# ISSUES IN DEVELOPING A VERY LOW BIT RATE VIDEOPHONE CODER

A THESIS
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL
AND ELECTRONICS ENGINEERING
AND THE INSTITUTE OF ENGINEERING AND SCIENCES
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

By
Roy Mikael Mickos
December 1993

# ISSUES IN DEVELOPING A VERY LOW BIT RATE VIDEOPHONE CODER

A THESIS
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING
AND THE INSTITUTE OF ENGINEERING AND SCIENCES
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

By
Roy Mikael Mickos
December 1993

Roy mikael Mickos
tarafından bağışlanmıştır

TK
5102
.M53
1993

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

[illegible]

Assoc. Prof. Dr. Levent Onural (Principal Advisor)

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

[illegible]

Assoc. Prof. Dr Enis Çetin

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

[illegible]

Assist. Prof. Dr Orhan Arıkan

Approved for the Institute of Engineering and Sciences:

[illegible]

Prof. Dr. Mehmet Baray
Director of Institute of Engineering and Sciences

## Abstract

The issues of a suitable transmission image size, general behaviour, and buffer control of a very low bitrate videophone video signal coder to be used in future mobile and public switched telephone networks are addressed. A software simulator of the coder was built so that the performance of the coder and the various alternative methods under consideration could be tested by subjective evaluation. In the case of transmission image size a clear choice between the two alternatives, QCIF and NCIF, is achieved: QCIF. The behaviour of the coder is explained on the basis of some statistical parameters extracted from it. With head-and-shoulders sequences without buffer regulation the coder is succesful in allocating bits to those regions in the image containing the most important information. Finally, the buffer control scheme of the coder is analyzed and an alternative method, based on framewise analysis of the bits created for that frame, is developed which is shown to be better than the original.

*Keywords :* Very Low Bitrate Video Coding, Videophone, Data Compression, Video Coding

## ÖZET:

Dar frekans bantlı telefon ağlarını kullanarak goruntülü telefon iletişimini gerçekleştirme hedefi, çok düşük veri hızıyla sayısal kodlama yapmayı zorunlu kılmıştır. Böyle bir kodlayıcının video sinyali kodlayan bölümü; genel özellikleri, uygun goruntü boyutları ve tampon kontrolu ele alınmıştır. Kodlayıcı için önerilen alternatif yöntemlerin perforamansını görüntülü telefon kullanıcısının gözüyle değerlendirebilmek için, kodlayıcının yazılımla gerçekleştirilmiştir. Görüntü boyutları için, 176x144 (QCIF standardı) boyutlarının diğer seçenekten daha iyi olduğu sonucuna varılmıştır. Kodlayıcının genel özellikleri, bazı istatistik parametrelerine dayanarak açıklanmıştır. Portre görüntü dizileri için, kodlayıcının tampon düzenlemesi yokken dahi 'bit'leri en önemli bilgileri içeren bölgeler için kullanmakta başarılı olduğu gözlenmiştir. Kodlayıcının tampon kontrol yöntemi incelenmiş, ve alternatif bir yöntem önerilmiştir. Önerilen yöntemin, öncekinden daha iyi olduğu gösterilmiştir.

*Anahtar Sözcükler :* Çok Düşük Veri Hızında Video Kodlama, Goruntülü Telefon, Veri Sıkıştırma, Video Kodlama

## Acknowledgements

Facing a new culture both in- and outside the university world when working towards this thesis has been many times an exhausting experience but always rewarding. It is because of the support and friendship of the following people that this thesis has been made possible; but I also want to thank them for many things *not* written in this thesis.

Assoc. Prof. Levent Onural has been the supervisor of this thesis. I am thankful for the inspirating aspects and insights towards scientific work I have got to now in the course of this work, and also for the severity and thoroughness in guiding my work.

Also I would like to thank the folks in the digital image processing group: M. Sc. Gözde Bozdağı, M. Sc. Bilge Alp, M. Sc. Levent Öktem and M. Sc. Şennur Ulukuş for their encouragement and help in countless problems, many of which had nothing to do with this thesis. But I still think that a person can do without a car

Prof. Yrjö Neuvo is acknowledged for getting me into Turkey in the first place.

Thanks to B. Sc. Okan Yılmaz, B. Sc. Ersin Unal and B. Sc Inanç Yıldırım for their company in the search for the true meaning of life, for example in issues ranging from Turkish cuisine and operational areas to bazaar behaviour.

Thanks to all the people in the department of Electrical and Electronics Engineering at the Bilkent University for such an excellent working atmosphere.

*Only a whim makes life worth living, because in following a whim one takes ones destiny into one's own hands and reconciles oneself to the faith*

Mika Waltari, freely translated by the undersigned.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 General

The field of digital video signal processing is currently undergoing a phase where technology is being transferred from the research laboratories into commercial products. This process has started in the television with the development of High Definition Television (HDTV) standards, the first of which is currently being considered for approval in the United States. Compared to analog technology, digital technologies offer far more flexible ways of manipulating the video data allowing for techniques like redundancy reduction to take place. Digital technology also allows for some previously unseen products like a visual extension to the every-day telephone: the video telephone. In fact, there already exists a standard for cable-connected video telephone services standardized by the CCITT (Comité Consulative Internationale de Téléphone et Télégraphe, also known as ITU-TS which comes from the English translation of the french name: International Telegraph and Telephone Consultative Committee (ITU-TS)) which is known as H.261. This standard is intended for communication through the ISDN (Integrated Services Digital Network) networks. It is also among the first standardization efforts in the field, settled in 1991. This thesis deals with extending this standard to portable services, i.e. video telephone services operating via radio waves like the already existing portable telephones or through the Public Switched Telephone Networks

| Service | Bit rate | Typical Resolution | Notes |
|---|---|---|---|
| HDTV, cable and satellite | ~32 Mbit/s | $1260 \times 1152$ | |
| HDTV, terrestrial | ~15 Mbits/s | $1260 \times 1152$ | |
| Video Telephone, H.261 | $p \times 64$ kbits/s | $352 \times 288$ | H.261, CIF |
| Video Telephone, COST | $p \times 8$ kbits/s | $176 \times 144$ | QCIF |

Table 1.1: Characteristics of some digital video services under consideration. CIF and QCIF are acronyms for respective resolutions, COST is the group promoting the respective rate. Values given for HDTV (High Definition TeleVision) bit rates and resolutions are approximate as no standard has been set and because proposed systems have many different resolution formats.

(PSTN).

To put the discussion into more solid basis, table 1.1 gives the data rates in bits for various digital video services. These rates are determined, of course, by the bandwidth requirements of the media used for transmission. It can be seen that the difference between the extremes are enormous: HDTV systems are intended for giving very high quality images comparable to film quality (the resolution is so good in fact that it is practically impossible to display these signals in full resolution using the current cathode ray tube technology) while the video telephones offer "recognizable" quality. Still, one of the key issues in the standardization work currently under way is the interoperability of these services. Someone owning a portable video telephone may wish to watch terrestrial television broadcasts with his/her device. This is referred as *scalability* or *compatibility* of the transmission. This can be accomplished through *hierarchical* transmission where the signal is decomposed into several resolution and/or quality levels.

Referring again to table 1.1 it is common to refer the rates at $p \times 64$ kbits/s as *low bit rate coding* and rates at $p \times 8$ kbits/s as *very low bit rate coding* (in MPEG4 this is defined as rates between 4.8 – 64 kbits/s).

The fundamental difference between HDTV and video telephone is that the video telephone allows for two-way communication. Thus video telephones are likely to become the predecessors of future multimedia terminals. As an example, already such services like facsimile and interactive digital data transmission through a modem can be carried through the existing telephone lines in addition to their normal use.

Mainly the computer industry is currently developing products called personal assistants which combine a laptop computer with a portable telephone, which is to combine the abovementioned uses of the telephone network. But, in the foreseeable future these terminal devices will also be able to handle the compatible tv-broadcasts already mentioned, multimedia electronic mail, remote sensing, electronic newspapers, interactive multimedia databases, multimedia videotex etc. ([11], [8][1]), latter of which directly utilizes the possibility of two-way communication offered by a telephone network.

We need a flexible digital standard for the format of the bit stream that allows the terminal device to utilize all the possibilities that a two-way digital communication can offer.

Despite the seemingly vast difference in performance requirements all video coding systems currently under consideration for a standard (see the listing below to which can be added the so called Grand Alliance proposal for a HDTV standard for the USA) have some common ground: except for quantization, these are built around linear time-invariant methods of signal processing, and linear transforms (mainly the discrete cosine transform, shorted as DCT) for compression. Specifically, linear time-invariant methods are suitable for frequency-domain formulations as their eigenfuctions are sine waves. However, image data (as opposed to audio data) cannot be satisfactorily modeled as comprising of superimposed sine waves unless the frequency space is allowed to extend to infinity. Thus, it would seem that nonlinear methods which have had success especially within the field of image processing would have a fundamental advantage in this aspect. However, these techniques are emerging and their immaturity makes them currently unsuitable for commercial applications of this scale. Nonlinear methods are one of the active research areas within signal processing.

In the following, a list of existing and emerging standards related to the video

[1]This is a general reference to all MPEG-related issues dealt in this chapter

signal processing and portable services are listed.

- *JPEG* is a standard for still video or plain image compression and coding. Offers adjustable rate/distortion coding, and is used in photography and image storing. JPEG players plus image data are available for computers. Standardized by ISO (International Standardization Organization).

- *H.261* is a standard for transmitting video signals over the integrated serviced digital network (ISDN). Developed within a joint European group called COST211bis and standardized by the CCITT in 1991. Products supporting this standard are on sale. Bit rates supported are $p \times 64$ kbits/s.

- *MPEG1* is the first of a series of standards under development by ISO and IEC (International Electro-technical Commission) jointly. This standard specifies formats for the storage of video signals for multimedia applications. It has a video resolution comparable to today's VCRs and audio capability matching that of CD's. Data rates used are up to 1.5 Mbits/s. The standard comprises of four parts (systems, video, audio, and implementation) first three of which reached a draft stage at the end of 1992. Software simulators supporting MPEG1 are available.

- *MPEG2* is an extension of the MPEG1, and it aims to be a generic (application independent) standard for coding moving video. It supports interlaced formats as well as progressive and multi-resolution bit stream allowing the interoperability issues discussed earlier. It is developed for data rates above 3 Mbits/s. The most notable application area will be HDTV. The standard has already reached an advanced stage (as of summer -93 they are optimizing their basic coder). The Grand Alliance proposal [9] currently under consideration for a HDTV standard for USA is claimed to be MPEG2-compatible.

- *MPEG4*[2]. This is a standardization project, again under ISO and IEC, that will start in autumn -93. The purpose is to create standard for very-low bit rate coding (both storage and communication), possibly using a novel

[2]MPEG3 has become obsolete and it is absorbed to MPEG2

method (other than waveform based) for video coding. This effort will address all possible uses of a digital network including those already mentioned in discussing benefits of two-way communication. It will also consider electronic surveillance, games and deaf sign language captioning. It aims to be operational with ISDN and LANs (Local Area Networks). Data rates involved will probably be 4.8 – 64 kbits/s.

- *COST211ter.* This is a joint european working group aiming to produce a proposal for a CCITT standard for portable audiovisual services by September 1993. The standard to be issued is going to be a H.261 -based. Later this standard will probably be merged into the MPEG4 standard which is supposed to be wider (it will also be one of the aims of the upcoming MPEG4). This group works exclusively on video signal coding, other services are not on the agenda. The CCITT standard will issue video coding only at rates $p \times 8$ kbits/s.

- *GSM.* A standard for digital transmission of audio signals. Already established, networks are operational and expanding, products are on sale. Operates at $p \times 8$ kbits/s, which directly carries over to the video telephone world (it is expected that future video telephones will use GSM equipment for transmission). The standard is international but implemented mainly in northern and continental Europe. Recently an US mobile phone operator set up a consortium to promote a GSM-like system for North America. Features data encryption for privacy and signal compression. Users carry 'identity cards' allowing them to use any GSM phone at their disposal.

In video signal processing research is being conducted in source coding and filtering (pre- and post-processing), and it is focused on the following areas:

- Nonlinear methods. Currently the emphasis is in signal restoration and image enhancement operations. Not much has been done in coding. Methods under study include rank-order filters and mathematical morphology.

- Parametric signal processing. This branch of research deals with well-specified types of image sequences, most notably head-and-shoulders sequences. It is based on creating a (either two- or three-dimensional) model of the person speaking and then estimating the parameters of the model and transmitting them. It employs techniques of computer graphics and computer vision. The performance of these methods are not yet satisfactory but they are considered as strong candidates for the future especially in the field of very-low bit rate coding (MPEG4).

- Motion estimation. This part is critical for predictive coders both predictive and model based coders because it is the chief method to achieve redundancy reduction or prediction gain. More is said about this in the next chapter.
- Fractal coding is an offspring from fractal graphics. The idea is to find a suitable contractive function and use it repeatedly to the image to be coded until it converges to a small set of values to be transmitted. There have been claims of very high compression ratios using fractal methods but they have been unverified. Reliable sources report compression ratios of 1:30 at best.

## 1.2 Framework of the thesis

This work has been done within the COST211ter group (it will be referred to as "the COST group"). Therefore, the nature of the work is largely set by this group. As said earlier, the aim is to propose a standard for CCITT for audiovisual services at $p \times 8$ kilobits/s. The goal is to develop the already existing H.261 coder for this purpose. This means firstly that there were strict constraints over the work, and secondly that the research was done to solve a number of separate problems whereas the other bodies worked on other problems. The group does not concern itself with the audio coding part, and this thesis work addresses only the source- and rate/distortion coding of the incoming (digital) video signal.

The biggest single task was to build a software simulator for the simulation models agreed upon within the COST, and to verify their performance. This simulator was then modified as needed to study the problems to be solved. The ultimate goal was to produce as high subjective image quality as possible, as it is expected that the future consumer will refer to a subjective measure when making a choice between various standards. Thus, methods were evaluated on a subjective basis, mainly by viewing alternatives simultaneously.

The problems addressed were the following:

- To construct simulators for the simulation models (three of them: SIM1 (September -92), SIM2 (December -92), and SIM3 (March -93) ) and to evaluate their performance.
- To simulate the two proposed transmission image sizes (QCIF and NCIF) and to recommend either of them.
- To develop and simulate bit rate regulation methods.

## 1.3 Outline

This section gives a brief outline of the thesis. There are six chapters including this one:

Chapter 2 Defines the simulation models. This is based on work done within the COST group.

Chapter 3 Discusses the determination of the image size used in transmission (to separate it from the image that is presented to the viewer). The filters used were agreed upon in the COST group, but the evaluation of these filters together with the reverse-engineering work done on these filters are original.

Chapter 4 Gives statistical data of the performance of the coder without bit stream regulation, with and without the so-called "forced update". All material in this chapter is based on simulations done for this thesis. It should be noted that forced update is not yet supported by the COST simulation models, so the implementation is original but based on recommendation given in H.261.

Chapter 5 Discusses bit rate regulation, which means mechanisms to select the quantizer step sizes and the temporal decimation factor. In this chapter two methods for bit rates regulation is constructed and they are compared to a reference method by COST. Original parts include the design and evaluation of the two methods and the implementation and evaluation of the reference method.

Chapter 6 Draws the conclusions from this work

Appendix A Contains the tables defining a binary representation for each symbol used in communication between the transmitter and the receiver. These were given by the COST group and were used to compute the amount of bits generatated.

Appendix B Gives a structural representation of the bitstream created.

Appendix C Contains two sets of simulated images. One has simulated images supporting material of chapter 3, and the other has simulated images for chapter 5.

# Chapter 2

# The Source Coder

This chapter explains the structure of the source coder of the simulation models used by the COST group. This coder resembles somewhat the H.261 source coder and differences will be pointed out.

To simplify the discussion and to clarify the structure of the coder it will be broken into two parts which are called the *frame coder* and the *rate/distortion coder*. The frame coder contains those parts of the coder directly dealing with the images: frame buffers, motion estimation, mode selection and the transformer. The rate/distortion coder takes the data produced by the frame coder and assigns bit representations for these data and manages bit stream regulation (adjustment of the quantizer step size and determination of the temporal decimation factor). Figure 2.1 depicts this division.

We will assume arbitrarily that the coder receives a sequence of digital images in the CIF (Common Intermediate Format) format (the motivation for this approach is given later), and that it will output a bit stream for further channel coding.

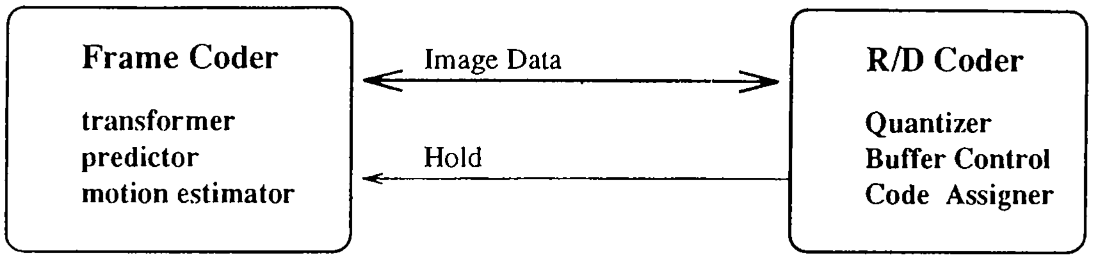


Figure 2.1: Division of the source coder into Frame coder and R/D (rate/distortion) coder. The hold signal is the physical realization of the temporal decimation factor: it tells the image coder not to grab a frame from the frame stream while the signal is active.

| Size | Resolution | | Macroblocks | Blocks,Y |
|---|---|---|---|---|
| | Y | U,V | | |
| CIF | 352 x 288 | 176 x 144 | 22 x 18 | 44 x 36 |
| QCIF | 176 x 144 | 88 x 72 | 11 x 9 | 22 x 18 |
| NCIF | 112 x 96 | 56 x 48 | 7 x 6 | 14 x 12 |

Table 2.1: Image sizes for the CIF family, in pixels, in horizontal x vertical order. QCIF is a quarter of CIF, but NCIF is only approximately a ninth of CIF (slightly less).

## 2.1 The structure of the frames

### 2.1.1 H.261

H.261 operates on two resolutions, CIF and QCIF (Quarter CIF). The CIF format is a color video format in the YUV-space. The YUV format for color image representation consists of the black-and-white component called the luminance (or luma for short) Y and two colour difference signals called the chrominance signals (or chroma for short) U, V. This signal space is inherited from the analog tv-technology and it has the advantage that the chrominance signals can be sub-sampled without a big visible degradation. Table 2.1 lists the resolutions of various components for the CIF, QCIF and NCIF formats, the last to be dealt with in the subsequent chapters. It is seen that the chroma resolution is a fourth of the luma resolution.

The image is further divided into *macroblocks* which in turn consist of six *blocks*.

Of the six blocks, four are taken from the luminance component and one block is taken from each chrominance component so that each macroblock represent a unique full-color spatial area (see figure 2.2.) The size of each block is $8 \times 8$ pixels, which is the size used for the discrete cosine transform. Each pixel is represented as bytes, i.e. with 8 bits of information giving 256 levels to represent the amplitude of the signal.

In H.261 the image structure has an intermediate level called the *group of blocks layer (GOB)*.The CIF frame is divided into $2 \times 6$ GOBs and the QCIF image has $1 \times 3$ GOBs. Each GOBs consists of $11 \times 3$ macroblocks. This level is abandoned in the COST simulation models.

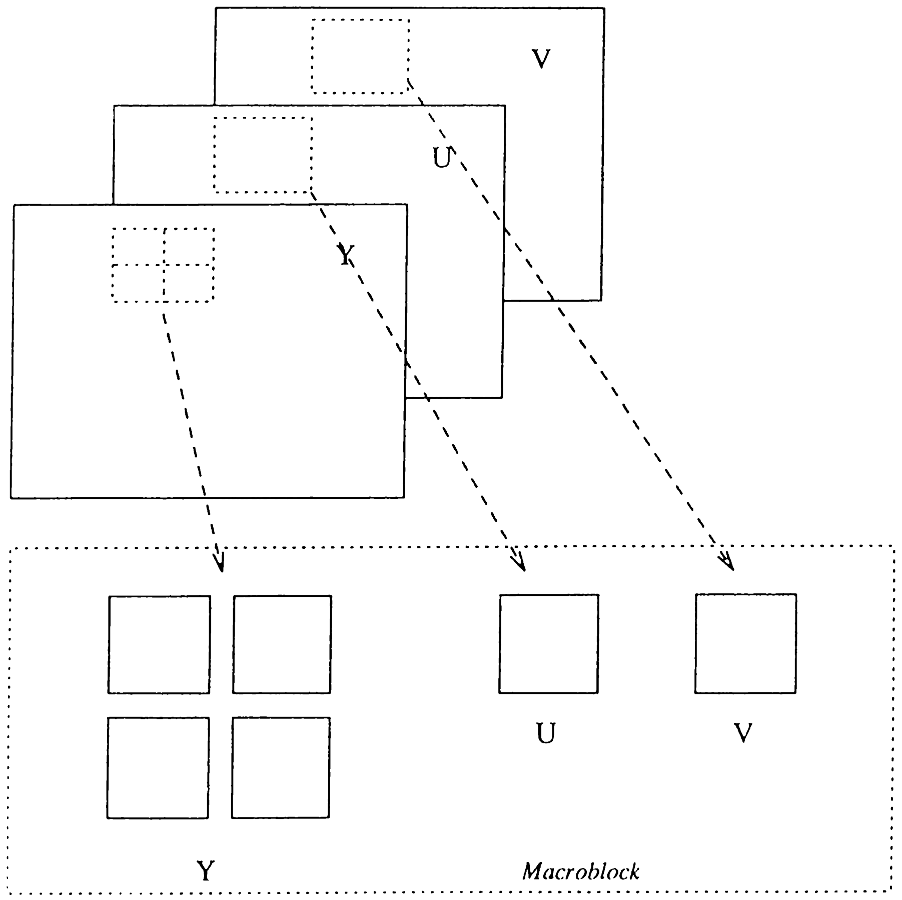


Figure 2.2: The structure of the macroblock

### 2.1.2 COST

COST simulation models (SIM) have determined CIF to be a hypothetical display resolution but because of the very low bit rates required, the image sizes used in transmission have to be smaller. Two potential formats were considered for this, the QCIF and NCIF formats 2.1. The decision between these two formats are discussed in the next chapter. The frame structure in QCIF and NCIF is simplified from that of H.261 by omitting the GOB layer.

In the following discussions we will work entirely with those image resolutions used for transmission and assume without explicit reference that the coder is interfaced to the input frame stream and to the display device with the appropriate decimators/interpolators.

## 2.2 The Frame Coder

### 2.2.1 The Discrete Cosine Transform

The heart of the coder consists of the discrete cosine transform which is the tool used in redundancy reduction. Mathematically it is defined for the H.261 and the SIM models as

$$F(u,v) = \frac{1}{4}C(v)C(u)\sum_{x=0}^{7}\sum_{y=0}^{7}\cos[\frac{\pi(2u+1)x}{16}]\cos[\frac{\pi(2v+1)y}{16}] \tag{2.1}$$

where

$$C(z) = \begin{cases} \frac{1}{\sqrt{2}}, & \text{if z=0} \\ 1, & \text{otherwise} \end{cases}$$

for the forward transform. The inverse transform is:

$$f(x,y) = \frac{1}{4}\sum_{u=0}^{7}\sum_{v=0}^{7}C(v)C(u)\cos[\frac{\pi(2x+1)u}{16}]\cos[\frac{\pi(2y+1)v}{16}] \tag{2.2}$$

It is noted that the transform is underscaled so that the transform domain coefficients range from -2048 – 2047.

The most important features of the DCT are:

1. For correlated data, it is the near-optimal transform to use in terms of repacking the signal energy into a few transform coefficients which also are well decorrelated [2]. Images tend to have a low-frequency nature, meaning correlated data.

2. It is efficient in bit allocation since the low-frequency coefficients contain most of the energy. In practice, this means that if we send only the dc-values of the blocks we will get an image that roughly resembles our original image. If we then increase the number of coefficients transmitted one by one we will get the original image in increasing degrees of resolution. Therefore, it is said that DCT is efficient in bit allocation, i.e. it is easy and straightforward to make the rate/distortion trade-off.

3. There are fast algorithms to compute the DCT making it also computationally attractive when compared to some other methods of signal compression (notably vector quantization).

Plain DCT coding where the image data is just DCT coded and sent make up the so called *intramode* coding part of SIM2 for reasons explained later.

One of the most successful ways to code images to this date has been the combination of DCT with predictive coding, and these coders are often referred to as *hybrid DCT coders*. Both H.261 and SIM models employ this coder structure.

## 2.3 Predictive Coding

Another way of reducing the redundancy of a signal is to predict the future values of it by using the knowledge of its history. This prediction can be done spatially (within a frame) or temporally (based on the previous frame(s)). We call spatial prediction intraframe coding since all operations consider data within a single frame, and temporal prediction (as well as spatiotemporal prediction, a combination of the two prediction methods) as interframe coding since we use two or more frames. Usually we have a choice between these two ways of predicting. Therefore, we say that a given region is coded in either intermode or intramode. The plain DCT discussed earlier is an intraframe coding method and since in our coder we shall not use any other intraframe method we named it intramode.

Generally, in those regions of the image, with low or no motion, temporal prediction produces better results whereas in moving areas spatial prediction is favorable [4]. However, plain predictive coding that utilizes both temporal and spatial prediction cannot achieve enough compression in the data rate that is required is most of the applications be it linear or nonlinear [5]. But interframe coding has the property of producing zero prediction error in no-motion and near-zero error with small amounts of motion which in effect leads to a decrease in the picture area to be coded, making it a suitable companion for DCT.

## 2.4 Hybrid Coding

It therefore makes sense to combine the DCT with a temporal prediction of the image data. Since this prediction is most efficient when done in the direction of motion, the procedure utilized is called *motion estimation/compensation*. The combination of DCT and temporal prediction is achieved as follows: since DCT imposes a block structure on the image, the motion is estimated in a block basis where for each block in the current frame (the frame to be coded) we try to find the closest match for it in the previous coded frame using some suitable distortion criteria (we must use the coded frame since it is common to the receiver and transmitter). The area used in this search is of course limited and it is called the *search area* and it is defined as an offset from the origin of the block for which we want to find a match. This offset is usually chosen to be a power of two for efficient coding. By minimizing the distortion within the search area we find a *motion vector* for each block which represent the optimal values for the offset.

Based on the previous discussion it would sound reasonable that we would code only those parts of the image which cannot be predicted well with DCT. This, however, is not the case. The coder can be simplified, as follows.

Once the vectors are found we perform a motion compensation where using the motion vectors and the previous coded frame we make a best approximation for the current frame to be coded for all blocks. This is a prediction image. Note that the receiver having only the previous frame can perform the same operation if it is provided with the motion vectors. What is done is that we use this prediction for all blocks to obtain an error signal which is then transformed using DCT and sent. As this is somewhat contradictory to the previous reasoning (a difference signal is not bound to have a strong dc-component which could be efficiently compressed by the DCT) this part will be elaborated.

As was mentioned earlier, the motion estimation/compensation method used a block approach which in turn was imposed by the DCT. This is not a realistic assumption for real-life images so in practice if we were to view the blocks of the prediction error image (which are to be transformed by DCT), we would find that for those blocks containing errors the error signal is usually the result of an intersection of the square block with an area which consist partially of background and partially of a moving object. That is to say that while part of the block will have zero amplitude, there are regions with a significant amplitude. This effect will reduce if we reduce the size of the block, and the error signal will become more noiselike, but in this case the block size is $16 \times 16$ which is a large one when compared to the overall frame dimensions. Therefore, the type of error signal described here is suitable for DCT processing.

As all blocks are processed using DCT it is time to check what happens with those difference blocks having either low-signal or high-signal content ("signal" is used here as a difference from zero, and "high" and "low" measure the amount of this zero difference with respect to the area covered within a block). Low-signal blocks (successful prediction in stationary parts) are zero except for noise occurring in the imaging devices and the effects of quantization as we are predicting between a coded and an yet uncoded frame. Of these components, the noise factor is typically small in amplitude and white in nature so it will cause a very small signal in all

DCT bands. Differences resulting from the coding of the images are mostly due to the truncation of some low-amplitude transform coefficients because of quantization and it is therefore very likely that the effects of these will again be quantized out. As a result, the coding of these blocks will usually produce zero output to the channel, as desired. For high-signal blocks (unsuccessful prediction, parts with contain motion) where the estimation has failed, we note that we attempted to predict these parts of an image with another part of the image. In this case subtracting a low-frequency signal from another results in yet another low-frequency signal (this must be so since we know that our prediction has failed), and the arguments for using the DCT are still valid in this case.

These considerations simplify our coder in the following manner:

- We now perform the same operations (motion compensation and DCT) for all blocks.
- Since all blocks are processed through DCT we can suffice with a single set of source code alphabet. That is to say, we do not have to design another set of codes for another coding method since, in this case, we can use all methods through DCT.

The latter simplification is more significant. But the coder also has a pure intramode (use DCT without any prediction) in order to send the first frame and to be able to use forced update. In section "Mode Decision" we find another possibility for its use.

So in order to keep things simple the abovementioned approach is chosen. There are few points to note:

- With image sequences there are timing constraints in processing between consecutive frames. Since the search for motion vectors is the most time consuming operation in the coders the search is sometimes done in the macroblock level (this is the case in H.261). This of course strengthens the arguments described above concerning the nature of the prediction signal.
- In video frames represented in the YUV-format the search is done at the Y

component.

- The motion estimation can only be done at the transmitter since only it knows the current image. Thus this way of coding is noncausal and there is a need to transmit side information (the motion vectors). However, the structure of the receiver becomes very simple, and the extra costs of sending the vectors are more than balanced by the savings in the bit stream when compared to straightforward causal DPCM-type prediction.

- The method for motion estimation as described above does not take into account that motion is rarely an exact multiple of a pixel. One way to improve this situation is to use fractional-pixel accuracy. The one most commonly used is *half-pel accuracy*.

- It is seen that the argument for using DCT for the error signal is due to the blockwise motion estimation and compensation. To obtain better prediction results a finer grid of motion vectors is required. Part of the current research activity in video signal processing is devoted to find more efficient motion search algorithms that would give a finer grid of motion vectors. As the predictions get better the arguments for DCT get weaker since the signal will have less regions of constant prediction error and less low-frequency nature in general. Instead of DCT, *vector quantization* may be used. Further, the square block structure could be abolished in favor of a more suitable shape.

- Temporal prediction relies on that both the sender and receiver have identical images so that the prediction image created at the sender and duplicated at the receiver with the help of the motion vectors would be identical. We may therefore view the coder and decoder (=codec) as state machines where the previous frame is the state. A problem associated with this is discussed in the next paragraph.

One practical problem arises when combining predictive coding and DCT this way, namely the matching problem. For example, H.261 does not give specifications on the algorithm to be used in the computation of the inverse DCT. It only specifies the accuracy of the inverse transform. This leaves some space for algorithm design but also permits that the outcomes of different algorithms may vary. In this case it means that the decoder and coder inverse transforms produce slightly different results, which in combination with the predictive coding accumulates this variation leading to the need to refresh the data within a macroblock from time to time by coding it in pure intramode. This is referred to as *forced updating.*

## 2.5 The SIM1/2/3 Frame Coder

### 2.5.1 Motion estimation

The frame coder is depicted in figure 2.3. The biggest difference with the previous discussion is the more complicated motion vector search which is done in three stages (MV1–MV3). In stage MV1 a full-pixel accuracy, macroblock-scale motion vector search is done on a search area consisting of $\pm 15$ pixels vertical and horizontal. The search is done between the current frame and the previous uncoded frame. The reason for not using the previous coded frame in this first stage is that in very low bit rate coding coded images suffer from strong blocking effects, i.e. images have areas of constant dc value, so that the motion vector search algorithm may find many minimal points. Depending on the implementation of the motion vector search this may push motion vectors into extreme values, i.e. the results depend more on the algorithm used to find motion vectors than on the underlying true motion vector field[1] This effect can be avoided by first searching a seed vector from the previous original frame.

The distortion criteria used is SAD (sum of absolute differences), defined as:

$$\mathrm{SAD(x,y)} = \sum_{i=0}^{15}\sum_{j=0}^{15} |C(i,j) - P(x+i, y+j)|$$

where $C$ is the current block and $P$ is the block in the previous frame. The SAD(0,0) is reduced by 100 to favor zero motion vector when the difference is not significant (this is done in all the subsequent stages, too).

[1]It has been observed that blockwise motion vector search algorithms fail to give an accurate reproduction of the motion vector field.

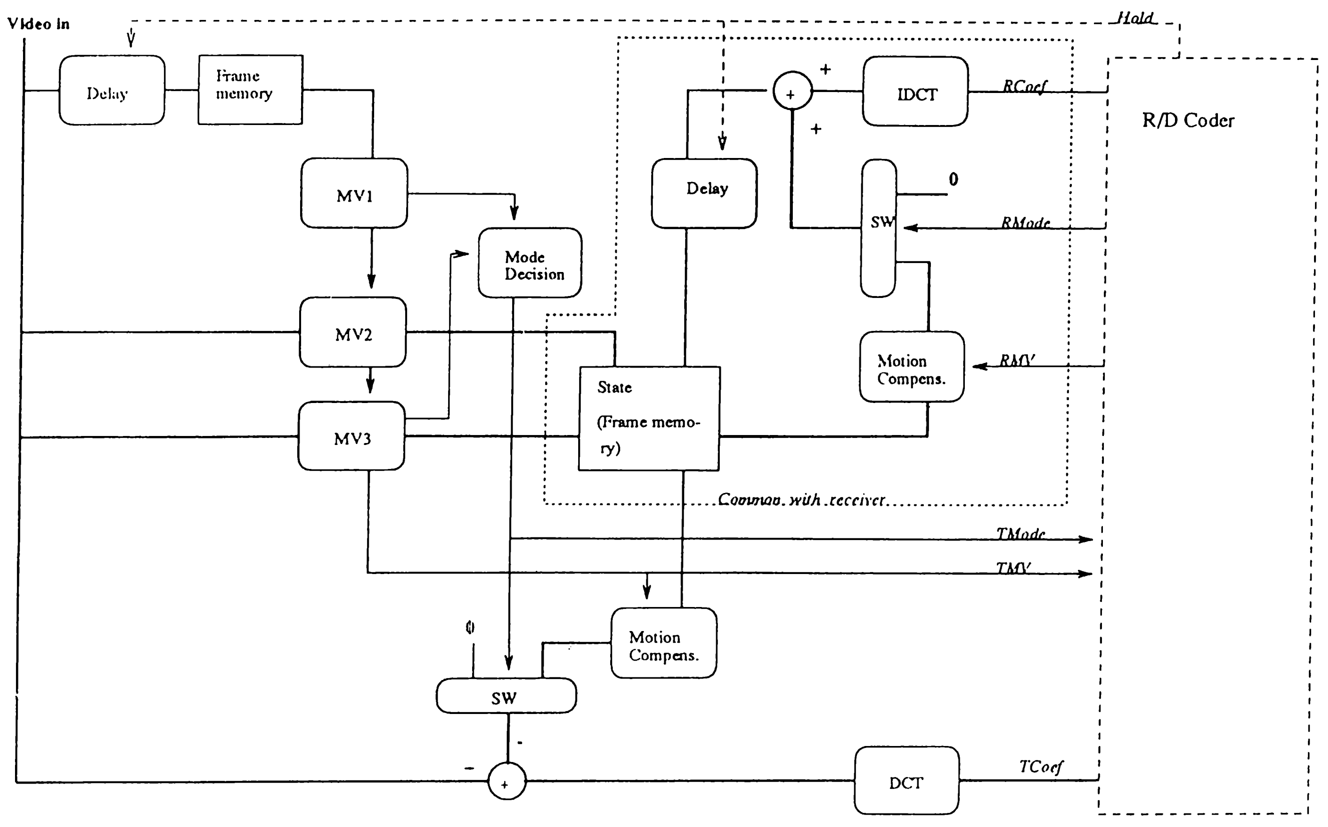


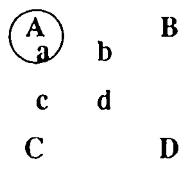
Figure 2.3: The structure of the frame coder

The second stage, MV2, is a half-pel search over a search area of ±1 half pixels around the motion vector given by the first stage. The half-pixel values are interpolated in a straightforward manner (see figure 2.4).

In the final third stage the macroblock structure is broken and the motion vector search is done separately for the four Y blocks. The search area is ±2 half pixels around the motion vector found in stage 2. The SADs associated with the four optimal vectors ($SAD8_i$) is compared to the stage 2 result, and if the following condition is satisfied:

$$\sum_{i=1}^{4} SAD8_i < SAD(\text{stage } 2) \times 0.9 - 100$$

then the prediction is done in block basis. SADs usually take values between 0 – 3000.



**a=A** **b=(A+B)/2**

**c=(A+C)/2** **d=(A+B+C+D)/2**

Figure 2.4: Formulas for interpolating missing pixels for half-pixel search. Note that capital A and small a denote the same point.

### 2.5.2 Mode decision

We have three coding modes at our disposal: INTRA, INTER-1, INTER-4. INTRA means intramode coding, using DCT without motion estimation/compensation. Modes INTER-1 and INTER-4 are for hybrid coding using one or four motion vectors, respectively. The coder determines the best mode for each block with a number of rules described here.

Mode decision is done at two stages: first to determine whether to use intra- or intermode for coding. If the intermode is chosen a further decision is needed on what kind of motion compensation is used. The latter decision have been described at the end of the previous section.

The first decision is made after the first stage of motion vector search. Here we attempt to estimate whether the block comprises mainly of a single value. If it does, we will choose intramode, otherwise we will continue with the motion vector search. Specifically, the intramode is chosen if

$$\sum_{i=0}^{15}\sum_{j=0}^{15}|C(i,j)-\bar{C}| < (\mathrm{SAD}(\text{stage }1)-500)$$

where $\bar{C}$ is the average of the values inside the block $C$. Further motion estimation is done only with blocks that are chosen to be coded in intramode.

It may seem surprising that INTRA mode is still considered in the course of normal coding (in all simulations done in the course of this work the coder never chose an

INTRA mode spontaneously (without being forced to do so)). Looking at the rule for choosing INTRA mode we see that it requires that the values are strongly concentrated around one value. Further, if there is an area with this property within the search range in the previous image our algorithm will find it and turn out a SAD of very low value. But, in the case where quantization has shifted the value of that level it might be more advantageous to send the true level with fair resolution (in chapter 5 we see that the dc coefficient of an INTRA coded macroblock receives special treatment allowing it to be transmitted with high accuracy; the blocks considered here are likely to transmit only that one dc-coefficient) than to code a difference which, since it will be quantized, may again shift the level to another value resulting in oscillation and unnecessary coding.

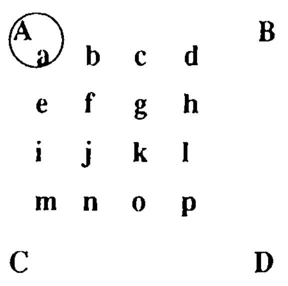
### 2.5.3 Prediction

Having got the motion vectors for those blocks to be intercoded a prediction is done. The prediction for the Y-component presents no problem as we use the values we interpolated during the motion vector search. For the chrominance components it is more tricky, since they are sub-sampled already. It turns out that because of this subsampling and half-pixel motion estimation there are no less than 16 possible interpolations between the true samples. This is depicted in figure 2.5 together with the filter definitions.

### 2.5.4 Summary of the frame coder

As a summary, there are three modes of coding, which henceforth will be labeled as follows: intramode coding with no prediction (INTRA), intermode coding with a single motion vector per macroblock (INTER1), and intermode with four motion vectors per macroblock (INTER4).

The transmitter sends to the receiver motion vectors, mode information, the quantized transform coefficients, and the quantizer step size. The coding of all this data is done at the rate/distortion coder, which sends back to the frame coder the same information it sends to the receiver so that both ends have the same information.



| | | | |
|---|---|---|---|
| a=A | e=(3A+C)/4 | i=(A+C)/2 | m=(A+3C)/4 |
| b=(3A+B)/4 | f=(9A+3B+3C+D)/16 | j=(3A+b+3C+D)/8 | n=(3A+B+9C+3D)/16 |
| c=(A+B)/2 | g=(3A+3B+C+D)/8 | k=(A+B+C+D)/4 | o=(A+B+3C+3D)/8 |
| d=(A+3B)/2 | h=(3A+9B+C+3D)/16 | l=(A+3B+C+3D)/8 | p=(A+3B+3C+9D)/16 |

Figure 2.5: Predicting chrominance pixels. Capital letters denote true samples and small letters interpolated ones. (Except for capital A and small a which denote the same point). In the filters, integer division with rounding towards nearest integer is used.

The R/D coder can freely process all data handed to it because it notifies both the receiver and transmitter of its operations. The receiver gets all its data through the R/D-coder which also returns the data it transmitted back to the frame coder of the transmitter. In this manner both transmitter and receiver can reconstruct exactly the same frames so that successive motion compensations produce the same results (i. e. the states remain the same).

## 2.6 Rate/Distortion Coder

The Rate/Distortion (R/D) coder is depicted in figure 2.6. Here both the transmitter and the receiver parts are depicted. The blocks labeled CA (code assigners) at the sender part associate with each of its input alphabet a Huffman code. The corresponding decoders (DC) in the receiver perform the inverse operation. Further, at the transmitter preceding the code assignment there are modules to process each of the three data items. This processing is in general connected to bit stream regulation. The most important one is the quantizer which quantizes the transform coefficients. Following the quantizer is the runlenght coder (the corresponding DC in the receiver performs both the inverse code assignment and the runlenght decoding).

The mode data is combined with *coded block pattern (CBP)* data, which tells which blocks within the macroblocks have coefficients different than zero.

In the diagram, an option is reserved for the possibility of modifying the motion vectors also, although this is not currently used. The only data to be subject to bit stream regulation is thus the transform coefficients. This has some implications in the final bit stream. In very low bit rate coding it is often the case that due to heavy quantization many macroblocks end up having no coefficients to transmit. Large savings can be achieved if in these cases we can mark the whole macroblock as not coded (as is done in all of the COST simulation models, but this can only be done if the motion vectors are zero.

The bit stream specifications (output multiplexing) together with the Huffman codes are given in the appendix.

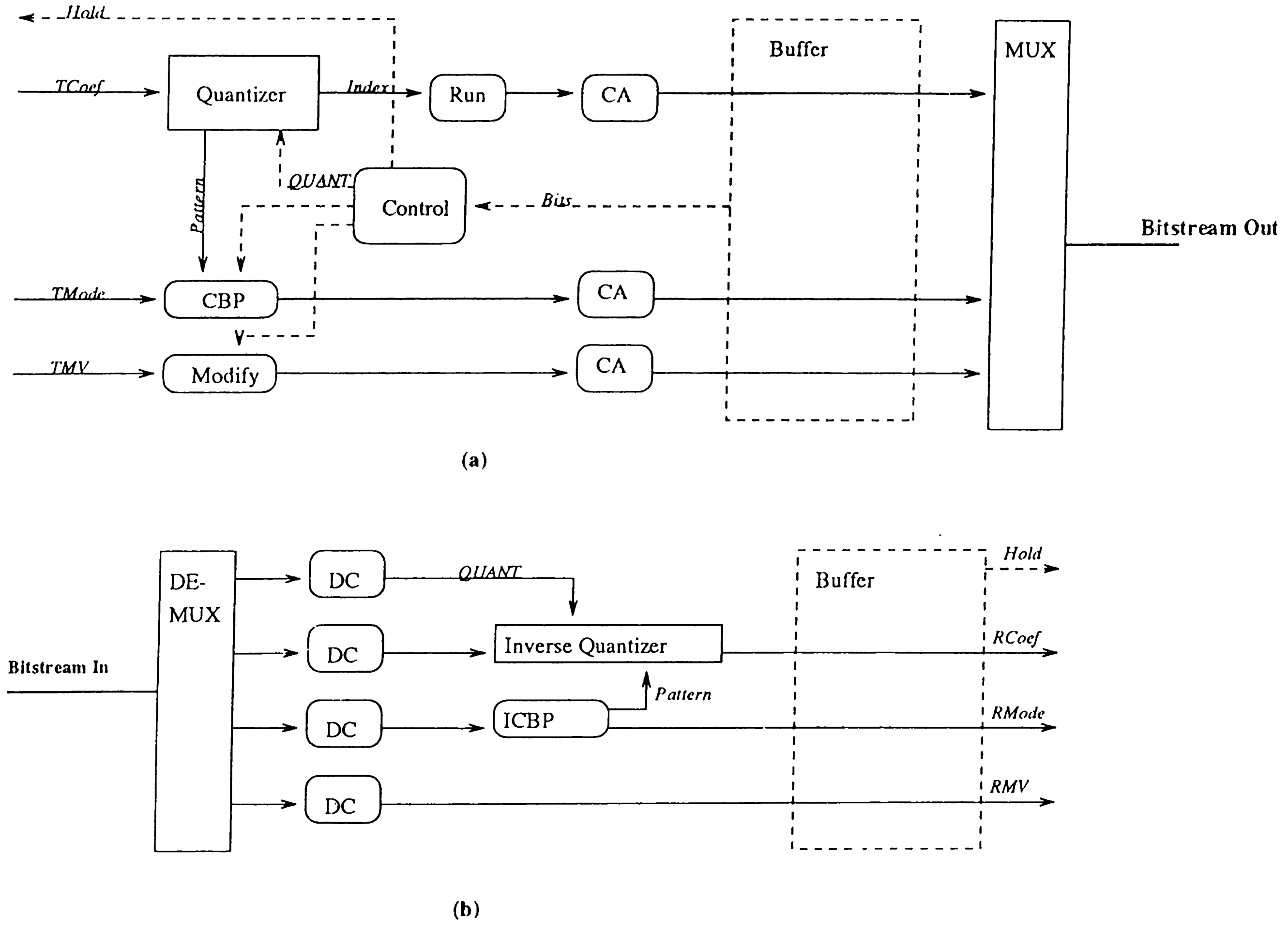


Figure 2.6: The rate/distortion part of the coder and the receiver. Note that the transmitter also has a "receiver" so that data integrity is preserved (see in figure 2.3 the dotted box): both the transmitter and receiver draw their data through the rate/distortion coder.

# Chapter 3

# Determination of the image size

## 3.1 Introduction

Previously it was noted that the COST group decided to use CIF as the hypothetical display size. The factor favoring CIF is its H.261 compatibility.

As we have chosen to assume that the size of the image is CIF, we have to look for transmission image sizes which comply with this assumption. The multirate signal processing tells us that the simplest (and fastest) implementations of decimation and interpolation are achieved when the display size is chosen to be an integer multiple of that of the transmission size. Hence there are two sizes to be considered for the transmission image size: NCIF and QCIF (see table 2.1). It is noted that we have other constraints as well, the image sizes should be integer multiples of macroblocks also. The QCIF size, which is exactly half of that of CIF in each dimension does not pose a problem, but NCIF, which tries to be a third of each dimension runs into trouble with this latter requirement. The horizontal dimension falls one macroblock short from CIF, and in our case we handle this problem as follows: we discard 8 pixels from left and right of the CIF image and process the rest.

Terminology: a *frame* or *image* refers to the transmitted image, and *display frame/image* refers to the image to be displayed.

It should be noted that if the frequency content of the display image is low then decimation serves as a redundancy reduction method as well. This can be seen in table 3.1 where the entropy of the CIF, QCIF, and NCIF is listed for the sequence *Claire* (for a discussion of this sequence, see below). Entropy is measured as

$$H(X) = -p(x)\log_2 p(x)$$

where $X$ is a random variable representing the image signal, and $p(x)$ is the empirical distribution of the pixel values measured for one frame. Values given in the table are average entropies over ten frames. It can be seen that the uncertainty of the signal has increased because of the decimation. Since our model sequence in this case has low noise content we can take the view that the uncertainty of the signal is due to the information it contains, albeit this interpretation is loose. The pixels of the decimated signal therefore can be thought of being more precious. On the other hand, the rather small increment of the entropy with decimation also indicates that information is lost.

A smaller transmission image size allows us to allocate more transmission capacity per pixel, so we can transmit more information of the image. But if we work with a large transmission image size, the quality of that image is better so perhaps we could afford to loose some of these details, and if our compression method works well the sacrifice might not be a big one. This can also be seen in our table for the entropies: since the increase in the uncertainty is not a large one when comparing QCIF and NCIF, our compression (redundancy reduction) method might pack the data into the same number of bits in each case. We can formulate the problem in two ways:

- Is it better to decrease the resolution of the frames or to increase the distortion due to coding ?
- Which is a more efficient rate reduction method for our coder: sub-sampling or quantization after transforming?

| Size | Entropy | | |
|---|---|---|---|
| | Y | U | V |
| CIF | 1.8990 | 0.5206 | 0.4561 |
| QCIF | 1.9282 | 0.5296 | 0.4624 |
| NCIF | 1.9810 | 0.5322 | 0.4565 |

Table 3.1: Average entropies, in bits per pixel, of the 10 first frames of Claire. The entropy was computed separately for each frame and the figures shown are averages over 10 frames using decimation and interpolation procedures as defined by COST (see section 3.2).

In the most important type of video telephone signals, the head-and-shoulders, it is usually the case that the person's face and hair does contain higher frequency components which suffer most of decimation. From the viewpoint of the utilized coding techniques, it is unfortunate that these regions are likely to capture most of the attention from the viewer so their quality requirements are critical.

In this section a sequence named Claire is used in demonstrations. (In the appendix there is a picture of the original image). This sequence is relatively simple one with uniform background and with the person occupying a small area of the total image area. However, with this setup the motion estimation is critical, especially for the 4-vector mode as moving parts are rather small. Secondly, in most of the cases head-and-shoulders images possess a stationary background, so their content have little effect on the coding output after the first few frames.

In this chapter we will first define the decimation/interpolation filters and then discuss their performance and design. After this we will look at the coding results and make our decision based on these results.

## 3.2 Filter definitions

The filters must meet a number of practical constraints that limit their optimality for their purpose. Since the main aim is to produce a cheap commercial product the output of the filters should be easily computed. The filters, then, have made some concessions from optimality to efficiency:

- they are to be used in the spatial domain (not frequency) thus avoiding fft-calculations
- spatial domain filters should not have a large region of support
- they should be separable
- they should employ integer arithmetic with such weights that allow the scaling to be done with a simple shift operation.

There are also a number of general requirements [6]: passband quality is as important as stopband attenuation, and interpolative filters should leave original pixels untouched. Also, the filters should have a linear phase response, which is satisfied when the coefficients of the filter are chosen to satisfy $h(n) = h(-n)$. However, space-variant interpolation is also utilized.

All the filter definitions were given by the COST group.

### 3.2.1 QCIF

The decimation is done by first low-pass filtering the CIF-image and then subsampling the result taking only every other sample. The filter equation is given in one dimension, and since it is separable, we can apply it either by convolving first rows then columns (or vice versa) with this one dimensional filter or treating this definition as a vector, taking the outer product and then by using two-dimensional convolution achieve the same results:

$$\frac{-1 \quad 0 \quad 9 \quad 16 \quad 9 \quad 0 \quad -1}{32} \tag{3.1}$$

This filter is used for both luminance and chrominance.

The interpolation operation is done by first inserting zeros in a "quincunx" manner, and then using the same filter as above over two lines simultaneously (see fig. 3.1), first in the horizontal direction. It is seen that when the filtering is done on two lines simultaneously, we can always find a nonzero sample at each column (see figure 3.2). This does not preserve the original samples. Note that by this method we filter two lines at the same time. After we have increased the number of columns to CIF dimensions, we proceed with the same operation on the rows.

The filter in 3.1 is a half-band filter where the coefficients are chosen according to the Lagrange interpolation formula [7]. The filter has an advantage that the filter coefficients directly satisfy the contraints concerning computational complexity: they are integers and the scaling can be done with a shift operation. The filter also possesses a maximally flat response. The one-dimensional frequency response is given in figure 3.2.2. Filtering tests show that the quality of the filter is good, images are fairly sharp and no ringing effect is present (see Appendix) (ringing effect can occur in filters designed by inverse transforming the ideal response: due to the high sidelobes of the sinc-function, there appears "echoes" of sharp edges in the image, much resembling the rings occurring when a stone is dropped in water).

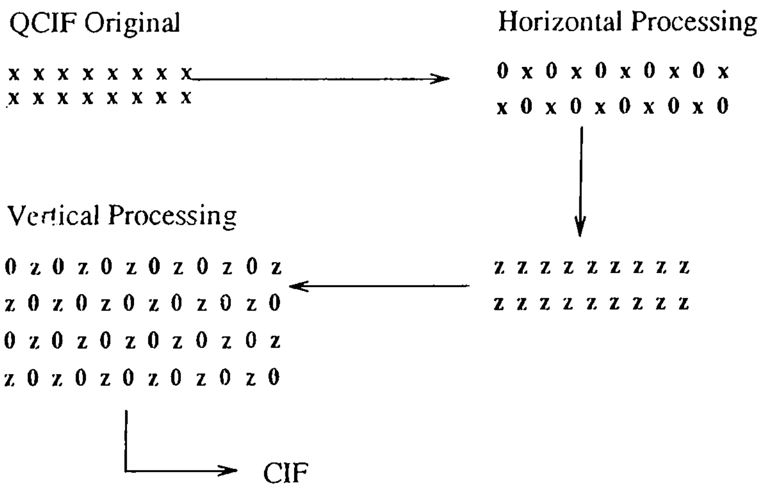


Figure 3.1: Interpolation from QCIF to CIF

### 3.2.2 NCIF

The filtering procedure is much more complicated this time. For the decimation part we use different filters for horizontal and vertical directions for the Y component, but for chrominance signals we use the same filter in both directions. Here

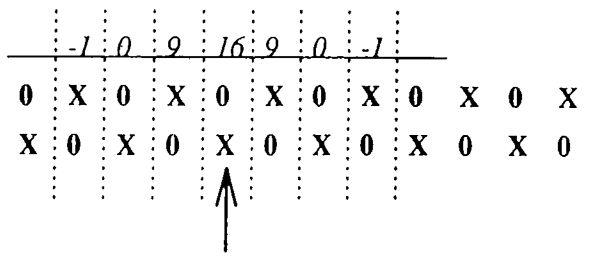


Figure 3.2: Interpolating QCIF images to CIF. In each column, the nonzero entry is taken as input to the filter. In the filtered image, the filter output is placed in the column marked by the arrow, in both positions.

is the filter for the horizontal filtering of the Y-component:

$$\frac{-2 \quad 5 \quad 31 \quad 59 \quad 70 \quad 59 \quad 31 \quad 5 \quad -2}{256} \tag{3.2}$$

For the vertical luminance and both vertical and horizontal chrominance the following filter is used:

$$\frac{2 \quad 3 \quad 6 \quad 3 \quad 2}{16} \tag{3.3}$$

Interpolation phase is also more complicated. In fact, this time no space invariant filter is actually used, but rather a complicated interpolation formula. We first interpolate horizontally by putting two zeroes between each sample, then use the formula 3.4 to interpolate a full horizontal resolution, then append two rows of zeroes between each row of this intermediate resolution and apply 3.4 in the vertical direction. Note that the original NCIF pixels are left intact.

$$\begin{array}{cccccccccccc}
X & 0 & 0 & X & 0 & 0 & X & 0 & 0 & X & & \\
-12 & & & 200 & * & & 75 & & & -7 & /256 \\
-7 & & & 75 & & * & 200 & & & -12 & /256
\end{array} \tag{3.4}$$

The stars denote the place where the filtered values end up. The $X$'s denote the samples inherited from the NCIF image.

The one-dimensional frequency responses of 3.2 and 3.3 are shown in figure 3.2.2. Considering the objectives given at the beginning of this section, the frequency response of the decimation part filters is not very satisfactory. Partly this is understandable from the so called uncertainty principle [3] that tells us that the quality of the response of a fixed lenght filter increases with the passband bandwidth. Filtering tests with the decimation filters show strong blurring effect, especially when comparing to QCIF (see Appendix). Also, some ringing is present.

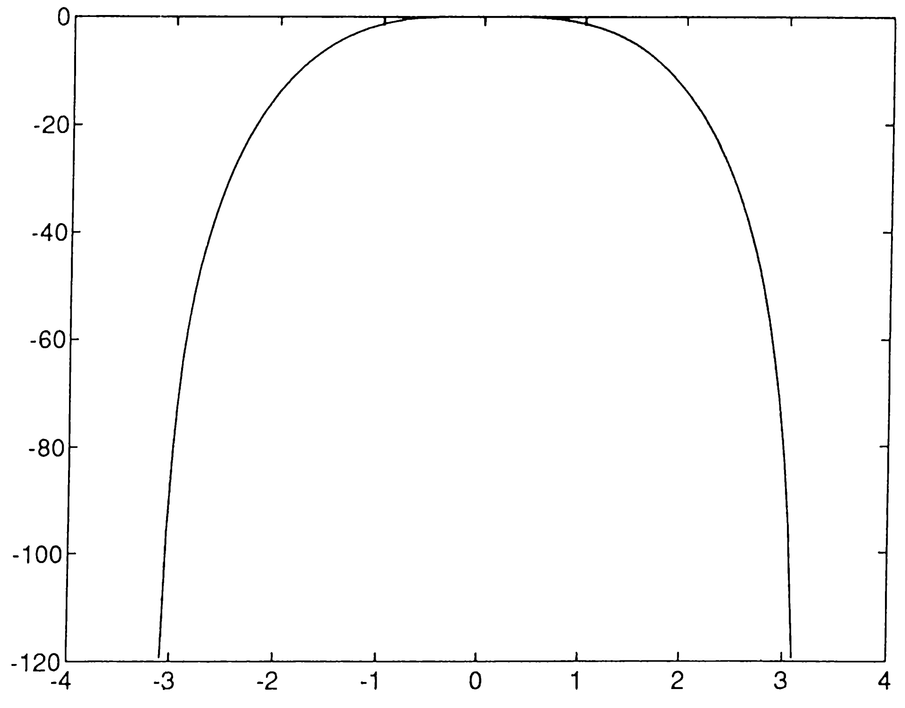
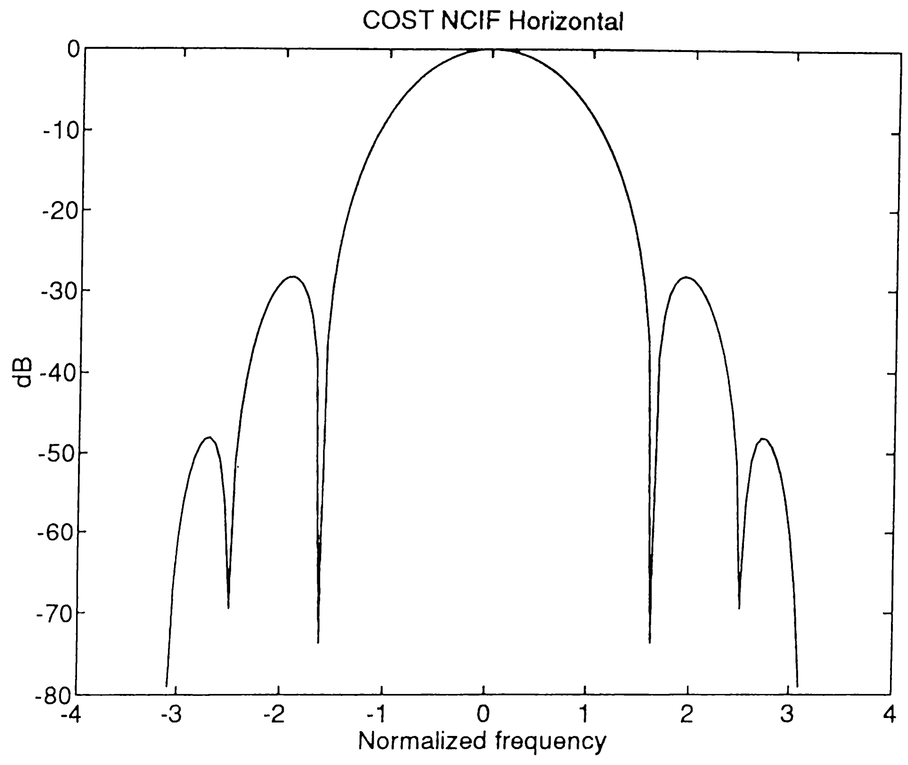


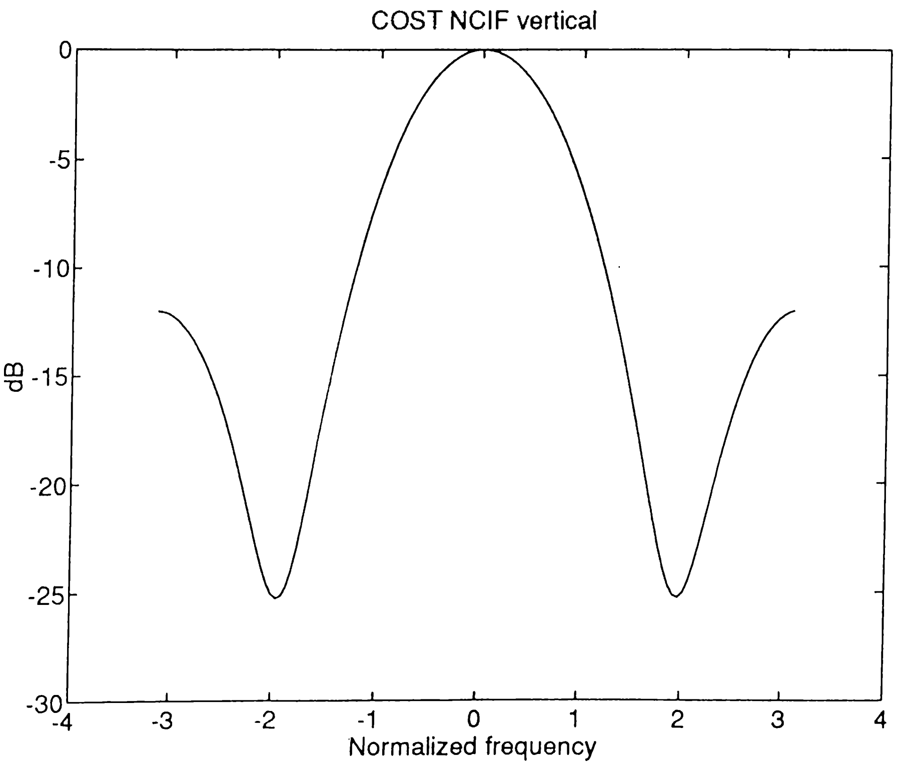
Figure 3.3: One dimensional plot of the frequency response of the QCIF filter.

## 3.3 Coding results

Next, coding was employed. At this stage the coder was stripped from its buffer control, and by trial and error suitable quantizer step sizes were found so that the output bit rate over 100 source frames taken at 25 Hz would produce a figure as close as possible to the desired bit rate. Although we used a source producing frames at 25 Hz, this rate was decimated so that the low target rates could be reached. Two picture frequencies (8.33 Hz and 5 Hz) plus three target bit rates (8, 16, and 32 kbits/s) were tried. Thus the number of frames in the 8.33 Hz case



(a)



(b)

Figure 3.4: One dimensional frequency response of (a) the horizontal and (b) vertical NCIF decimation filters.

| Rate | QCIF | | NCIF | |
|---|---|---|---|---|
| | 8.33 Hz | 5 Hz | 8.33 Hz | 5 Hz |
| 8 | 30 | 24 | 16 | 14 |
| 16 | 18 | 14 | 10 | 8 |
| 32 | 12 | 8 | 6 | 4 |

Table 3.2: Quantizer step sizes for the coders for various rates and with no buffer control.

is 33 and in the 5 Hz case 18 (the first frame is not included in the calculations). The quantizers used were uniform with the given step size so that the centermost representative value is zero. The quantizer step sizes are listed in table 3.2. As the step size is increased, more of the low-amplitude transform coefficients are turned into zero and hence the quality gets worse.

Although the step sizes for NCIF gets very low, even in the higher rates the picture quality is clearly better with QCIF. The biggest reason for this is that using the given interpolation/decimation schemes for NCIF are much worse for than those for QCIF.

It is of interest to examine the possibilities of NCIF in the absence of these constraints. For this purpose a $37 \times 37$ separable low-pass filter for decimating a CIF-image to NCIF was constructed. An image from the sequence Claire was filtered with the COST's QCIF filter and with the unconstrained NCIF filter designed here. Comparing, the NCIF version was nearly as sharp as the QCIF image (see the appendix for the images). This means that considerably more computing power has to be allocated for NCIF filtering in order to catch QCIF's quality, but at the same time there is no reason not to use the same computing power on QCIF material also. Why this is the case is concluded in the next section.

## 3.4 Conclusion

In this chapter, extensive attention has been drawn to the filtering stage of the decimation process in order to analyze the problem of transmission image size.

However, in the next chapter it is seen that the coder has been constructed in such a way that it attempts to code those parts of the image that are considered to be important. Filtering cannot, of course, take such subjective considerations into account. Therefore it is better to present the coder with good source material and let it decide how to allocate the bits. In this case, it means using QCIF images.

These results [12] were supported by other bodies as well, so the COST group decided to drop the NCIF format. Thus hereafter only the QCIF transmission coder will be considered.

# Chapter 4

# Coder Operation and Statistics

In this chapter the workings of the frame coder is studied. Also the distribution of bits when there is no bit rate regulation is studied. This will serve as the groundwork for the following chapter.

First the structure of the bit stream and a categorization for the consumption of capacity is given. Results are then presented according to this categorization. Then the framewise performance of the frame coder is presented and finally the effects of forced update are also treated. It should be noted that the subject of regulating the bit rate is the topic of the next chapter, and all results obtained here have been gotten without buffer control and using a fixed frame rate.

## 4.1 Bit Stream Structure

Details of the bit stream structure can be found in the appendix. Here the main points are summarized: the coder wants to communicate transform coefficients, motion vectors, coding mode and the quantization parameter. Adding to this there are some header information providing hooks for additional services and synchronization codes at the frame layer; the macroblock header has the coded flag (one bit telling whether the macroblock is coded at all), pattern information telling which blocks within the macroblock have coded coefficients, the coding mode is also here, and finally the last simulation model (SIM3) permits quantization information to be sent at the macroblock layer also. (If the macroblock is coded in the intramode, all blocks transmit at least the first cosine transform coefficient which is named *dc-coefficient*. The pattern data therefore tells which blocks have coefficients other than the dc.)

Finally, the block layer contains the transform coefficients. Previously it was mentioned that the transform coefficients are runlegth coded. To improve the efficiency of this coding a special scanning mechanism called zigzag scanning is employed, where the coefficients are read so that the low-frequency coefficients are scanned first (see figure 4.1). If the macroblock is INTRA coded, the first coefficient (which represents the average of all pixels in the block and is hence always positive) receives special treatment. It will be coded with a special quantizer that has a fixed step size of 8, and the resulting 256 possible levels are directly coded with 8 bits (remember that the amplitude range of transform coefficients is -2048 – 2047, and since the dc value is always positive we are left with 2048 values from 0 to 2047). The rest of the data, including all the other coefficients of INTRA mode and all coefficients of the remaining two modes are quantized with the adjustable quantizers. After the quantizer the coefficients are coded in a runlenght-amplitude manner, where the runlenght represents the run of zeros before the next non-zero coefficient. In the appendix there is a table of the runlenght-amplitude pairs and their respective codes. Eventually most of the high-frequency coefficients are zero, and when this last string of zeros is encountered, it is not sent but instead a short end-of-block code is transmitted.

Motion vectors are coded differentially. If the macroblock is coded in the four vector mode, the motion vector for the upper left block is established first and the other vectors are sent as offsets from it. If the previous block was coded also in four vector mode, then the first vector is coded as an offset from the upper right motion vector of the previous macroblock. If the previous vector had only one motion vector, that one is used as the base vector for the first vector of the current block. Likewise, if the current macroblock to be coded happens to be of one vector type, then the base vector for prediction is either the only vector or the upper right vector of the previous block. Only consecutive INTER-blocks are coded in this

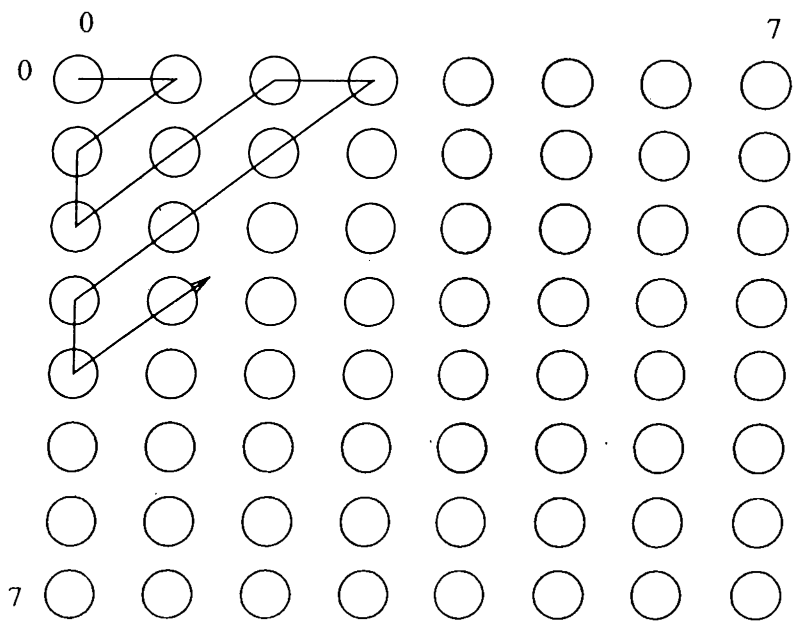


Figure 4.1: Zigzag scanning of transform coefficients.

way, the prediction is set to zero on all other occasions.

In short, the prediction process is the following: a predictive vector is chosen by picking the only motion vector from the previous macroblock if it is a INTER-1 coded one, or the upper right one from an INTER-4 macroblock. This vector, then, predicts either the upper left motion vector of a INTER-4 macroblock or the only motion vector of a INTER-1.

Three categories for bit consumption arises naturally:

1. Header and mode data
2. Motion vector data
3. Transform coefficient data

Also some statistical information is drawn from the coder: which modes were used, how many nonzero coefficients were there per block and what was the length of the last string of zeroes. We will also look into the motion estimation process to see the effect of the half-pixel accuracy motion vector search, and the nature and amount of macroblocks having a zero motion vector.

These quantities were measured using the two frame frequencies (8.33 HZ and 5 Hz) and three different rates (8, 16, and 32 kbits/s), but here the emphasis will be on 8.33 Hz and 8 kbits/s. In the worst case (8.33Hz-8kbits/s) we have 8000/8.33 = 960 bits per frame at our disposal. This means an average of 960/99 = 9.7 bits per macroblock supposing all of them were coded and excluding framewise header data.

Having now faced what very low bit rates mean in practice, it is evident that we have to reserve special treatment to the first frame which we have to code in INTRA mode in its entirety, because 960 bits are not sufficient even to send the dc-levels of each block. Common practice in simulating is to code the first frame with a fixed quantizer step size and "wait" until all data of the first frame is sent before starting ordinary coding of images. In this chapter the first frame is coded with the same quantizer used elsewhere in the sequence, and this quantizer is chosen such that the average bit rate over a given number of source frames is as close as possible to the desired rate. The number of frames used in simulations depend on the frame frequency used. Our source outputs a total of 100 frames at 25 Hz, so if we use 8.33 Hz we code 34, and with 5 Hz 19 frames.

## 4.2 Results without forced update

Let us first look at the motion estimation process (this part is common to the forced update case also). The first striking feature is that INTRA mode is never chosen in the normal operation. This is because even the normal electrical noise of the camera device present also in the digitized images is sufficient to create enough variation of levels even in seemingly uniform areas of an image. It is likely that this mode is not meant to be used in connection with natural images (other than in forced update). Next we try to assess the meaning of half-pixel search. The validity of the motivation behind it clearly increases when resolution gets coarser. To obtain a fair comparison, let us neglect those macroblocks with zero motion vector. We will consider the case where the frame rate is 8.33Hz and the transmission rate 8kbits/s. In ten coded frames there were 228 coded macroblocks, of which 112 (49%) had nonzero motion vectors. Of these, 80 (71%) were coded in 4-vector mode, a clear indication of its usefulness.

In the three figures 4.2, 4.3, and 4.4 the framewise distribution of these modes have been shown for rates 8, 16 and 32 kbits/s, respectively, and for a frame rate of 8.33 Hz. It can be seen that the coder is successful in allocating the 4-vector mode to those parts of the region where most complicated motion is encountered (the face and hair). One-vector modes are used at edges between the person and the background where the motion of the whole block is uniform. Also the one-vector mode with zero motion vectors are used at boundaries of the person and background where only a small part of a macroblock is effected by the motion. Further it can be seen that when the bit rate is increased, the coder successfully favors areas which are of importance to the human observer. (Some caution is appropriate in judging these images as they are only approximate silhouette drawings. The differences between the modes that have been coded in all frames are due to the different quantizers used (15, 9, and 6, respectively — no bit rate regulation is used)).

Table 4.1 shows what the respective proportions of the various categories of the bit stream are. Here we emphasize the results where the workings of the coder are manifested, the frames 2 – 34. With respect to the previous categorization we have 26% (273.606 vector bits of 1045.3 bits used per frame on the average) of motion vector data, 49% (493.061 luma bits and 21.2424 chroma bits) of transform coefficient data, and 25% of header data (the rest). On the average 23.7 coded macroblocks had 36.2 nonzero blocks (making the number of nonzero blocks in a macroblock to be 1.53) so that there is motivation for sending the pattern information. We are also sending 23 times more luminance information than chrominance information underlining the importance of the former. Lastly, we note that nearly 52 zeros are found at the lower-right part of the transformed block, leaving 12 coefficients for the upper left, but even of these 12 only 2 are coded and sent. Thus there are considerable runlengths in the upper part as well. Lastly, an average frame sends 72 transform coefficients to the receiver for reconstructing the next frame, a surprisingly low value. The comparison of the data between the first frames (all INTRA coded) and the rest (INTER coded) of the frames is postponed to the next section discussing forced update. Perhaps the most important notion is that in this worst case only 49% of the bit amount is controllable through quantization (excluding first frame). The corresponding figures for rates 16 and 32 kbits/s are 69% and 80%, respectively.

## 4.3 Effects of forced update

As mentioned earlier, the motivation for forced update stems from the freedom allowed in specifying the inverse DCT transform. To avoid undesirable error accumulation due to computational mismatches the macroblock should be forcibly coded in INTRA mode from time to time in order to reset the amount of error. The H.261 specifies that a macroblock should be forcibly updated at least once per every 132 times it is transmitted. As the transform specifications are taken directly from this standard, there is no reason not to follow this specification.

H.261 does not specify anything else, so that the exact specifications are left to the implementator. In this case the following strategy is adopted: a counter array is kept to count successive transmissions of each macroblock. Letting $N(i,j)$ denote the current count of transmissions of the $(i,j)$-th macroblock, the following procedure is then applied: if $N(i,j) < 30$ no measures are taken, if $30 \leq N(i,j) < 130$ forced update is used with a probability of 0.01, and if $N(i,j) = 130$ forced update is used with probability 1. The lower threshold was chosen to allow the bit stream to stabilize before putting additional stress on it (later when buffer regulation is discussed, the situation is that we have to code the first frame with some arbitrary quantizer step size which leads to a rather bad image quality. In frames immediately following the first one some amount of data is sent to increase the quality of image). The random approach was chosen so that certain macroblocks that are coded in almost all frames (blocks in the face area for an head-and shoulders sequence) will not be forcibly updated all at once. This way we have a 63% chance of using forced update before the count reaches 130. Note that in implementing this rule, a random independent, identically distributed (iid) source is required for each macroblock. The results are given in table 4.2.

Let us now turn to the results. Since the coder involves random functions, we performed five simulations and the results are shown in min/ave/max -form showing the minimum, average, and maximum values, respectively, of these simulations. The same parameters (8.33 Hz frame rate and a step size of 15) as in simulations without forced update were used. The comparison is made with the average values. The effect of forced update is manifested in the number of INTRA coded blocks now showing that an average of 0.21818 macroblocks/frame, compared to zero previously. Bit rate (8707.37 vs 9009.48, in order normal vs forced update) show an increase of 3.5%, which is not going to be a problem for our coder. We notice expected increases in luminance and chrominance bits and blocks (in INTRA mode all blocks are transferred), bits used to code mode data are essentially the same as before, and again an expected decrease in bits used to code motion vectors, as INTRA blocks are replacing INTER blocks. Also, by the properties of the frame coder the decrease in INTER4-coded blocks (9.15152 vs 8.95152) can be explained: the frame coder allocated these coding modes into areas of most complicated motion. With head-and-shoulders this is the face area, which also receives constant coding in successive frames. Hence these blocks are more prone to be forcibly updated.

Lastly we shall look into the pair of figures showing the average number of coefficients in a block (2.04303 vs 2.12224) and the average runlength of the last string of zeroes (51.8361 vs 51.9737). Recall the discussion on the coding modes in chapter 2. In these simulations we used a constant quantizer step size throughout the sequence, so we see that the (relatively) insignificant change in the latter pair of figures about the average runlegth of zeroes at the end (high-frequency corner) of the block is understandable from this viewpoint. Since the quantizer bit rates does not change, it will not allow for the transmission of higher-frequency coefficients unless the amount of these frequencies increases significantly. If we compare these numbers to the data obtained from coding the first frame (50.65), there is a more significant difference. We can combine this observation with the knowledge of the average number of coefficients (7.14 for the first frame) and the first pair of figures in this paragraph to get the obvious conclusion that prediction reduces the overall amplitude of most of the frequency components, even in the higher-frequency parts.

## 4.4 Does SAD relate to the amount of bits generated?

Bit rate regulation is considered in the next chapter, and here we want to use the data presented in this chapter to give some insights to this problem. Many buffer control schemes could benefit from an *a priori* estimate of the complicatedness or amount of motion present to anticipate some measures to be taken. Is there any simple way of obtaining such measures? SAD is used in conjunction with motion estimation so it seems that it would not require much extra work to compute the SAD of two successive frames to be coded. We can now determine whether the SAD between the previous frame and the current one is a good indicator on the number of bits the coder will generate for the current frame. It turns out that it is not, as can be seen in figure 4.5 where the SAD of a frame is plotted versus the number of bits that was the outcome. A reasonable correlation is evidenced, and this correlation becomes stronger as the quantizer step size is decreased. However, in controlling the buffer we need a figure that should be very reliable, and hence we would like to see a deterministic relationship between these two quantities. As it stands, we cannot utilize SAD in buffer control.

As a final note, it is observed that in the case of 8.33 Hz-8 kbits/s and constant quantizer step size, the maximum amount of bits produced for one frame was 1708 while the minimum amount of bits was 544, more than a threefold difference. As previously concluded the coder concentrates to send the new information due to motion in the frames. As expected, the amount of motion varies with time and the effect on the bit stream is very strong. This is a general observation in very low bit rate coding with motion compensation and waveform coding (the DCT is in fact a projection of the image signal on cosine waves of varying frequency) — the data rate varies considerably making the problem of regulating the data rate a hard one.

| ITEM | VALUE |
|---|---|
| FRAME 1 | |
| Number of total bits | 13386 |
| Number of luma blocks having more than DC coef | 140 |
| Number of chroma blocks having more than DC coef | 76 |
| Number of luma bits | 9472 |
| Number of chroma bits | 2382 |
| Number of mode + pattern bits | 752 |
| Average number of nonzero coefs | 7.14 |
| Average runlenght of zeros before EOB | 50.65 |
| FRAMES 2 - 34 | |
| Bitrate | 8707.37 |
| Average number of bits per frame | 1045.3 |
| Number of luma blocks | 33.1818 |
| Number of chroma blocks | 2.9697 |
| Number of luma bits | 493.061 |
| Number of chroma bits | 21.2424 |
| Number of mode + pattern bits | 137.394 |
| Number of vector bits | 273.606 |
| Average number of nonzero coefficients per block | 2.04303 |
| Average runlenght of zeros before EOB | 51.8361 |
| Number of inter-1 mode (macroblocks) | 14.5152 |
| Number of inter-4 mode (macroblocks) | 9.15152 |
| Number of intra mode (macroblocks) | 0 |

Table 4.1: Data for simulations without forced update. Parameters: framerate=8.33Hz, target bit rate = 8 kbits/s, quantizer step size = 30. Note that the Inter-1 mode statistics include also blocks with zero motion vector. "Mode + Pattern" represent bits spent for coding the coding mode and coded block pattern (CBP) at macroblock level), "Vector" gives the number of bits spent coding motion vectors. EOB is short for End Of Block.

| ITEM | MIN | AVE | MAX |
|---|---|---|---|
| FRAMES 102 - 134 | | | |
| Bit rate | 8925 | 9009.48 | 9078.19 |
| Average number of bits per frame | 1071.45 | 1081.7 | 1089.82 |
| Number of luma blocks | 33.303 | 33.903 | 34.6364 |
| Number of chroma blocks | 3 | 3.19394 | 3.39394 |
| Number of luma bits | 514.273 | 526.8362 | 531.727 |
| Number of chroma bits | 25.6667 | 28.1152 | 32 |
| Number of mode + pattern bits | 137.697 | 138.403 | 139.303 |
| Number of vector bits | 263.848 | 267.606 | 271.909 |
| Average number of nonzero coefficients per block | 2.04303 | 2.12224 | 2.16212 |
| Average runlenght of zeros before EOB | 51.8009 | 51.9737 | 52.2148 |
| Number of inter-1 mode (macroblocks) | 14.2424 | 14.6485 | 14.9697 |
| Number of inter-4 mode (macroblocks) | 8.75758 | 8.95152 | 9.12121 |
| Number of intra mode (macroblocks) | 0.151515 | 0.21818 | 0.30303 |

Table 4.2: Data for simulations with forced update. Parameters: frame rate=8.33Hz, quantizer step size = 30. Note that the Inter-1 mode statistics include also blocks with zero motion vector. The results are obtained by first coding 100 frames to load the counters after which data collection began. "Mode + Pattern" represent bits spent for coding the coding mode and coded block pattern (CBP) at macroblock level), "Vector" gives the number of bits spent coding motion vectors. EOB is short for End Of Block.

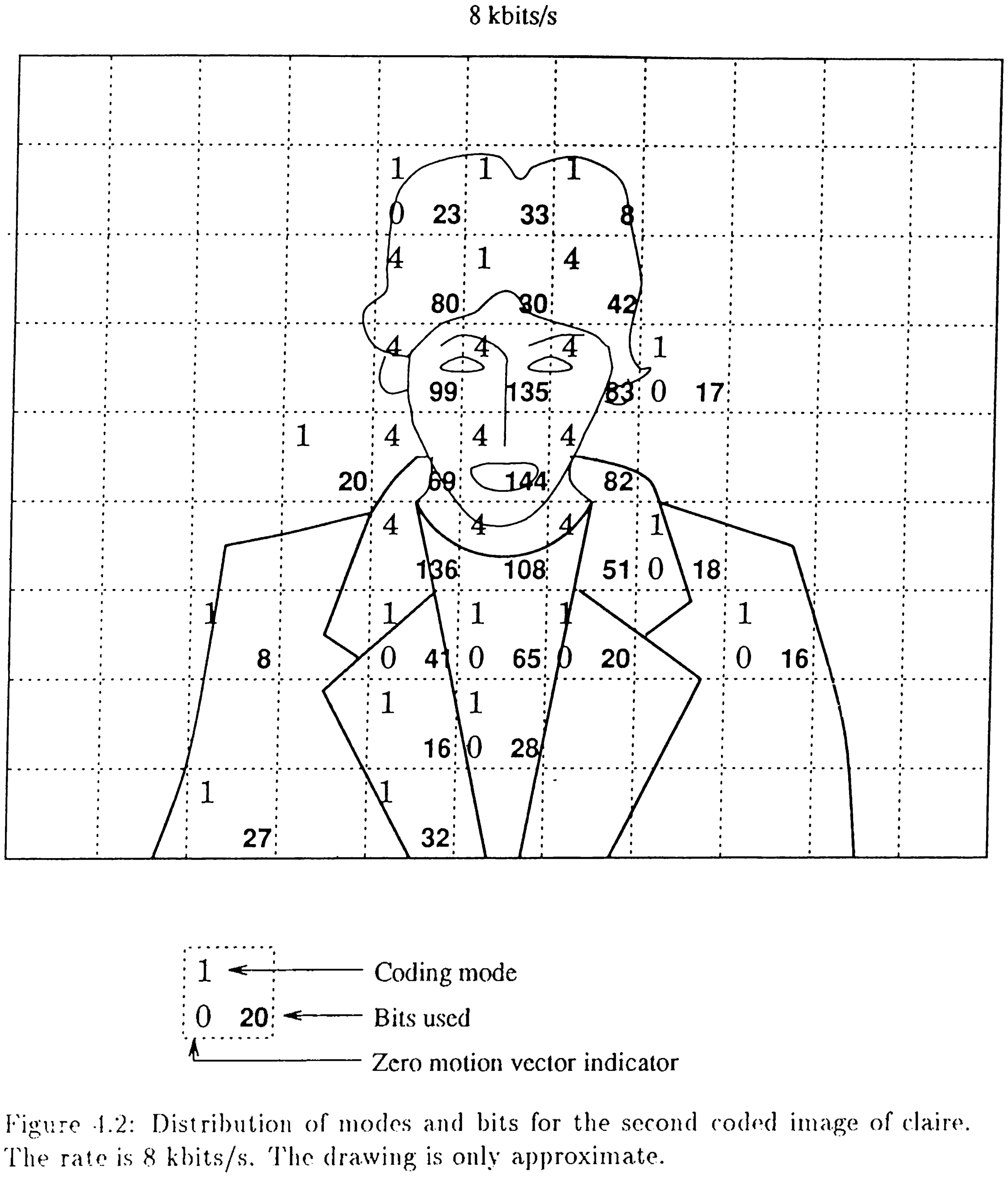


Figure 4.2: Distribution of modes and bits for the second coded image of claire. The rate is 8 kbits/s. The drawing is only approximate.

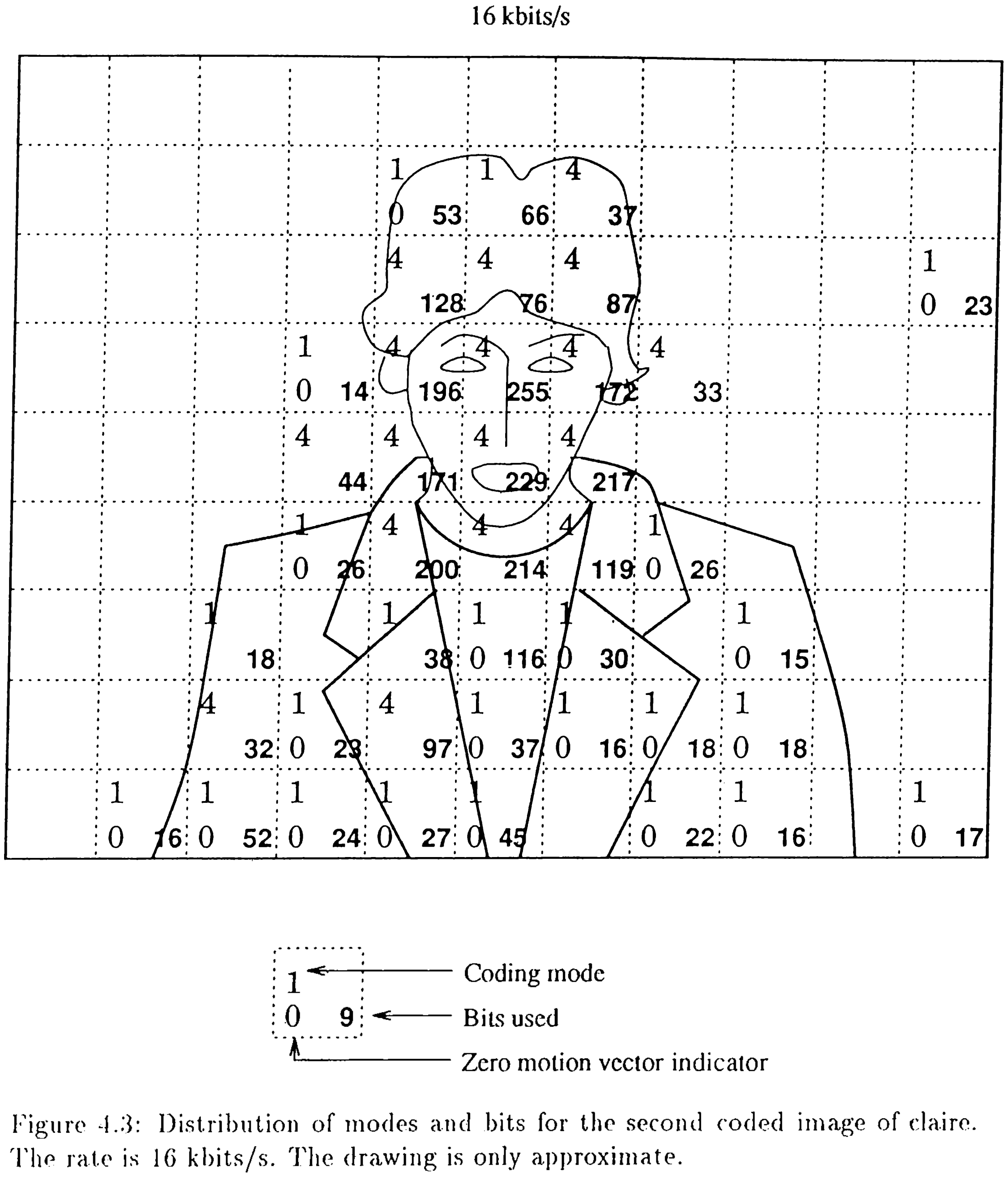
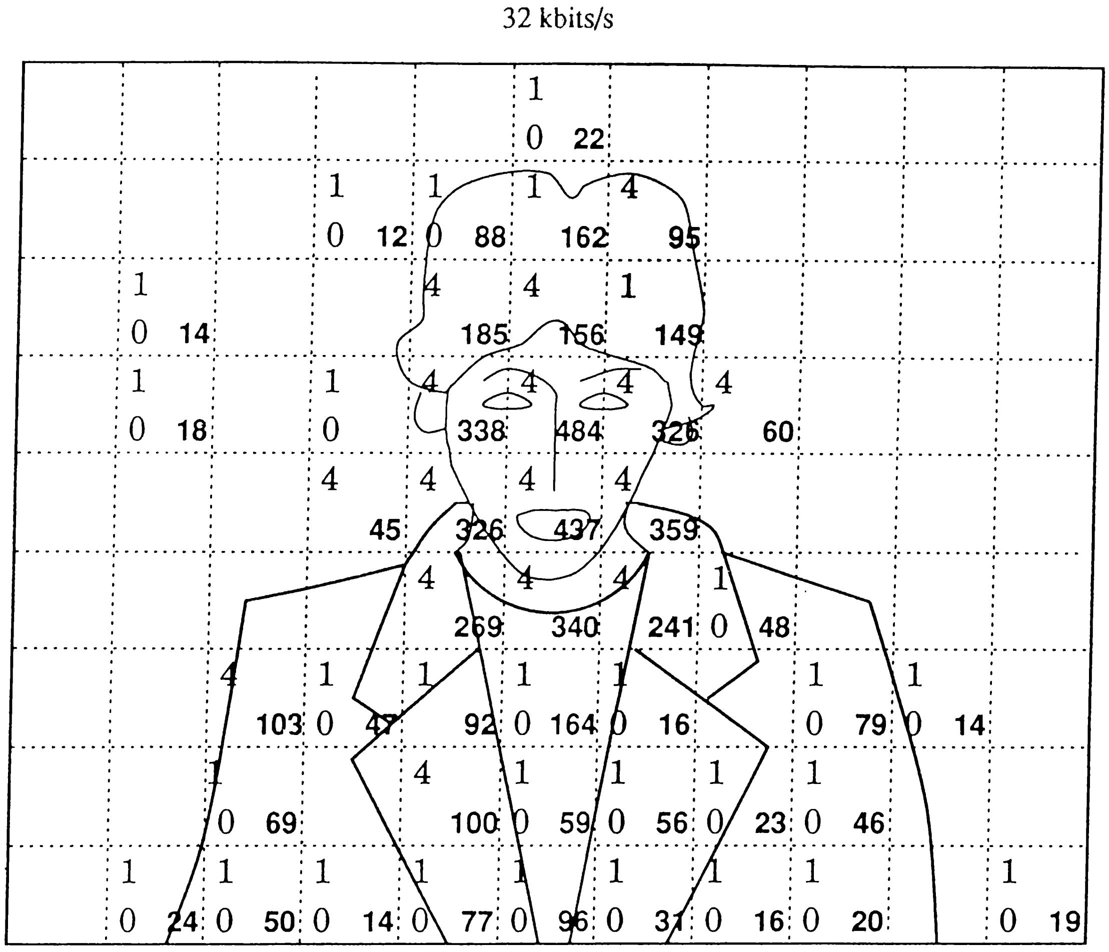


Figure 4.3: Distribution of modes and bits for the second coded image of claire. The rate is 16 kbits/s. The drawing is only approximate.



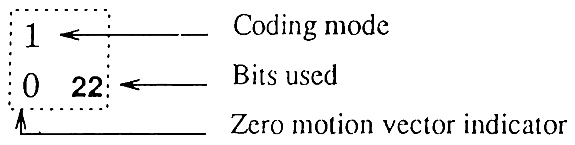


Figure 4.4: Distribution of modes and bits for the second coded image of claire. The rate is 32 kbits/s. The drawing is only approximate.

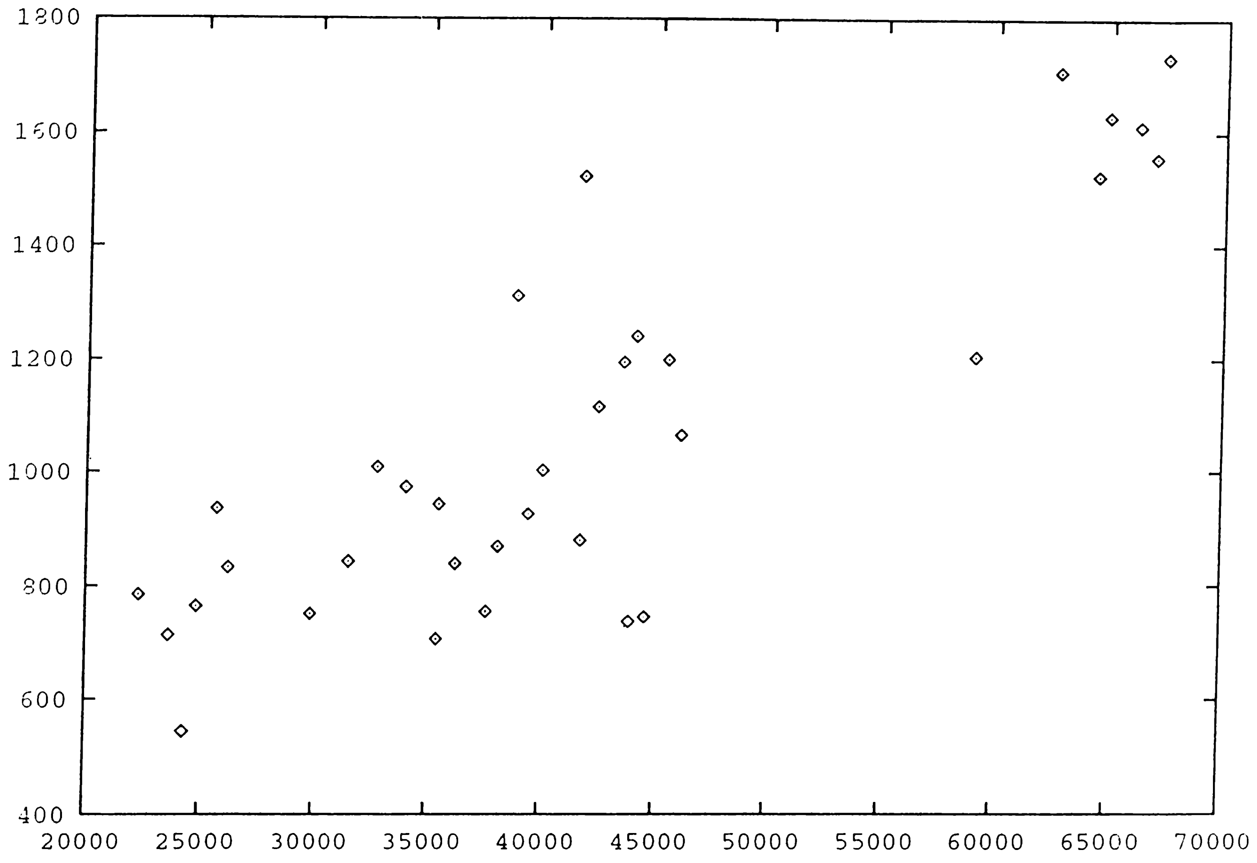


Figure 4.5: SAD versus bits generated for Claire using 8 kbits/s (step size=30) and 8.33 Hz

# Chapter 5

# Bit Rate Regulation

## 5.1 Introduction

In the previous chapter it became clear that in order to be efficient, the coder has to concentrate on sending just the new information between the successive frames. The amount of this new information can vary considerably, but still we have a limited channel capacity at our disposal. Bit rate regulation is about converting the varying bit stream of the frame coder into a smooth constant data stream going into the channel. There is another constraint, too. In order to synchronize the image data and the audio data there should not be to high delay between the two signals. Generally it is agreed upon that this delay should not exceed 0.5 seconds, and usually the *buffer* which stores the bits created by the coder before they are sent to the channel is sized accordingly: the buffer is big enough to store bits worth 0.5 s of transmission time depending on the rate used. Sometimes it happens that the coder fails to control the bit rate adequately, and tries to feed too many bits to the buffer. When the buffer is full it cannot receive any more bits, so these bits are inevitably lost, a condition that is called *buffer overflow*. However, coders that are designed to avoid buffer overflows altogether tend to be too pessimistic and not to use the capacity of the channel optimally. Hence, usually a limited number of overflows is tolerated.

The buffer control problem is especially harsh in very low bit rate coding, where the bit stream is already optimized in a way — it is supposed to be very low bit rate. It turns out that most normal buffer control methods intended for fixed-frame rate coders fail to keep the frequency of buffer overflows low (this claim is demonstrated in the next section). It is, therefore, sensible to allow the frame rate to vary so that the transmission of a frame might take as much time as required. In SIM, which is based on 25 Hz base frame frequency this means that the frame transmission time may take values from 3/25 – 12/25 seconds (the upper limit is chosen so that the buffer will not overflow in normal operation).

Let us look a bit closely how quantizing works. At the transmitter a *forward quantizer* divides the range of possible input values into nonoverlapping ranges whose size is equal to a given step size. These ranges are called *input cells* and the boundaries between the cells are called *decision levels*. Within a cell one value, usually the midpoint, is chosen to be the *representative level* of that cell. The cells are indexed such that the centermost cell, which has a representative level at zero, is given the index value zero. Values at the positive input side are given positive indices and those cells at the negative input range get negative index values. Given a value, the forward quantizer checks the cell it falls into and sends out the index of that cell. At the receiver (and at the transmitter, which contains a receiver) the *inverse quantizer* receives a cell index and outputs the representative value of that cell.

A *quantizer index* in this chapter means the index identifying a quantizer among many quantizers. In H.261 and SIMs 32 different quantizers are used, they are indexed with 1 – 31. The step sizes of these quantizers are twice their index value. All control of the quantizers attempt to produce the correct index of the quantizer, not the step size. In order to avoid confusion with the output of the quantizer to be transmitted which is also called an index (to the reproduction values of the cells) it is henceforth assumed that "controlling the quantizer" means controlling the quantizer's index, and when we are saying "the quantizer" we mean a quantizer with some index $i$.

## 5.2 Buffer control in fixed-frequency systems

The ancestor of SIM models, H.261 does not specify any special method for buffer control but rather provides support through the bit stream protocol for any possible method. Even in the SIM models it would not be necessary to specify any special method, but in order to demonstrate a working coder (and hence that the claimed bit rates can be achieved) a reference method needs to specified. Further, such a method helps us to find out to which extent the bit stream syntax must support any given buffer control method. For example, we could send only one quantizer index per frame and thus simplify the structure (and save some bits) of the bit stream, but we must now whether sending just one quantizer step size is enough to achieve the given data rates.

In fixed-frequency systems the only way to control output rate is to adjust the quantizer step size. Also, the buffer size has to be set so that the given frame frequency can be accomplished, which is much harder constraint than the 0.5 s discussed above. For example, if we have a coder operating on 25 Hz sequences (one of the base sequences used by COST, the other is 30 Hz), we must fit 12 frames in any half-second interval. Usually these schemes try to distribute the given bit quota evenly over the frame and, for simplicity, try to produce at most a given quota of bits per frame to maintain constant picture frequency. For example, a H.261 coder operating at 64 kbits/s might produce only 64000/25 = 2560 bits per frame. A simple such method, taken from the original reference model used in the development of H.261, is

$$Q(i,j) = \overline{Q(i,j-1) + K(B(i,j-1) - \bar{B})} \tag{5.1}$$

where $i, j$ are the indexes of current macroblock row and column, respectively, $Q$ denotes the quantizer index and $B$ is the amount of bits produced at a given macroblock. $\bar{B}$ is the mean number of bits available per macroblock and $K$ is a proportionality constant. The bar sign above the equation means that the value achieved after computing the formula is rounded to the nearest integer value. This method, to be stable, requires that the new quantizer step size should be updated after every macroblock (a new value of a quantizer cannot be used until the receiver also knows about it). However, this can lead to considerable amount of channel capacity being devoted to handling quantizer information. Therefore, the quantizer index is usually checked fewer times and in the COST reference model it is done at the beginning of each macroblock line. It should be noted that the previously described drawback is present in the reference models used to study H.261, and the proposed constraint of fewer checks on the quantizer index requires deeper study of buffer regulation methods to ensure that the constraint is realistic.

A slightly more complicated method of controlling the buffer step size is

$$Q_k(i) = \bar{Q}_{k-1} + K\frac{B_{k-1} - \hat{\bar{B}}}{\hat{B}} + L\frac{B_{k,mb} - \frac{mb}{99}\bar{B}}{R} \tag{5.2}$$

where $Q_k(i)$ is the quantizer for macroblock line $i$ and frame $k$, $\bar{Q}_{k-1}$ is the average quantizer for the previous frame, $K, L$ proportionality constants, $B_{k-1}$ the number of bits used to code the previous frame, $\hat{B}$ is a target number of bits for the current frame, $B_{k,mb}$ the number of bits used in the current frame up to the current macroblock $mb$, and $R$ is the bit rate in use. The macroblock number is counted from left to right starting from the upper left corner using integers 1 - 99. The constants $K$ and $L$ are given by COST as 16 and 300, respectively. This method is used as a reference method by COST and it is presented here in a simplified light for comparison purposes (here we are not assuming variable frame rate as in the complete method). For the case where frame rate is 8.33 Hz and bit rate 8 kbits/s, the target bit rate is 960 bits. It is seen that the first term invokes the "complicatedness" of the previous frame, second records the leftover capacity after the previous frame and the third effectively does the same job as the simple control scheme presented earlier. Also, the first two terms are constants for each individual frame, so it is seen that this is just a modified method of the first one.

If we try to code the sequence with the first method (eq. 5.1) given above, using Claire (remember the remarks on the nature of this sequence) and with checks on the buffer at the beginning of each macroblock line, we find that the results are not very encouraging. Specifically, using a fixed frame rate with parameters 8 kbits/s - 8.33 Hz the simulations show that in 33 coded frames there were 12 overflows which, of course, is unacceptable. With respect to the second method presented above (5.2), we noted that the nature of this method is essentially the same as the one tried out here, except that we can modify the behavior of this method by adjusting the two propotionality constraints. However, as these results suggest, we would have to severely restrict the effect of the third term in equation 5.2, which would force this system to use a nearly constant quantizer step size over the whole frame. This constant step size must be chosen carefully which may lead to a situation where we do not utilize the channel capacity to a full extent. Better results can be more easily achieved by allowing the transmission time for a given frame to vary depending on its contents: a more complicated difference frame is allowed to take more transmission time than an easy one.

Allowing variable frame rate does not lead to complicated control schemes but it in fact simplifies the task of bit rate regulation. Instead of trying to fit all frames in a fixed bit quota, we can now use any number of bits we like as long as it does not exceed 0.5 s worth of transmission time. This gives considerably more freedom in buffer control.

## 5.3 The COST reference model

The COST reference models buffer control mechanism is just a fixed-frame rate type control plus the added help of variable frame rate. The equation 5.2 is used with the following changes of interpretation:

- for each frame an individual target frame rate and thus target number of bits is computed before coding that frame
- the second term now reflects the differences between target frame rates of consecutive frames.

The target bit rate is determined as:

$$f_{\text{target}} = 10 - \frac{\bar{Q}_{k-1}}{4} \qquad 4 < f_{\text{target}} < 10 \tag{5.3}$$

where $f_{\text{target}}$ is the new target frame rate: after which the target bit rate can be determined as:

$$\bar{B} = \frac{R}{f_{\text{target}}} \tag{5.4}$$

The contents of the buffer is checked at the beginning of each macroblock line.

To save bits, it is advantageous to send just the offset of the new quantizer index from the current one. In the case of SIM, there are two bits reserved for this purpose coding values {-2,-1,1,2} (in case of zero offset the quantizer information is simply not sent: the information telling whether the quantizer index is present or not is coded together with the mode information). Further, experimenting with this kind of buffer control shows that these offsets are not sufficient for controlling the bit stream (remember that the contents of the buffer is checked at the beginning of each macroblock line). Therefore a scaling constant $N$ is chosen so that the offset represents values {-2N,-N,N,2N}. The first problem is to determine this constant $N$.

Looking at the sequences of quantizers, it can be seen that with $N = 1$ the steps are nearly always taken as either -2 or 2, but the values do not vary very much. On one hand, there were one buffer overflow with Claire, indicating that (since Claire is a relatively easy to code) the response is, indeed, too slow. With $N = 2$ there were no overflow problems but the oscillation between values grew stronger. Simulations with $N = 3$ showed that there is a trade-off between the range of quantizers used on the other and the stability of the bit stream on the other (with $N = 3$ step sizes 6 and 60 were used within the same frame in the worst case). The value $N = 2$ was chosen as a compromise. With this value two typical quantizer sequences was chosen from simulations with Claire with 8 kbits/s datarate: 13-9-5-9-13-17-17-21-19, 11-7-3-7-11-15-19-23-27. Each value correspond to a line of macroblocks, where the check on the index is made.

So, for a head-and-shoulders sequence it can be seen that when the coder is coding stationary background the quantizer step size gets very low until the scan hits the first row with motion (see the silhouette drawings at the end of the previous chapter). Sometimes the first two rows of motion will contribute as much as half of the overall bit amount for that frame. With a head-and-shoulder sequence this behavior has its advantages, since it is likely that most of the bits are used in the lines containing most of the face, a behavior clearly seeked with this method.

The common feature among all these simulations is a drastic reduction of frame rate (and thus also a drastic increase in image quality). The frame rate in the case $N = 2$ is in average 3.30 Hz corresponding to an average of 2423 bits per frame (when the bit rate is 8 kbits/s) which can be compared to that of H.261's 2560 bits per frame for 64 kbits/s transmission.

## 5.4 Proposed buffer control mechanisms

Although it can be seen that the buffer control mechanism of COST usually manages to use the lowest step size around the face area, it can be done in a more controlled way by adjusting the quantizer step size by looking at the quantizing problem on a framewise basis. We get the added benefit that we do not have to adjust the quantizer step size in the middle of the frame thus saving some bits. A way to achieve this is explained later.

Framewise control of quantizer step size requires some data of the frame as a whole. Also we may try to adjust the quantizer in *a priori* or *a posteriori* basis. As the previous chapter more or less ruled out any simple way of measuring the complexity of a frame beforehand, *a posteriori* approach has a bullet-proof figure for doing this: the amount of bits produced per frame. We are coding the picture into actual bits to be transmitted and use it as an estimate of the complexity of the image.

In order to utilize this approach, we should first choose an initial quantizer and see how many bits we create using it, then decide whether we need to make any adjustments to the bit amount. To achieve a good image quality, it would be advisable to optimistically choose a rather low value for the quantizer index and reduce the amount of bits if needed. With this approach, we expect to use the bit reduction technique frequently. (The common way to reduce bits is to increase the quantizer index, but we have implemented another method and therefore speak generally about "reducing the bit amount" instead of "increasing the quantizer index". The details are later in this section.)

We choose an initial quatizer for the frame based on the previous frame. There is a problem: is the amount of motion in a sequence smooth enough to justify the selection of the initial quantizer based on the previous frame? Based on our experiments where we monitored the amount of bits produced per frame with a fixed step size, we noted an unfortunate feature: usually the motion is smooth, but there are abrupt changes which are significant in magnitude. Let us deal with these two cases separately. If the motion is smooth, the arguments for using the coding results of the previous frame are valid. In the case of large, abrupt changes the situation is more complicated. If the change is from high motion content to low, the principle of choosing the initial step size optimistically will protect the coder from coding a frame too harshly. But there clearly is a problem if the change is of opposite direction, that is from low motion to high. To overcome it, we chose a two-stage bit reduction scheme: first (which we expect to use frequently) is "gentle" in nature and the second is very harsh intended to cope with situations where large savings are required. To implement this method, there are two ways: one could look at the amount of savings required and then choose between these methods, or simply try the gentle method first, and if the amount of savings is not sufficient, use the second method. Here the latter was chosen.

Now we turn to the bit reduction techniques. We are given the unquantized transform coefficients, and as noted the rate/distortion coder can freely process them as it wishes since the processing done here affects both ends of the transmission channel and data integrity is not violated. So, the R/D coder faces two alternatives: it may choose to use different quantizers in different parts of the image, or different quantizers within a macroblock (or both). In this latter case the idea is to "mutate" the data into a form that is more efficient to code. The receiver uses the remapping given by the original quantizer, and this poses no problem as previously stated because of the previous remark on data integrity.

To see how this is done, consider a macroblock quantized with some initial quantizer. In this case the macroblock contains data to be transmitted, each coefficient being replaced by the quantizer cell index of that coefficient. Assume that we need to decrease the amount of bits generated from that macroblock. The only reliable way to gain savings is to increase the runlengths of zeros in the blocks. We could accomplish this by increasing the quantizer sufficiently. But we may also take a shortcut by looking at the already once quantized values. Those indexes having a value close to zero (like 1, -1, etc) are the most likely to become zero after increasing the step size. So we might as well set them to zero directly and reconstruct the remaining values with the original quantizer, thus achieving better resolution for the reconstructed values.

The harshness of this operation can be adjusted according to where we are within a picture. Here two methods will be tried out and compared with each other. First we may make an assumption that the most important parts of the image are likely to be in the middle of it, typical for head-and-shoulders sequences. Let us call this method P1. Another method is based on the knowledge of the behavior of the frame coder: we know that the frame coder is good at putting the motion vectors into blocks which are in the most important area (see figures 4.2, 4.3, and 4.4). So we just check whether the macroblock has a non-zero motion vector(s), and define these to be blocks that should be treated more gently than others. This method will be referred to as P2. The reason for choosing another similar method to P1 is that P2 does not make an assumption about the type of the image sequence. It just assumes that those parts of the image that have a motion vector also have a higher information value. It should be noted that in order to be truly "assumptionless" in any other sense, we should redefine the basic principle of P2: code those blocks which have a motion different from the *trend* of motions more carefully. For comparison purposes using head-and-shoulders, the formulation chosen here will suffice.

Formally, this buffer control mechanism is expressed as follows: let `HI` be those macroblocks considered to be in the high quality area, and `LO` those which are not. Let `MB` denote a macroblock. Algorithmically, the algorithm is given in table 5.4.

Note that "too many bits" refers to the case when the number of bits exceeds bit rate/2, 0.5 seconds worth of transmission. There are two stages, first one tries to do a civilized job, the second is to make sure no overflows occur. In practice, this algorithm can be implemented in one stage as we may first code the `LO` parts of the image, then check whether enough savings were gained, if not, proceed to process the `HI` parts of the image while blocking the transmission of the transform coefficients of the `LO` parts. Note that in the normal mode of operation we want to

```
code using initial quantizer
if too many bits then
      set all indices $I$ satisfying $|I| \leq 1$ to zero if MB $\in$ LO
      is still too many bits
              set all indices to zero if MB $\in$ LO
              set all indices $|I| \leq 1$ to zero if MB $\in$ HI
      end if
end if
```

Table 5.1: Algorithmical representation of the proposed buffer control algorithm.

see the coder proceeding frequently to the first stage of bit reduction. This way we can keep an optimistic value for the initial quantizer and thus achieve good image quality.

One unresolved point is adjusting the initial quantizer step size. This is done by adjusting the quantizer based on the amount of bits produced in the previous frame. By the previous paragraph we should use the amount of bits after the first stage of reduction, as this is considered a normal case of operation. If we do not have to use the first stage, we check the the amount of bits produced only against the lower limit, otherwise nothing needs to be done. and if we have to resort to the second stage of reduction, we still use the number of bits after the first stage. Let us define $B_0$ the amount of bits produced with the initial quantizer, $B_1$ the amount of bits after the first stage and $B_2$ after the second. The following formula gives the new quantizer step size:

$$Q_{\mathrm{new}} = \begin{cases} Q_{\mathrm{old}} - 2 & \text{if } B_1 < \frac{4}{25}R \\ Q_{\mathrm{old}} + 1 & \text{if } \frac{7}{25}R < B_1 < \frac{10}{25}R \\ Q_{\mathrm{old}} + 2 & \text{if } B_1 \geq \frac{10}{25}R \\ Q_{\mathrm{old}} & \text{otherwise} \end{cases}$$

where $R$ is the bit rate in use, $Q_{\mathrm{old}}$ is the initial quantizer index used to code the current frame, and $Q_{\mathrm{new}}$ is the initial quantizer for the next frame. This formula is used whenever any bit rate reduction is performed. In case there is no bit rate reduction, this formula is used:

$$Q_{\text{new}} = \begin{cases} Q_{\text{old}} - 2 & \text{if } B_0 < \frac{4}{25}R \\ Q_{\text{old}} & \text{otherwise} \end{cases}$$

With this setup, the frame rates are approximately the same as in the comparative COST method.

## 5.5 Results

Sequences were compared to each other. The most severe subjective disturbance common to all buffer control methods were found on the neck of Claire. There is practically no visible difference between the two proposed methods, and compared to these the SIM comparative method appeared "nervous" with more fragmentation and more fake activity in the sequence. This is probably due to the strong variation of the quantizer step size. A notably bad effect was encountered in SIM when the person in the sequence nods downwards. In this case her mouth (which contains the most complicated motion, the general motion of the face combined with the moving of the mouth as the person speaks) gets to the lower macroblock lines in which higher step sizes are used (see Appendix). The reader is referred to the appendix for selected frames from different sequences.

Comparing these three methods it can be argued that the assumption behind the P1 method may sound artificial, but as it has turned out there are as strong assumptions behind the COST method. Therefore, P2 should be preferred among these as it is more flexible than the others.

# Chapter 6

# Conclusions

This thesis describes work done as a part of a multinational effort to create an international recommendation for very low bit rate coding. The work was done within an international working group and it had to be co-ordinated with the work of other bodies, so specific problems were asserted within a complex system. Of course, this could not have been done without a global understanding of how the coder works as a whole. Specifically, three problems were under study in this thesis: first, to determine a proper image size for transmission, second, to understand the statistical behavior of the image sequence coder, and thirdly, to develop and evaluate a buffer control method.

In the first problem, we had to select an image size to be used in communicating the image from the coder to the decoder, using the assumption that the image data was delivered to us in CIF format. This size is considered to be too large for the low bit rates required, so two smaller image sizes derived from it were under consideration. These were QCIF and NCIF, of which QCIF is a fourth of the CIF size, and NCIF is approximately a ninth of CIF. The following factors affected our choice: the decimation and interpolation filters used, and the rate/distortion mechanism of the coder. In short, it was important to find out which was more harmful to the image: filtering the data out or to let the coder quantize out the excess data. However, this simplification was found out not to be valid in practice because of the strong performance constraints facing the conversion filters. These constraints affect especially the NCIF filter, whose quality is so poor that the information content of the resulting image is so low that the more gentle coding cannot gain the losses of the filtering. This was verified with a constraint-free filter design. Under the circumstances QCIF is preferable.

The behavior of the coder stripped from buffer control was studied. This served to give valuable information on how the coder actually performs its task. The most important observations during coding of a typical head-and-shoulder sequences were:

- The coder never chooses INTRA mode in normal operation.
- Of the two INTER-modes, the 4-vector mode is used where complicated motion is encountered, i.e. the face area, and the 1-vector mode where larger objects are in motion, for example the shoulder area. Also, the 1-vector mode is used in places where the motion affects only a small part of the corresponding macroblock in which case the motion vector is set to zero.
- Motion estimation at half-pixel accuracy is clearly desirable.
- The amount of side information (motion vectors and mode data) comprises about 50% of the data stream at the lowest (8 kbits/s) bit rate. This amount gets less at higher rates.
- The sum of absolute differences correlates with the bits produced per picture, but this correlation is not sufficient to allow for control actions to be based on it.

Lastly, buffer control was studied. The motivation for controlling the bit stream lies in the fact that the coder under consideration bases its prediction in motion estimation-compensation cycles. Because the coder uses exclusively the INTER-mode for coding the amount of bits depends on the amount of motion in the sequence, which of course varies considerably. The variation in bit stream is especially strong for very low bit rate coders. Therefore, regulation techniques built for standard fixed-frequency applications fail to perform adequately. This was demonstrated using a buffer control method adopted from the simulation model used in developing the H.261 which resulted in frequent overflows. Hence, the motivation for using a variable frame rate was demonstrated.

Two task were undertaken at this point: first, a reference method delivered by the working group was evaluated and an optimal value for an undetermined parameter was found. Then, an alternative method was constructed for bit rate regulation. The reference method was found out to be a straightforward extension to some buffer control methods used for fixed frame-rate systems. This method attempted to distribute bits evenly throughout the picture. Since the coder utilized a standard left-to-right, up-down raster scanning of the macroblocs, it was found out to lower the quantizer step size to a low value so that the coder consumed most of the bits in the face area, using a high step size for the remaining parts. The step size varied strongly within a typical frame, and this led to some artifacts in the image, specially when motion occurred in the lower half of the image.

In the alternative design a more global approach was taken, and two similar algorithms were tried. Bits were assigned on a framewise basis, attempting to allocate more bits to the most important parts of the image. The two methods differ in the way they try to determine those parts of the image that deserve more bits. The first one simply assumes that the center part of the image is more important than the boundary parts, and the second bases its decision on the properties of the image coder: since the image coder assigns motion vectors very efficiently to those regions which are considered to have more value than others, the second algorithm favors blocks with motion vectors. The methods were compared to each other using head-and-shoulder sequence. A common observation concerning all methods was that the picture frequency had to be lowered considerably to gain a significant improvement in image quality. An average of 3 Hz was encountered. The two alternatives were found to be better than the reference method, since they performed more uniformly than the reference method.

In this thesis some current research trends within video signal coding were pointed out. With respect to the coder type addressed in this thesis, the most important part is the search for better motion estimation methods. The logic is that with a better estimation method a better prediction can be made and there are less data to transmit. However, since the current coder estimates motion vectors with a fairly large blocksize by minimizing the absolute error, better methods would eventually bring about a finer grid of motion vectors. There are two problems with this logic. First, more motion vectors are always also more motion vectors to transmit (recall that the amount of side information already comprises 50 % of the data already). Second, the better our prediction is the more uncorrelated the error signal becomes and the worse the relative gain gotten by using DCT.

Therefore, it seems that using this structure for coding there is no way of getting significant improvement in the image quality. As a sign of this, there seems to be an unofficial concencus that the work done in the upcoming MPEG4 standardization effort will concentrate on other methods of coding.

# Appendix A

# Bit-tables used in coding

This appendix contains the bitstream definitions as supplied by the COST group. These tables were used to compute the bits used in simulated transmissions.

| | |
|---|---|
| | 11 |
| | 1011 |
| | 1010 |
| | 1001 |
| | 1000 |
| | 0111 |
| | 0000 11 |
| | 0010 1 |
| | 0110 |
| | 0000 10 |
| | 0101 |
| | 0010 0 |
| | 0100 |
| | 0001 1 |
| | 0001 0 |
| | 0011 |
| | 0000 01 |

Table A.1: Codes for the the coded block patern, luminance component

| Chroma pattern | Mode | Code |
|---|---|---|
| 0-0 | INTRA | 0001 11 |
| 1-0 | INTRA | 0000 0011 |
| 0-1 | INTRA | 0000 0010 |
| 1-1 | INTRA | 0000 011 |
| 0-0 | INTER-1 | 1 |
| 1-0 | INTER-1 | 0101 |
| 0-1 | INTER-1 | 0100 |
| 1-1 | INTER-1 | 0001 10 |
| 0-0 | INTER-4 | 011 |
| 1-0 | INTER-4 | 0010 1 |
| 0-1 | INTER-4 | 0010 0 |
| 1-1 | INTER-4 | 0001 01 |
| 0-0 | INTER-1/Q | 0000 010 |
| 1-0 | INTER-1/Q | 0000 0001 1 |
| 0-1 | INTER-1/Q | 0000 0001 0 |
| 1-1 | INTER-1/Q | 0000 0000 1 |
| 0-0 | INTER-4/Q | 0011 |
| 1-0 | INTER-4/Q | 0001 00 |
| 0-1 | INTER-4/Q | 0000 11 |
| 1-1 | INTER-4/Q | 0000 10 |

Table A.2: Table of codes for the combined pattern-prediction mode data. INTER-modes are duplicated so that they can carry the quantizer modification flag.

| Run | Level | Code |
|---|---|---|
| 0 | 1 | 11s |
| 0 | 2 | 0100s |
| 0 | 3 | 00101s |
| 0 | 4 | 0000110s |
| 0 | 5 | 00100110s |
| 0 | 6 | 00100001s |
| 0 | 7 | 0000001010s |
| 0 | 8 | 000000011101s |
| 0 | 9 | 000000011000s |
| 0 | 10 | 000000010011s |
| 0 | 11 | 000000010000s |
| 0 | 12 | 0000000011010s |
| 0 | 13 | 0000000011001s |
| 0 | 14 | 0000000011000s |
| 0 | 15 | 0000000010111s |
| 1 | 1 | 011s |
| 1 | 2 | 000110s |
| 1 | 3 | 00100101s |
| 1 | 4 | 0000001100s |
| 1 | 5 | 000000011011s |
| 1 | 6 | 0000000010110s |
| 1 | 7 | 0000000010101s |
| 2 | 1 | 0101s |
| 2 | 2 | 0000100s |
| 2 | 3 | 0000001011s |
| 2 | 4 | 000000010100s |
| 2 | 5 | 0000000010100s |
| 3 | 1 | 00111s |
| 3 | 2 | 00100100s |
| 3 | 3 | 000000011100s |
| 3 | 4 | 0000000010011s |

| Run | Level | Code |
|---|---|---|
| 4 | 1 | 00110s |
| 4 | 2 | 0000001111s |
| 4 | 3 | 000000010010s |
| 5 | 1 | 000111s |
| 5 | 2 | 0000001001s |
| 5 | 3 | 0000000010010s |
| 6 | 1 | 000101s |
| 6 | 2 | 000000011110s |
| 7 | 1 | 000100s |
| 7 | 2 | 000000010101s |
| 8 | 1 | 0000111s |
| 8 | 2 | 000000010001s |
| 9 | 1 | 0000101s |
| 9 | 2 | 0000000010001s |
| 10 | 1 | 00100111s |
| 10 | 2 | 0000000010000s |
| 11 | 1 | 00100011s |
| 12 | 1 | 00100010s |
| 13 | 1 | 00100000s |
| 14 | 1 | 0000001110s |
| 15 | 1 | 0000001101s |
| 16 | 1 | 0000001000s |
| 17 | 1 | 000000011111s |
| 18 | 1 | 000000011010s |
| 19 | 1 | 000000011001s |
| 20 | 1 | 000000010111s |
| 21 | 1 | 000000010110s |
| 22 | 1 | 0000000011111s |
| 23 | 1 | 0000000011110s |
| 24 | 1 | 0000000011101s |
| 25 | 1 | 0000000011100s |
| 26 | 1 | 0000000011011s |

Table A.3: Runlenght-amplitude codes for certain combinations of runs of zeros and the following quantized transform coefficient. The last 's' stands for sign: '0' positive,'1' negative. Missing entries are coded as a combination of an escape code (0000 01) to escape from the table, after which the runlenght value is coded using 6 bits and the following amplitude gets 8 bits.

# Appendix B

# Bitstream syntax

Below there are a series of figures illustrating the layers of the bitstream. In the figures, sharp boxes indicate fixed length codes, and rounded boxes variable length codes. Figure B depicts the highest level, data sent once per frame. Figure B shows data sent per each macroblock, and finally figure B shows transmitted data at block level.

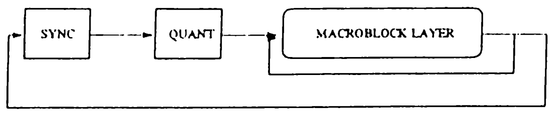


Figure B.1: Picture layer of the bitstream.

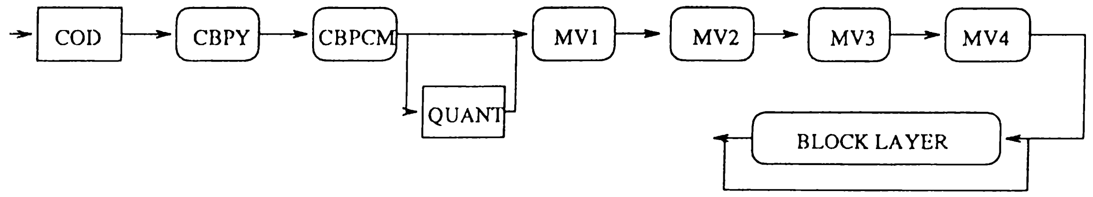


Figure B.2: Macroblock layer of the bitstream

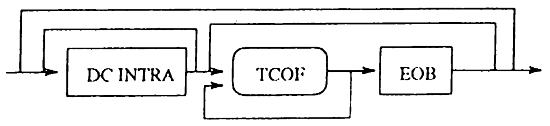


Figure B.3: Block layer of the bitstream.

# Appendix C

# Pictures

This appendix comprises of two sets of images. The first set comprises of filtering results using the filters defined in chapter 3. The second set is for comparing buffer control methods defined in chapter 5.

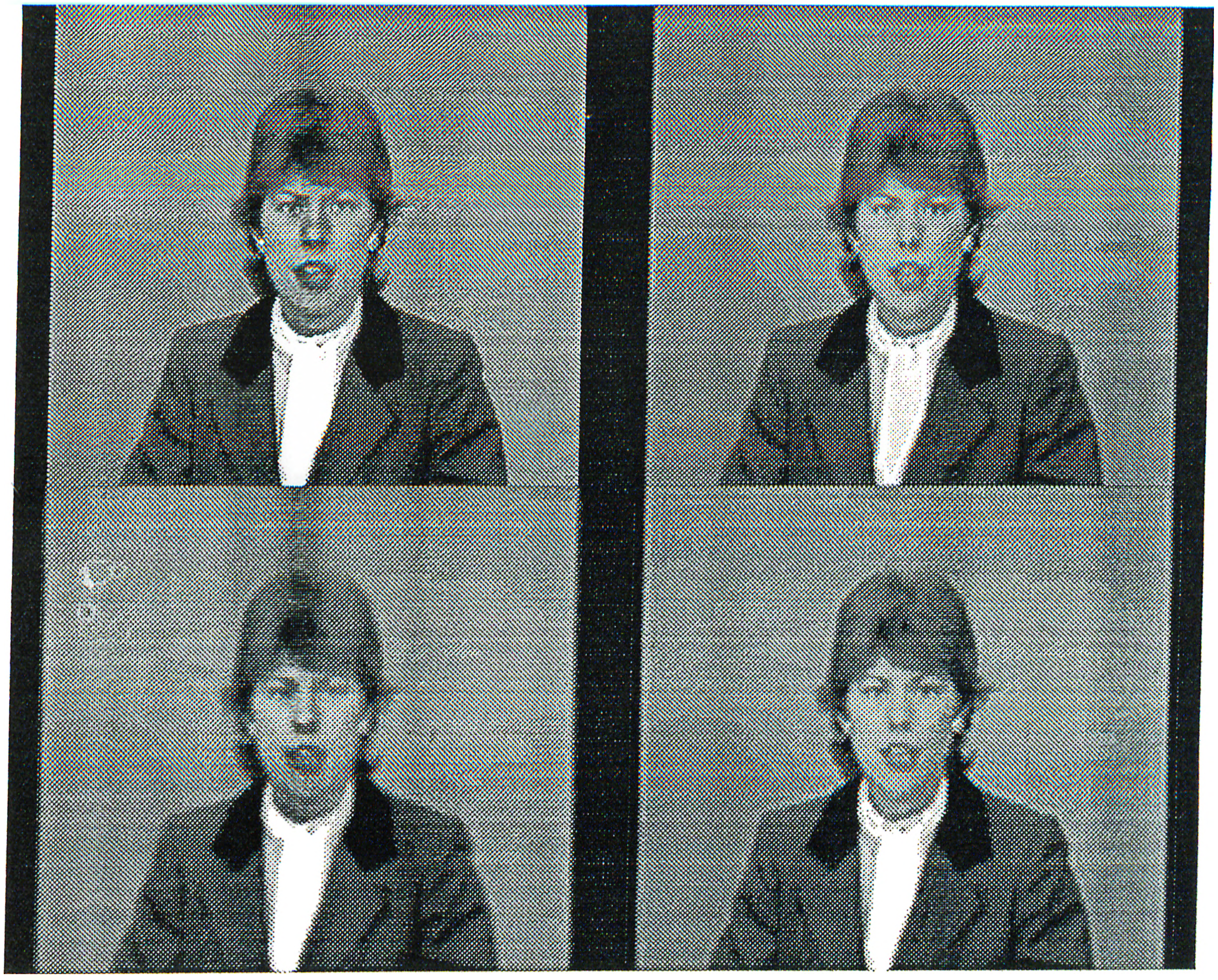


Figure C.1: Filtering results using the decimation filters. In clockwise order starting from upper left corner: a) original image, the first image in the sequence Claire, b) QCIF/COST, c) NCIF/COST, and d) the unconstrained NCIF design using a $37 \times 37$ window, a transition band of $0.17\pi$ and an attennuation of 50 dB.

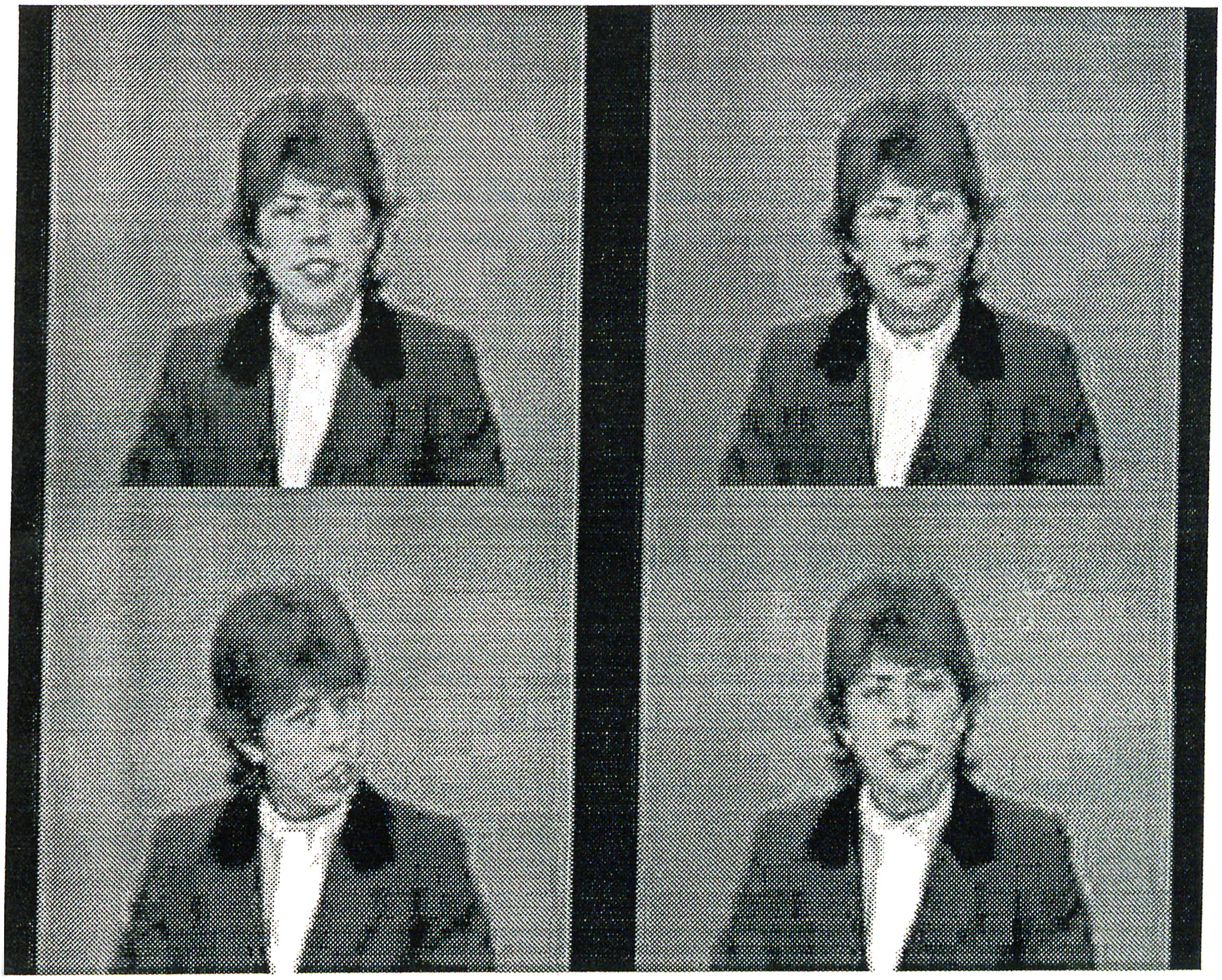


Figure C.2: Comparing buffer control methods. In clockwise order from upper left corner: a) proposed method P1, b) proposed method P2, c) the COST reference method, and d) artefact example for the COST reference method. In d), the person nods downwards. Note that the frame shown in each case is not the same, since variable framerate allowes for the coders to choose a different set of frames.

# Bibliography

[1] COST211ter Simulation Subgroup, *Simulation model for very low bitrate image coding (SIM1)/(SIM2)/(SIM3)*, COST211ter Simulation Subgroup 1992–1993

[2] Ali N. Akansu and Richard A. Haddad, *Multiresolution Signal Decomposition*, Academic Press, 1992

[3] J. W. Woods, ed., *Subband Coding of Images*, Kluwer Academic Publishers 1992.

[4] A. N. Netravali and J. O. Limb, *Picture Coding: A Review*, Proc. IEEE, Vol. 68, No. 3, March 1980

[5] Roy Mickos, *DPCM coding of image sequences with median predictors*, Master's Thesis, Tampere University of Technology, 1992

[6] Ronald W. Schafer and Lawrence R. Rabiner, *A Digital Signal Processing Approach to Interpolation* Proc. IEEE, Vol. 61, No. 6, June 1973

[7] R. Ansari, C. Guillemot, and J. F. Kaiser, *Wavelet Construction Using Lagrange Halfband Filters*, IEEE Tran. on Circuits and Systems, Vol. 38, No. 9, September 1991

[8] *MPEG New York Meeting*, Press Release by MPEG, 16.7.1993

[9] *HDTV "Grand Alliance" Proposal Will be Considered be FCC Advisory Committee* Press Release by Grand Alliance, 24.5.1993

[10] D. E. Dudgeon and R. M. Merserau, *Multidimensional Digital Signal Processing*, Prentice-Hall 1988

[11] *MPEG Meeting Report, 2.-6.11. 1992 London*, Internal report of the Moving Pictures Expert's Group (MPEG)

[12] *Comparison between QCIF and NCIF in SIM1*, internal report of COST, Bilkent University, 1992

[13] *A Buffer Control Scheme for SIM3*, internal report of COST, Bilkent University 1993