, Plenum Press, pp. 537-542, New York 1988.

[74] D. H. Ballard and C. M. Brown, *Computer Vision*, Prentice-Hall, New Jersey, 1982.

[75] R. T. Chen, H. K. Wan, D. L. Stover, and R. D. Iverson, "A One-Pass Thinning Algorithm and Its Parallel Implementation," *Computer Vision, Graphics, and Image Processing*, vol. 40, pp. 30-40, October 1987.

[76] E. L. Hall, *Computer Image Processing and Recognition*, Academic Press, New York, 1979.

[77] H. D. Ellis, M. A. Jeeves, F. Newcombe, and A. Young, *Aspects of Face Processing*, Martinus Nijhoff 1986.

[78] J. Shepherd, G. M. Davies, and H. Ellis "Studies of Cue Saliency", in *perceiving and Remembering Faces*, Edited by G. Davies, H. Ellis, and J. Shepherd, pp. 105-131, Academic Press, London 1981.

[79] T. J. Stonham, "Practical Face Recognition and Verification With Wisard," in *Aspects of Face Processing*, Edited by H. D. Ellis, M. A. Jeeves, F. Newcombe, and A. Young, pp. 426-441, Martinus Nijhoff 1986.

[80] R. S. Malpass and K. D. Hughes, "Formation of Facial Prototypes," in *Aspects of Face Processing*, Edited by H. D. Ellis, M. A. Jeeves, F. Newcombe, and A. Young, pp. 154-162, Martinus Nijhoff 1986.

[81] R. J. Clarke, "On the Relation Between the Karhunen-Loève and Cosine Transforms," *IEE Proc. Part F: Communication Radar Signal Processing*, vol. 128, pp. 359-360, 1981.

[82] E. H. Adelson and J.R. Bergen, "Spatio-temporal Energy Models for the Perception of Motion," *J. of the Optical Society of America*, vol. 2, No. 2, pp. 284-299, February 1985.

[83] B. R. Groshong, G. L. Bilbro and W. E. Snyder, "Fitting a Quadric Surface to Three Dimensional Data," *Center for Communications and Signal Processing*, Technical Report, CCSP-TR-85/17.

[84] W. K. Pratt, "Sampled Image Quality Measures," *Digital Image Processing*, Chapter 7, pp. 162-198, John Wiley and Sons, New York, 1978

[85] J. O. Limb, "Distortion Criteria of the Human Viewer," *IEEE Trans. on Systems, Man Cybernetics*, December 1979.

[86] M. Miyahara, "Quality Assessments for Visual Service," *IEEE Communication Magazine*, pp. 51-60, October 1988.

CHAPTER 9

APPENDICES

9.1. Appendix A

The following is the solution for the optimization of the parameters of the ellipse that best-fits the head contour.

Let a, b, c, d, and e denote the true parameters of the ellipse of best fit, and let a', b', c', d', and e' denote an approximate values of these parameters. If (x_i , y_i) denotes a point that lies on the contour of the head, and (X_i , Y_i) denotes the nearest point on the ellipse, then

$$f(X_i , Y_i ; a , b , c , d , e) = 0 \qquad (A.1)$$

Where $X_i = x_i - \Delta x_i$, $Y_i = y_i - \Delta y_i$, $a = a' - \Delta a$, $b = b' - \Delta b$, $c = c' - \Delta c$, $d = d' - \Delta d$, $e = e' - \Delta e$. Assuming Δx_i, Δy_i, Δa, Δb, Δc, Δd, and Δe are very small, Eq. (A.1) can be expanded around the point $(x_i , y_i , a' , b' , c' , d' , e')$ by a Taylor series expansion. When the higher order terms are ignored, the expansion is given by

$$f(x_i\ ,\ y_i\ ,\ a',\ b',\ c',\ d',\ e') - \frac{\partial f_i}{\partial x}\Delta x_i - \frac{\partial f_i}{\partial y}\Delta y_i - \frac{\partial f_i}{\partial a}\Delta a$$

$$- \frac{\partial f_i}{\partial b}\Delta b - \frac{\partial f_i}{\partial c}\Delta c - \frac{\partial f_i}{\partial d}\Delta d - \frac{\partial f_i}{\partial e}\Delta e = 0 \qquad (A.2)$$

for all $(i=1,2,...,n)$, where $\frac{\partial f_i}{\partial x}\Delta x_i$ represents the value of $\frac{\partial f}{\partial x}\Delta x$ at $(x_i\ ,\ y_i\ ,\ a',\ b',\ c',\ d',\ e')$.

This problem is a constrained optimization problem in which the performance measure S (see Eq. (6)) is to be minimized under the constraint of Eq. (A.2). The Lagrange multiplier method can be used to solve this problem. If we denote h_i by the left hand side of Eq. (A.2), the problem is reduced to minimizing the following augmented function

$$g = \frac{1}{2}S - \sum_{i=1}^{n} p_i\ h_i \qquad (A.3)$$

where p_i is the Lagrange multiplier and S is the performance measure. Differentiating this equation with respect to Δx_i , Δy_i , Δa, Δb, Δc, Δd, and Δe respectively, and setting the result equal to 0, produces the following equations

$$\sum_{i=1}^{n} p_i\ \frac{\partial f_i}{\partial a} = 0 \qquad (A.4a)$$

$$\sum_{i=1}^{n} p_i \frac{\partial f_i}{\partial b} = 0 \qquad (A.4b)$$

$$\sum_{i=1}^{n} p_i \frac{\partial f_i}{\partial c} = 0 \qquad (A.4c)$$

$$\sum_{i=1}^{n} p_i \frac{\partial f_i}{\partial d} = 0 \qquad (A.4d)$$

$$\sum_{i=1}^{n} p_i \frac{\partial f_i}{\partial e} = 0 \qquad (A.4e)$$

$$\Delta x_i + p_i \frac{\partial f_i}{\partial x} = 0 \qquad for\ (i=1,2,...,n) \qquad (A.5a)$$

$$\Delta y_i + p_i \frac{\partial f_i}{\partial y} = 0 \qquad for\ (i=1,2,...,n) \qquad (A.5b)$$

Substituting Eq. (A.5) back into Eq. (A.2) and solving for p_i yields,

$$p_i = \frac{1}{m_i}(f_{ai}\,\Delta a + f_{bi}\,\Delta b + f_{ci}\,\Delta c + f_{di}\,\Delta d + f_{ei}\,\Delta e - f_i) \qquad (A.6)$$

where

$$m_i = (\frac{\partial f_i}{\partial x})^2 + (\frac{\partial f_i}{\partial y})^2$$

$$= (2a'x_i + 2b'y_i + d')^2 + (2b'x_i + 2c'y_i + e')^2$$

$$f_{ai} = \frac{\partial f_i}{\partial a}$$

$$= x_i^{\,2}$$

$$f_{bi} = \frac{\partial f_i}{\partial b}$$

$$= 2x_i y_i$$

$$f_{ci} = \frac{\partial f_i}{\partial c}$$

$$= y_i^{\,2}$$

$$f_{di} = \frac{\partial f_i}{\partial d}$$

$$= x_i$$

$$f_{ei} = \frac{\partial f_i}{\partial e}$$

$$= y_i$$

$$f_i = f(x_i , y_i , a', b', c', d', e')$$

Substituting Eq. (A.6) into Eq. (A.4) produces the following set of linear equations

$$\Delta a \sum_{i=1}^{n} \frac{f^2_{ai}}{m_i} + \Delta b \sum_{i=1}^{n} \frac{f_{ai} f_{bi}}{m_i} + \Delta c \sum_{i=1}^{n} \frac{f_{ai} f_{ci}}{m_i} + \Delta d \sum_{i=1}^{n} \frac{f_{ai} f_{di}}{m_i} +$$

$$\Delta e \sum_{i=1}^{n} \frac{f_{ai} f_{ei}}{m_i} - \sum_{i=1}^{n} \frac{f_{ai} f_i}{m_i} = 0 \qquad (A.7a)$$

$$\Delta a \sum_{i=1}^{n} \frac{f_{bi} f_{ai}}{m_i} + \Delta b \sum_{i=1}^{n} \frac{f^2_{bi}}{m_i} + \Delta c \sum_{i=1}^{n} \frac{f_{bi} f_{ci}}{m_i} + \Delta d \sum_{i=1}^{n} \frac{f_{bi} f_{di}}{m_i} +$$

$$\Delta e \sum_{i=1}^{n} \frac{f_{bi} f_{ei}}{m_i} - \sum_{i=1}^{n} \frac{f_{bi} f_i}{m_i} = 0 \qquad (A.7b)$$

$$\Delta a \sum_{i=1}^{n} \frac{f_{ci} f_{ai}}{m_i} + \Delta b \sum_{i=1}^{n} \frac{f_{ci} f_{bi}}{m_i} + \Delta c \sum_{i=1}^{n} \frac{f^2_{ci}}{m_i} + \Delta d \sum_{i=1}^{n} \frac{f_{ci} f_{di}}{m_i} +$$

$$\Delta e \sum_{i=1}^{n} \frac{f_{ci} f_{ei}}{m_i} - \sum_{i=1}^{n} \frac{f_{ci} f_i}{m_i} = 0 \qquad (A.7c)$$

$$\Delta a \sum_{i=1}^{n} \frac{f_{di} f_{ai}}{m_i} + \Delta b \sum_{i=1}^{n} \frac{f_{di} f_{bi}}{m_i} + \Delta c \sum_{i=1}^{n} \frac{f_{di} f_{ci}}{m_i} + \Delta d \sum_{i=1}^{n} \frac{f^2_{di}}{m_i} +$$

$$\Delta e \sum_{i=1}^{n} \frac{f_{di} f_{ei}}{m_i} - \sum_{i=1}^{n} \frac{f_{di} f_i}{m_i} = 0 \qquad (A.7d)$$

$$\Delta a \sum_{i=1}^{n} \frac{f_{ei} f_{ai}}{m_i} + \Delta b \sum_{i=1}^{n} \frac{f_{ei} f_{bi}}{m_i} + \Delta c \sum_{i=1}^{n} \frac{f_{ei} f_{ci}}{m_i} + \Delta d \sum_{i=1}^{n} \frac{f_{ei} f_{di}}{m_i} +$$

$$\Delta e \sum_{i=1}^{n} \frac{f^2_{ei}}{m_i} - \sum_{i=1}^{n} \frac{f_{ei} f_i}{m_i} = 0 \qquad (A.7e)$$

Arranging these equation in matrix form produces

$$\begin{bmatrix} \sum_{i=1}^{n} \frac{f^2_{ai}}{m_i} & \sum_{i=1}^{n} \frac{f_{ai} f_{bi}}{m_i} & \sum_{i=1}^{n} \frac{f_{ai} f_{ci}}{m_i} & \sum_{i=1}^{n} \frac{f_{ai} f_{di}}{m_i} & \sum_{i=1}^{n} \frac{f_{ai} f_{ei}}{m_i} \\ \sum_{i=1}^{n} \frac{f_{bi} f_{ai}}{m_i} & \sum_{i=1}^{n} \frac{f^2_{bi}}{m_i} & \sum_{i=1}^{n} \frac{f_{bi} f_{ci}}{m_i} & \sum_{i=1}^{n} \frac{f_{bi} f_{di}}{m_i} & \sum_{i=1}^{n} \frac{f_{bi} f_{ei}}{m_i} \\ \sum_{i=1}^{n} \frac{f_{ci} f_{ai}}{m_i} & \sum_{i=1}^{n} \frac{f_{ci} f_{bi}}{m_i} & \sum_{i=1}^{n} \frac{f^2_{ci}}{m_i} & \sum_{i=1}^{n} \frac{f_{ci} f_{di}}{m_i} & \sum_{i=1}^{n} \frac{f_{ci} f_{ei}}{m_i} \end{bmatrix}$$

$$\begin{bmatrix} \sum_{i=1}^{n} \frac{f_{di} f_{ai}}{m_i} & \sum_{i=1}^{n} \frac{f_{di} f_{bi}}{m_i} & \sum_{i=1}^{n} \frac{f_{di} f_{ci}}{m_i} & \sum_{i=1}^{n} \frac{f^2_{di}}{m_i} & \sum_{i=1}^{n} \frac{f_{di} f_{ei}}{m_i} \\ \sum_{i=1}^{n} \frac{f_{ei} f_{ai}}{m_i} & \sum_{i=1}^{n} \frac{f_{ei} f_{bi}}{m_i} & \sum_{i=1}^{n} \frac{f_{ei} f_{ci}}{m_i} & \sum_{i=1}^{n} \frac{f_{ei} f_{di}}{m_i} & \sum_{i=1}^{n} \frac{f^2_{ei}}{m_i} \end{bmatrix}$$

$$\begin{bmatrix} \Delta a \\ \Delta b \\ \Delta c \\ \Delta d \\ \Delta e \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} \frac{f_{ai} f_i}{m_i} \\ \sum_{i=1}^{n} \frac{f_{ai} f_i}{m_i} \\ \sum_{i=1}^{n} \frac{f_{ai} f_i}{m_i} \\ \sum_{i=1}^{n} \frac{f_{ai} f_i}{m_i} \\ \sum_{i=1}^{n} \frac{f_{ai} f_i}{m_i} \end{bmatrix} \qquad (A.8)$$

The above equation can be easily solved for Δa, Δb, Δc, Δd, and Δe.

Using equation (A.8), an iterative method may be used to solve for the parameters of the ellipse. The algorithm starts by calculating the deltas using Eq. (A.8), and an initial guess of the parameters of the ellipse. Then, the guess is updated by the deltas to obtain better parameter estimates and the deltas are recalculated using Eq. (A.8) and the new parameters. The process is repeated until the deviation is small enough to indicate that a', b', c', d', and e' are acceptable estimates of the true parameters of the ellipse.