Ranveig Nygaard

# Shortest path methods in representation and compression of signals and image contours

NTNU Trondheim
Norges teknisk-naturvitenskapelige universitet

Doktor ingeniøravhandling 2000:64
Fakultet for elektroteknikk og telekommunikasjon

HØGSKOLEN I STAVANGER

621.391 N98a

# Shortest Path Methods in Representation and Compression of Signals and Image Contours

by

Ranveig Nygaard

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOKTOR INGENIØR

Department of Electrical and Computer Engineering
Stavanger University College
Norway

2000

*"Keep your heart with all diligence,*
*For out of it spring the issues of life."*

(Proverbs 4, 23)

# Abstract

Signal compression is an important problem encountered in many applications. Various techniques have been proposed over the years for addressing the problem. The focus of this dissertation is on signal representation and compression by the use of optimization theory, more specific shortest path methods.

Several new signal compression algorithms are presented. They are based on the coding of line segments which are used to approximate, and thereby represent, the signal. These segments are fit in a way that is optimal given some constraints on the solution. By formulating the compression problem as a graph theory problem, shortest path methods can be applied in order to yield optimal compression with respect to the given constraints.

The approaches focused on in this dissertation mainly have their origin in ECG compression and is often referred to as time domain compression methods. Coding by time domain methods is based on the idea of extracting a subset of *significant* signal samples to represent the signal. The key to a successful algorithm is a good rule for determining the most significant samples. Between any two succeeding samples in the extracted sample set, different functions are applied in reconstruction of the signal. These functions are fitted in a way that guarantees minimal reconstruction error under the given constraints. Two main categories of compression schemes are developed:

1. Interpolating methods, in which it is insisted on equality between the original and reconstructed signal at the points of extraction.

2. Non-interpolating methods, where the interpolation restriction is released.

Both first and second order polynomials are used in reconstruction of the signal. There is also developed an approach were multiple error measures are applied within one compression algorithm.

The approach of extracting the most significant samples are further developed by measuring the samples in terms of the number of bits needed to encode such samples. This way we develop an approach which is optimal in the rate-distortion sense.

Although the approaches developed are applicable to any type of signal, the focus of this dissertation is on the compression of electrocardiogram (ECG) signals and image contours. ECG signal compression has traditionally been tackled by heuristic approaches. However, it is demonstrated in this dissertation that optimization algorithms outperform these heuristic approaches by a wide margin with respect to reconstruction error in terms of sum of squared errors. We also develop an approach for compression of image contours, which is an important problem in for instance the MPEG-4 standard.

Using a varied signal test set, extensive coding experiments are presented. Results from our coding methods are compared to traditional time domain ECG compression methods, as well as, to more recently developed frequency domain methods. Evaluation is based on the percentage root mean square difference (PRD) performance measure, the maximum error, execution time and visual inspection of the reconstructed signals. The results demonstrate that the optimization methods have superior performance compared to both traditional ECG compression methods and frequency domain methods, like the ECG optimized filter bank presented in Section 2.3.1.

# Preface

This dissertation is submitted in partial fulfillment of the requirements for the degree *doktor ingeniør* at the Norwegian University of Science and Technology (NTNU), Trondheim, Norway. Associate professor Dag Haugland, University of Bergen (former employee at Stavanger University College (Høgskolen i Stavanger, HiS)), and Professor John Håkon Husøy at HiS have been supervisors.

The work has been carried out at the Department of Electrical and Computer Engineering at HiS from August 1996 to June 2000. Included in the work are compulsory courses corresponding to one year of full time studies, as well as four months of undergraduate lecturer duties at HiS. Seven months were spent as a visiting scholar at the Image and Video Processing Lab, Electrical and Computer Engineering Department, Northwestern University (NU), Evanston, USA. The work has been funded by HiS. The abroad stay at NU was a funded by a scholarship from the Norwegian Research Council (grant number 125677/410).

In the summer of 1996, when I was employed as a research fellow at HiS, Jan Gustav Heber was preparing his dissertation *Compression of ECG signals - Time and frequency domain approaches* [39]. This dissertation covered a broad range of ECG compression methods, and showed some interesting results. The work of my dissertation was initiated by Heber's work, but with a much more narrow scope. While Heber had focused on both time and frequency domain approaches, the time domain approaches became the core of my research.

Most of the main chapters in this dissertation have been presented earlier at national or international conferences or in journals [64, 66, 65, 67, 69, 71, 70, 68, 4].

# PREFACE

# Acknowledgments

First of all, I would like to thank my two supervisors Associate Professor Dag Haugland and Professor John Håkon Husøy. Thank you, Mr. Haugland, for sharing your skills in optimization theory with me in addition to your enthusiasm, guidance and new ideas. This has been crucial for me in keeping my research objectives in perspective through the past four years. Furthermore, thanks a lot for constructive suggestions for improvement of this dissertation. I am also very grateful to you, Prof. Husøy, for inspiring me to start this work, and for being an invaluable support through many discussions, sharing your insight and valuable remarks with me. I would also like to thank Professor Aggelos K. Katsaggelos for making my stay at Northwestern University an unforgettable experience. You really made me feel welcome.

My most sincere thanks to you, Kjersti Engan. I could not have found a better friend and colleague. You have supported me through ups and downs through many years and have been my moral support in both work-related issues and those of a not-so-professional nature. Thank you, Sven Ole Aase for sharing your wealth of ideas with me, in the technical, gourmet an real estate disciplines. I would also like to address words of thanks to my other colleagues in the signal processing group at Stavanger University College for creating a friendly and stimulating working atmosphere and providing both intellectual and social stimulation. I am also indebted to my lab mates during my abroad stay at Northwestern University: Gerry, Chun-Jen, Passant, Faisal, Lisimachos, Stavros, Andrew, Jay, Fei and Laura. Thank you for welcoming me with open arms and including me in the group.

Not only people within the academic world have contributed to the completion of this dissertation. I thank all friends and relatives and especially my dear family for their support. I also express my most sincere gratitude to a very important person to me: Thank you, Per Erik, for all your patience and support and for giving the best hugs in the world!

Finally my thanks go to you, God, who has given me life and the ability to enjoy it.

# ACKNOWLEDGMENTS

# Contents

# Nomenclature

| | |
|---|---|
| $\lvert \cdot \rvert$ | Cardinality of a set |
| $\lVert \cdot \rVert$ | $l_2$-norm |
| $*$ | Optimality indicator |
| $\partial$ | Differentiation operator |
| $E$ | Expectation operator |
| $a_{kij}$ | Coefficients in a polynomial between nodes $i$ and $j$ |
| $A$ | Arc set |
| $C$ | Compression set |
| $C(\cdot)$ | Cost function |
| $d_{ij}$ | Sum of squared errors between samples $i$ and $j$ |
| $d_{ij}^{\infty}$ | The maximum error between samples $i$ and $j$ |
| $\bar{d}^{\infty}$ | Upper bound on the maximum error |
| $D$ | Total sum of squared errors |
| $D^{\infty}$ | Total maximum error |
| $\delta x_k$ | First order difference of the coordinate point $x_k$ |
| $\delta_{y(\cdot)}$ | First order difference of the amplitudes, $y(\cdot)$ |
| $\delta_{\hat{y}(\cdot)}$ | First order difference of the reconstructed amplitudes, $\hat{y}(\cdot)$ |
| $\delta y_k$ | First order difference of the coordinate points $y_k$ |
| $\delta\hat{y}_{mid}(k)$ | $l_2$-distance between $\hat{y}(n_k)$ and the mean value of $\hat{y}(n_k)$ and $\hat{y}(n_k - 1)$ |
| $\delta_{n(k)}$ | First order difference of the sample indices, $n_k$ |
| $\Delta_Q$ | Quantizer step size |
| $\Delta x_n$ | $x$-component of $l_2$-distance between original and reconstructed point |
| $\Delta y_n$ | $y$-component of $l_2$-distance between original and reconstructed point |
| $\varepsilon$ | Maximum error bound in the FAN algorithm |
| $\epsilon_{nij}$ | $l_2$-distance between sample $y(n)$ and $\hat{y}(n)$, $i \leq n \leq j$ |
| $f$ | Parameter probability mass function |

| | |
|---|---|
| $f_{ij}(n)$ | Presumed reconstruction of $y(n)$ between nodes $i$ and $j$ |
| $F$ | Family of parameter probability mass functions |
| $G$ | Graph, consisting of $(V, A)$ |
| $G_{SBC}$ | Coding gain |
| $\gamma_{ij}$ | Slope of the straight line joining samples number $i$ and $j$ |
| $h_k(\cdot)$ | Analysis filter in channel no. $k$ |
| $H(i,j)$ | Convex hull of all samples from index $i$ to $j$ |
| $H_k(z)$ | z-transformed of analysis filter in channel no. $k$ |
| $i$ | Index |
| $I$ | Index set |
| $j$ | Index |
| $J$ | Lagrangian cost function |
| $k$ | Index |
| $K$ | Number of arcs in the graph after reduction |
| $l$ | Index |
| $l(j,m)$ | The length of $P_{j,m}$ |
| $\lambda$ | Lagrangian multiplier |
| $L$ | Filter length |
| $M$ | Upper bound on the number of vertices on a path |
| $M_{sub}$ | Number of subbands |
| $n$ | Index |
| $n_k$ | Extracted sample index no. $k$ |
| $num_{iter}$ | Number of iterations |
| $N$ | Signal length |
| $N_Y$ | Cardinality of $Y$ |
| $\mathcal{O}(\cdot)$ | Order of complexity |
| $p$ | Width of admissible sample point band |
| $p(j,m)$ | Predecessor of $j$ in $P_{j,m}$ |
| $\mathbf{p}$ | Point in the plane made up of $(x, y)$ |
| $\hat{\mathbf{p}}$ | Reconstructed point in the plane made up of $(\hat{x}, \hat{y})$ |
| $P_{j,m}$ | Shortest path to node $j$ visiting exactly $m$ vertices |
| $P_n$ | A path from node 1 up to node $n$ |
| $r_{nln'l'}$ | Number of bits needed to encode the segment $((n,l),(n'l'))$ |
| $R$ | Total bit rate |
| $R(D)$ | Rate distortion function |
| $S$ | Sample point set |
| $\hat{S}$ | Reconstructed sample point set |
| $\sigma^2_{PCM}$ | Variance of reconstruction error associated with basic PCM |
| $\sigma^2_{SBC}$ | Variance of reconstruction error associated with subband coding |
| $\sigma^2_{u_k}$ | Variance of subband channel no. $k$ |

| | |
|---|---|
| $\sigma_y^2$ | Input variance |
| $t_Q$ | Quantizer threshold level |
| $u_k$ | Subband signal in channel no. $k$ |
| $V$ | Vertex set |
| $w_{nln'l'}$ | The cost of arc $((n,l),(n'l'))$ |
| $w_{win}$ | Window size |
| $x_k$ | $x$-coordinate to a point in the plane |
| $\hat{x}_k$ | $x$-coordinate to a reconstructed point in the plane |
| $\bar{y}$ | Mean value of original signal |
| $y_k$ | $y$-coordinate to a point in the plane |
| $\hat{y}_k$ | $y$-coordinate to a reconstructed point in the plane |
| $y(n)$ | Original signal at time index $n$ |
| $\hat{y}(n)$ | Reconstructed signal at time index $n$ |
| $y(n,l)$ | Signal value at index $(n,l)$ in a 2D-plane |
| $Y$ | Admissible point set |
| $\mathcal{Y}$ | Input signal set |
| $\hat{\mathcal{Y}}$ | Reconstructed signal set |
| $\mathcal{Y}_C$ | Compressed representation of the signal set |
| $Q$ | Quantization |
| $R_{yy}(\cdot)$ | Autocorrelation function estimate |

# NOMENCLATURE

# List of Abbreviations

| | |
|---|---|
| acf | auto correlation function |
| AZTEC | Amplitude Zone Time Epoch Coding |
| BIH | Boston's Beth Israel Hospital |
| bpp | bits per point |
| bps | bits per sample |
| CCSP | Cardinality Constrained Shortest Path |
| CORTES | Coordinate Reduction Time Encoding System |
| digraph | directed graph |
| DC | Direct Current |
| DCT | Discrete Cosine Transform |
| DP | Dynamic Programming |
| ECG | ElectroCardioGram |
| EOB | End Of Block |
| FB | Filter bank |
| FIR | Finite Impulse Response |
| HP | Hewlett Packard |
| i.e. | id est |
| JPEG | Joint Photographic Experts Group |
| Kbits | Kilobits |
| LOT | Lapped Orthogonal Transform |
| mitxxx_yyyy | Record number xxx, starting at time yy:yy |
| MIT | Massachusetts Institute of Technology |
| MLII | Modified limb lead II |
| MPEG | Moving Pixel Expert Group |
| MSE | Mean Square Error |
| MUX | Multiplexer |
| NSR | Normal Sinus Rhythm |
| ORD | Operational Rate Distortion |
| PCM | Pulse Code Modulation |
| Pel | Picture element |
| Pixel | Picture element |

| | |
|---|---|
| PRD | Percentage Root-mean-square Difference |
| SAPA | Scan-Along Polygonal Approximation |
| SBC | Subband coding |
| TP | Turning Point |
| VLC | Variable Length Coder |
| WDD | Weighted Diagnostic Distortion measure |

# Chapter 1

# Introduction

The term *signal* often refers to a continuous time and amplitude signal. We are concerned with *digital signals*, which means the signal is obtained by sampling and quantization of a continuous signal. We shall use the word signal when speaking about digital signals throughout this dissertation. Thus a signal will be a sequence whose value at any time is a discrete[1] value. It can also refer to an image, or part of an image, where the amplitude is a function of two spatial coordinates instead of one spatial coordinate and one time variable. In cases where we refer to a sequence of numbers, or vectors (as is often the case with images), the word *data* is often used as a synonym for signal.

A pet child gets many names and this is also the case with the term *compression* in the literature. It is referred to as source coding, data compression, bandwidth compression, and signal compression. These terms may have slightly different meaning, depending on the context. In this context we will use the term *signal compression* and by that we understand a process intended to provide efficient representation of the signal while preserving the essential information contained in it [32].

When talking about a compression technique or a compression algorithm we are actually referring to two algorithms: The compression algorithm taking as input the signal $\mathcal{Y}$ and generating the compressed output $\mathcal{Y}_C$, and the reconstruction algorithm operating on the compressed representation $\mathcal{Y}_C$ to generate the reconstruction $\hat{\mathcal{Y}}$. This is shown schematically in Figure 1.1. We follow the convention of referring to both the compression and reconstruction part of the system together as the compression algorithm.

[1]Taking on a finite or countable infinite number of values.

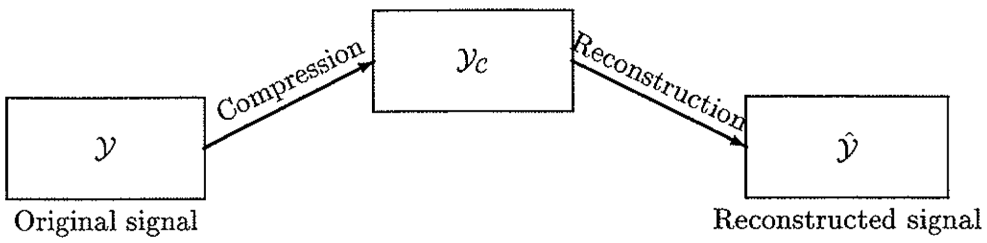


Figure 1.1: Compression and reconstruction.

Compression algorithms can be classified as either *lossless*, which involve no loss of information, or *lossy* where the signal generally cannot be recovered or reconstructed exactly. The algorithms we focus on can belong to either of the two categories, depending on the adjustment of some input parameters. However, to obtain a significant degree of compression, our algorithms will belong to the latter category for most applications, and are often referred to as *near-lossless* compression algorithms. This means that a small amount of distortion can be tolerated. Exactly what is a "small amount" of distortion depends on the particular application and will be discussed in more detail in later chapters.

Many signal processing applications benefit from compression technology, such as image-, video- and audio processing. The current set of compression algorithms[2] have proved successful for a wide range of well specified tasks. However, because signal compression comprises such a wide variety of techniques and applications, many promising avenues for further development are available. The methods we focus on are based on optimization theory. Webster's Dictionary [73] (5 Feb. 2000) defines the word **optimization** as "an act, process or methodology of making something (as a design, system or decision) as fully perfect, functional, or effective as possible; *specifically*: the mathematical procedures involved in this". Applied to our case this means that we define the compression problem in strict mathematical terms. Given some constraints, we aim to find the best possible solution to the problem. This is accomplished by formulating the problem in terms of graph theory. The problem of finding

[2]The word algorithm originates from the name of an early ninth century Arab mathematician, Al-Khwarizmi. He wrote a treatise entitled *The Compendious Book on Calculation by al-jabr and al-muqabala* in which he develops a system for the solutions of various linear and quadratic equations via rules or an "algorithm". The name was changed to algoritni in Latin and from this we get the word algorithm.

the shortest path (path of minimum length) from source node $s$ to terminal node $t$ in the graph is called a *shortest path problem*. By formulating the problem this way, shortest path algorithms may be applied in solving it. The solution will thus be based on a firm mathematical foundation and will be the best solution possible under the given constraints.

*Coding* is the conversion of signal representation from one form to another for some purpose. We use the term coding when referring to the process of converting information from a higher level of representation to the lowest level, i.e., representing it in terms of bits. We generally compress the signal in two steps; First, we *approximate* the signal by extracting information from it. The extracted information is a *representation* of the original signal. We then apply a coding strategy to convert the signal to a bit representation. The *encoder* is the device executing this conversion. Generally we have two distinct coding processes in a digital communication system: *Source coding*, where we attempt to remove redundancy in the signal and represent it in a way as compact as possible, and *channel coding*, where the compressed bit stream is translated into a signal suitable for either storage or transmission. When talking about coding or encoders in this dissertation, we are referring to the part of the system associated with the source coding. Generally the term *coding* of the signal refers to the whole process, both approximation and encoding and the term *coder* includes the whole compression system, both the approximation and encoding part.

This dissertation is focused around the terms signal representation and compression and shortest path methods. This introductory chapter is organized in the following way: In Section 1.1 the different application areas of compression algorithms based on optimization theory are discussed. This is meant as a motivation for the work and is accompanied by the scope and major contributions of this work in Section 1.2. In Section 1.3 we present an outline of the dissertation.

## 1.1 Applications of compression schemes based on optimization theory

The compression techniques developed in this dissertation are in general applicable to any kind of digital signal. However, the methods might be computationally expensive, and for this reason they are best suited for one-dimensional signals. We will in the following bring some examples of application areas for the compression techniques we have developed.

### 1.1.1 Compression of electrocardiograms

An ElectroCardioGram (ECG) is a graphic display of the electrical activity of a heart. It provides essential information to the cardiologist and is used for both monitoring and diagnostic purposes. A typical ECG monitoring device generates large amounts of digital data. The sampling rate typically varies from 125 to 500 Hz and each signal sample may have 8, 12 or 16 bit resolution. This will lead to an accumulation of ECG signal ranging from 60 to 480 Kbits per minute. In addition, there may be several streams of data obtained from different sensors placed on the patient's body. These ECG signals may be stored in a small portable device for later analysis, transmitted to the hospital for on-line diagnosis or stored for long time usage in a case record. There is clearly a need for compression for both storage and transmission of these signals and for this reason many ECG compression techniques have been developed during the last 30 years [25, 6, 22, 61, 44, 90, 92, 96, 72, 77, 7, 52, 42, 11, 91, 56, 19, 37].

### 1.1.2 Compression of image contours

Applications like digital libraries or content-based storage and retrieval have to allow access to data based on object descriptions, like for instance the shape of an object. The importance of shape coding within object-oriented video coding, such as the the MPEG-4 standard [12], further motivates the work of describing object shapes in an effective way.

Object-based treatment of video sequences, necessitated by the emerging content-driven applications, requires efficient representation of object boundaries. The ultimate goal is to allocate an available bit budget optimally among the video scene components (shape, texture, motion) and within each component. In this dissertation we are concerned with the shape part, i.e., given a contour of an object in an image (or a video stream), how can we represent this in the most effective way?

There has been, and still is, significant research activity in this area, see for example [28, 15, 53, 26, 63, 41, 50, 86, 85].

### 1.1.3 Compression of images

Traditionally, image compression is performed by subdividing the image into blocks, each of which is processed by means of a transform in order to obtain a more efficient representation of data.

An alternative viewpoint is to consider the data as samples of a one-dimensional waveform. The waveform can be compressed by approximating it by a piecewise function, with the break points of the function composing the compressed representation. These functions can be fitted in an optimal way, under the given constraints by the use of optimization theory.

Image compression schemes based on this idea are reported in [16, 79, 84].

## 1.2 Contributions of this work

The focus of this dissertation is on development of new approaches to signal representation and compression. A common approach to signal compression is to apply a transform such as the Discrete Cosine Transform (DCT) or the wavelet transform to obtain more efficient representation of data.

We propose a new way of looking at the compression problem. Formulating the problem in terms of graph theory, shortest path methods are applied to solve the problem. Briefly summarized, the major contributions of this dissertation are as follows:

- Development of a general theoretical framework for signal compression from an optimization point of view. Proposal of a compression method based on this - the *Cardinality Constrained Shortest Path method* (CCSP).
- Development of a compression scheme based on the coding of linear line segments which are used to approximate the signal. The segments are fit in an optimal way under the given constraints. Implementation of a coder based on this approach.
- Further development of the compression algorithm, extending it to an approach where polynomials of second order are applied to approximate the signal. Implementation of a coder based on piecewise second order polynomial interpolation.
- Proposal of a new version of the compression scheme where we do not insist on exact equality between the original and the reconstructed signal at any specific points, i.e., a non-interpolating approach. Implementation of two coders, one based on linear approximation and one based on second order polynomial non-interpolating approximations.
- Comparison of piecewise approximations of a signal by the use of polynomials of different order. Investigation of which polynomials result in the overall best performance.

- Compression of image contours by the fitting of linear line segments in an optimal way given some constraints.
- Theoretical framework for optimal signal compression in a rate-distortion sense. Development of two coders based on this approach.

Many of the contributions listed here, as well as related material, have been published in [64, 66, 65, 67, 69, 71, 70, 68, 4].

## 1.3 Dissertation outline

This dissertation is divided into 3 parts; the introductory part, the cardinality constrained shortest path methodology part, and the rate-distortion optimal compression part.

### 1.3.1 Introduction

**Chapter 1:** Introduction to the compression problem. Definition of some central terms and a brief overview of applications.
**Chapter 2:** Background to ECG compression methods, with an introduction to the different approaches previously used. Discussion of the performance measures used in ECG signal compression.

### 1.3.2 Cardinality constrained shortest path methodology

**Chapter 3:** Development of a general framework for signal compression from and optimization point of view. Definition of the problem in rigorous mathematical terms, formulation of the problem by the use of graph theory and presentation of a solution algorithm to the problem - the *Cardinality Constrained Shortest Path* method.
**Chapter 4:** Proposal of several new compression algorithms. They are based on an approach insisting on equality between the original and the reconstructed signal at some extraction points used to represent the signal. Between these points of extraction two different functions are applied in reconstruction of the signal resulting in two different versions of the algorithm; first order polynomials and second order polynomials. Furthermore, development of an approach based on the incorporation of two error measures into one compression scheme, offering control of both the maximum error and the sum of squared errors.
**Chapter 5:** Development of two new coders in which the interpolation restriction is released. Comparison to the coders developed in Chapter 4.
**Chapter 6:** Proposal of a coder for image contours based on the coding of line pieces which are used to approximate the signal.

### 1.3.3 Rate-distortion optimal compression

**Chapter 7:** Development of a new approach to signal compression, based on finding the minimum distortion solution given an upper bound on the number of bits available. The inverted problem is also solved, i.e., finding the minimum rate solution given an upper bound on the distortion.

Conclusions of the work are given in **Chapter 8**.

In addition there are two appendices:

**Appendix A:** Overview of the ECG test signals.

**Appendix B:** A detailed mathematical description of the computation of arc lengths in the CCSP algorithm in the polynomial interpolating reconstruction case of Section 4.3.

# Introduction

# Chapter 2

# Signal compression methods

Numerous compression algorithms have been developed over the years. It would be an enormous task to describe all of them, and doing so is not the aim of this dissertation. Although the compression schemes we develop are applicable to any kind of signal, we have chosen to focus our attention mainly on ElectroCardioGram (ECG) signals and apply our compression algorithms to these signals. To get an overview of the area of ECG compression, and an understanding of which aspects are important to ECG compression, we will give an introduction to ECG compression in this chapter.

An ECG is a graphic tracing of the variations in electrical potential caused by the excitation of the heart muscle and detected at the body surface [83]. These signals may be transmitted to a hospital in order to get an early diagnosis, stored for long time usage in a case record or stored in small hand-held or implantable devices to be analyzed later on. As we do not have unlimited transmission or storage capacity this calls for efficient compression methods in order to keep such signals in manageable sizes. The compression must be done in a way that guarantees accurate reconstruction of the signal. Many signal compression techniques for ECG waveforms have been proposed. Roughly, they can be classified into two categories:

- Direct signal compression methods which analyze the signal in the time domain.
- Other ECG compression methods.

An important issue in ECG signal compression is which *error measure* to apply. Since ECG signals are biomedical signals, it is of crucial importance that the diagnostic information contained in the original signal is not changed or removed by the compression algorithm. The different performance measures normally used in ECG compression are discussed in Section 2.1. A brief overview of the two main categories of ECG compression algorithms is given in this chapter. Direct signal compression methods are presented in Section 2.2 along with an introduction to one of the specific methods that belongs to this category, the FAN algorithm [25]. Results from the FAN algorithm are used throughout this dissertation in comparison with experimental results from the new techniques presented, and it is therefore appropriate to describe the mode of operation for the FAN algorithm as is done in Section 2.2.1. Section 2.3 is devoted to other ECG compression methods.

## 2.1 Measures of performance

The performance of a compression algorithm can be measured in a number of different ways like the distortion between the original and the reconstructed signal, the amount of compression, the complexity of the algorithm, the execution time on a given machine and the visual similarity between the original and the reconstructed signal. We will measure the performance of our compression algorithms in a manner as complete as possible and will cover all of these areas.

### 2.1.1 Distortion measures

Since ECG signals generally are compressed with lossy compression algorithms, we have to have a way of quantifying the difference between the original and the reconstructed signal, often called *distortion*. Traditionally, two different distortion measures have been applied in ECG compression algorithms: The *maximum error* and the *sum of squared errors*. The maximum error is given by

$$D^{\infty} = \max_{0 \le n \le N-1} |y(n) - \hat{y}(n)|, \tag{2.1}$$

and the sum of squared errors by

$$D = \sum_{n=0}^{N-1} (y(n) - \hat{y}(n))^2, \tag{2.2}$$

where $y(n)$ and $\hat{y}(n)$ are the original and the reconstructed signal, respectively, and $N$ is the signal length. Using the sum of squared error, we assure that

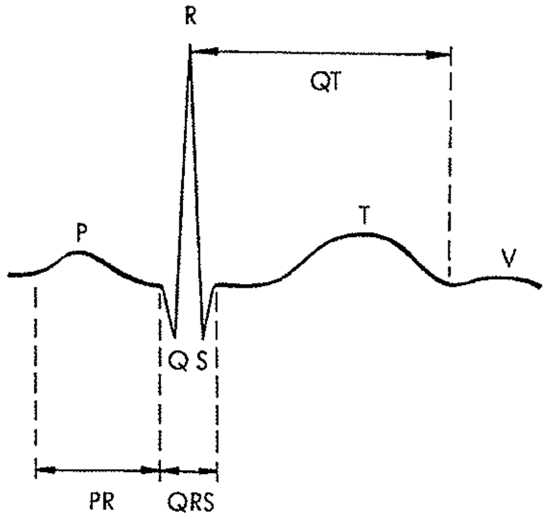


Figure 2.1: Normal Sinus Rhythm (NSR) [18].

the overall error will be as small as possible, however, we do not guarantee anything about the error at each time index, $n$, of the signal, as is the case for the maximum error. Which error measure is the most relevant one is an open question, and will be dependent on the specific application.

Recently, there has been some activity in finding a distortion measure which is more suitable for ECG signals, such as the Weighted Diagnostic Distortion Measure (WDD) [97]. This error measure is correlated with diagnostic information in the signal and is based on comparing the PQRST complex features of the original ECG signal and the reconstructed one, such as P wave duration, T wave amplitude, QRS shape, QT duration, ST elevation, etc., see Figure 2.1. This distortion measure limits the application of the compression algorithm to Normal Sinus Rhythm [1], which is a considerable limitation. As no general framework for distortion measures in ECG compression has been established yet, we have chosen to concentrate on the two traditional distortion measures reported in Equations (2.1) and (2.2).

In order to be able to do an inter-method comparison of the performance of the different coders, we need a quantitative measure for the entire reconstructed signal indicating the performance of the coders. We have applied two separate error measures, one based on the maximum error as given in Equation (2.1) and one based on the sum of squared errors, namely the commonly used Percentage

[1]Normal heart rhythm originating in the sinoatrial node [83].

Root-mean-square Difference (PRD) [93] defined as

$$PRD = \sqrt{\frac{\sum_{n=1}^{N}[y(n)-\hat{y}(n)]^2}{\sum_{n=1}^{N}[y(n)-\bar{y}]^2}} \times 100\%, \qquad (2.3)$$

where $\bar{y}$ is the mean value of the original signal. The smaller the PRD, the closer the reconstructed signal is to the original.

We would like to make a comment on the different performance measures in relation to different compression methods. It would be reasonable to evaluate all compression schemes in terms of the same distortion measure they are minimizing, that is, if the object is to minimize the sum of squared errors, the PRD should be used as an evaluation criterion. On the other hand, if we attempt to minimize the maximum error, the evaluation criterion in Equation (2.1) should be applied. However, in order to do an inter-method comparison, it is necessary to evaluate the results from different algorithms in terms of the same distortion measure. In the sections containing experimental results, we have chosen to evaluate all results in terms of both the PRD and the maximum error independent of which error criterion that is minimized. We accompany these with a visual inspection of the reconstructed signal in addition to reporting the computational complexity and the execution time of the algorithm. This will give a complete picture of the performance of the different methods.

### 2.1.2 Compression ratio

Compression by time domain methods are generally accomplished by keeping a set of *significant* signal samples. A logical way of measuring how well such compression methods compress a given signal, is to compare the number of retained samples in the compressed signal to the number of original signal samples. This is named the *sample reduction ratio*, defined as the number of samples in the original signal per retained signal sample.

In order to be able to compare results from different coding schemes in a fully justified way, encoding of the extracted signal samples will have to take place. This way, evaluation is in terms of *bit rate*, defined as the average number of bits used to represent one signal sample in the original signal.

### 2.1.3 Computational complexity

One of the most important behavioral aspects of a computation is the complexity of the computation, i.e., the amount of computation resources used.

The time it takes for a computer program to execute depends on how much data it has to input and how many operations it does with the data. We will express time requirements of algorithms in terms of the number of basic steps required for the execution of the algorithm. In this context basic steps may be arithmetic operations, comparisons, branching instructions and so on, and we assume that all these operations require unit time. To estimate how the time it takes to run a program varies with the size of the input, we consider all inputs of a given size $N$ together, and we define the complexity of the algorithm for that input size to be the worst-case behavior of the algorithm for any of these inputs.

Essentially we are interested in an estimate of the rate of increase in the time it takes to run a program as its input gets bigger, i.e., the rate of growth of the complexity of the algorithm. For example, some programs take about twice as long to run if you give them twice as much data. Other programs that are less efficient may take about 4 times as long to run if they are given twice as much data. To express the rate of increase in execution time related to an input, we apply the following definition [74]:

**Definition 1** *Let $f(n)$ and $g(n)$ be functions from the positive integers to the positive reals.*

a) *We write $f(n) = \mathcal{O}(g(n))$ if there exists a constant $c > 0$ such that, for large enough $n$, $f(n) \leq cg(n)$.*

b) *We write $f(n) = \Omega(g(n))$ if there exists a constant $c > 0$ such that, for large enough $n$, $f(n) \geq cg(n)$.*

c) *We write $f(n) = \Theta(g(n))$ if there exist constants $c, c' > 0$ such that, for large enough $n$, $cg(n) \leq f(n) \leq c'g(n)$.*

The definition of $\mathcal{O}(g(n))$ states that, up to a constant factor, the function $g(n)$ gives an upper bound on how the algorithm is performing for large $n$, saying in effect that as $n$ gets larger, the growth in execution time will be no worse than that shown by $g(n)$.

### 2.1.4 Execution time

By execution time, we mean the wall clock time an algorithm takes to run, measured on a specific system using specific hardware and software configurations. This is often referred to as a benchmark. Benchmarks are dependent on the specific machine, compiler, input values and programming environment. Nevertheless, they tell us something about the fulfillment of *real time* requirements, i.e., the ability of the algorithm to analyze the signal as it comes in. This is an important aspect of a compression algorithm.

### 2.1.5 Visual similarity

The purpose of an ECG compression algorithm is to achieve a maximum degree of compression while distorting the signal as little as possible. The quality required from the reconstructed signal is dependent on the specific application. For this reason, it is hard to give a general guideline as to what the maximum distortion that can be tolerated is, in order to obtain a reconstructed signal with the diagnostic information contained. This will have to be decided by visual inspection of the signal.

However, there are a some artifacts which are generally undesirable in the reconstructed ECG signal: Areas of the signal corresponding to the same electrical activities in the heart, should not be distorted in different ways between different periods of the signal. For example, the P-wave must not contain one artifact due to compression at one spot of the signal and a different at a another spot. This will apply to ECG signals of periodic character, such as Normal Sinus Rhythm (NSR). For other rhythms than NSR, with less periodical characteristics, an important issue is that areas of the signal where the rhythm suddenly changes should not be distorted. These areas are of vital diagnostic importance as they can be very useful in deciding what mechanism that lead to a change of rhythm, and it is desirable to be able to reconstruct these segments with high fidelity.

## 2.2 Direct signal compression methods

Direct signal compression methods are also known as time domain techniques dedicated to compression of ECG signal. The mode of operation is to extract a subset of *significant samples* from the original sample set[2]. Which signal samples are significant, depends on the underlying criterion for the sample selection process. To get a high performance time-domain compression algorithm, much effort should be put in designing intelligent sample selection criteria. The original signal is reconstructed by an inverse process, most often by drawing straight lines between the extracted samples. This category includes the FAN [25], CORTES [6], AZTEC [22] and Turning Point [61] algorithms. The *Cardinality Constrained Shortest Path* technique [37], presented in Chapter 3 also fits into this category, as well as, the rate distortion optimized method, presented in Chapter 7.

[2]This does not apply to the AZTEC coder which decomposes the ECG signal into segments identified as either plateaus or slopes.

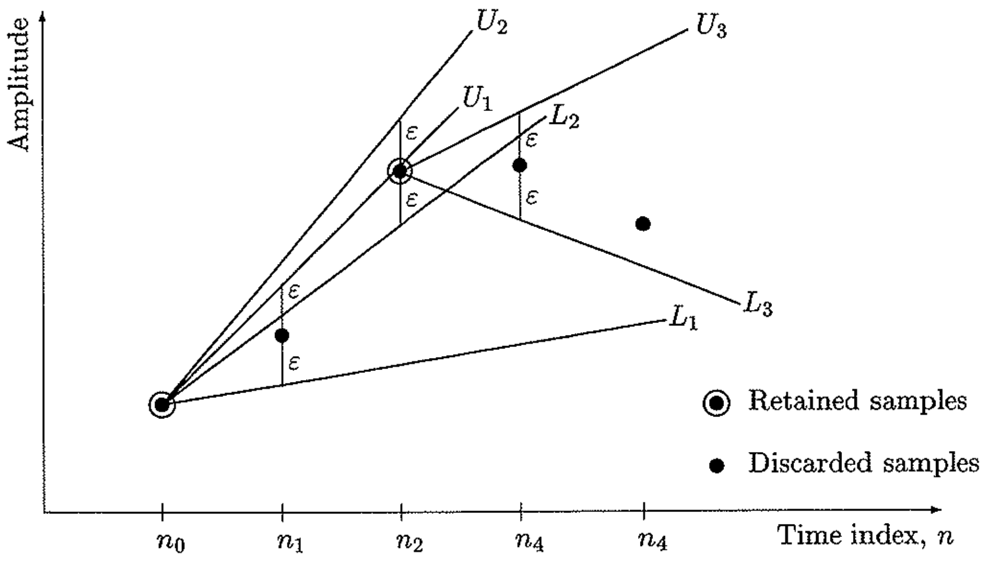


Figure 2.2: Illustration of the FAN algorithm.

### 2.2.1 The FAN algorithm

A frequently cited heuristic is the FAN algorithm [25]. The basic idea of this algorithm is to identify signal segments where a straight line serves as a close approximation, and to discard all but the terminal points along this line. When significant deviations from this line are detected, the corresponding samples are included in the extracted signal samples.

The FAN algorithm accomplishes the above idea by initially accepting the very first sample point as shown in Figure 2.2. Next it computes a range within which succeeding samples must be found if they are to be fit by a straight line. This is done by drawing two lines $(U_1, L_1)$ between the initial point and the next sample point plus a specified threshold $(\pm\varepsilon)$ as shown in Figure 2.2. If the third sample point falls within the area bounded by the two lines, new slopes $(U_2, L_2)$ are drawn between the initial point and the third sample point plus the same specified threshold. These new lines $(U_2, L_2)$ are then compared to the previously stored lines $(U_1, L_1)$ and the most restrictive lines are retained $(U_1, L_2)$. The process is repeated, comparing future sample values to the most restrictive lines. Whenever a sample falls outside the area bounded by the most restrictive lines, its predecessor is accepted as a significant sample, and the procedure above is repeated from this point on.

The FAN method offers control of the absolute error by guaranteeing that the error between the reconstructed and original signal is less than or equal to the threshold, $\varepsilon$. However, the compression ratio is beyond management. The FAN method was originally reported and tested on ECG signals by Gardenhire [30, 29]. It has been exhaustively used and tested on ECG signals [23, 17, 75].

### 2.2.2 Other direct signal compression methods

Numerous variations of different time domain coders have been suggested. There are different strategies in how to make the eliminate-or-keep decision on signal samples, and other clever ideas to compress signals. These include the original AZTEC (Amplitude Zone Time Epoch Coding) [22], the CORTES (Coordinate Reduction Time Encoding System) [6], the TP (Turning Point) [61] and the SAPA (Scan-Along Polygonal Approximation) [44] algorithms. There have also been some attempts of improvement to time domain algorithms such as SLOPE [90] and AZTDIS [92].

The FAN algorithm is a computationally efficient heuristic approach reported to yield high compression ratios, as well as, producing reconstructed signals of high fidelity [46, 39]. In this dissertation, results from the FAN algorithm are used as a "best case" representative for the whole class of traditional direct signal compression methods to be compared to experimental results from our own developed approaches.

## 2.3 Other ECG compression methods

Many different compression schemes have been applied to ECG signals. One main category is transform compression methods. These methods mainly utilize the spectral and energy distributions of the signal. Generally this means processing the input signal by means of some transform, and properly encoding the transformed output. Signal reconstruction is achieved by an inverse transformation process. This category includes traditional transform coding techniques applied to ECG signals such as the Karhunen-Loève Transform [96, 72], Fourier Transform [77], Cosine Transform [7] and Walsh Transform [52], as well as, subband-techniques [42, 11, 91].

Vector Quantization [56], Wavelet Transform [19] and other compression methods have also been applied to ECG signals [33, 54, 62].

Which methods that perform best among the direct and other ECG compression methods, is an ongoing discussion. We have compared a number of methods from both categories in [4]. In the following section we will focus on a subband coding technique from which we compare experimental results to results obtained with our rate-distortion optimal ECG encoding scheme in Chapter 7.

### 2.3.1 Subband coding

A subband coder splits the input signal into a collection of approximately disjoint frequency bands. If the resulting subbands have the same extent in the frequency domain, the subband decomposition is said to be uniform. Since the bandwidth of each subband is reduced by an amount corresponding to the number of subbands, - say $M_{sub}$, each subband can be sub-sampled by a factor of $M_{sub}$. Thus, the number of signal samples in the *critically* sampled subbands are the same as in the input signal. Since the importance of the various subbands is unequal, – compression is obtained by representing (quantizing) the less important subbands with a small number of bits. Normally, the signal energy is concentrated in the lower frequency subbands, implying that the higher frequency subbands can be represented with a small number of bits.

The subband splitting is performed by an *analysis filter bank*, see Figure 2.3. Reconstruction of the decoded signal is performed by a *synthesis filter bank* operating on the signals derived from the bit efficient representation of the analysis filter bank outputs. If, in the absence of quantization, the output of a cascade of an analysis and synthesis filter bank is identical to the input, the filter bank, or more correctly the filter bank pair, is said to possess the *perfect reconstruction property*. Most subband coders for speech and image signals are based on perfect or almost perfect reconstruction filter banks. More details on the theory of filter bank construction can be found in [94, 8, 76]. Figure 2.4 shows the main components in the subband coder system. The components are:

- The analysis filter bank performs the subband split.
- The quantizer and encoder represent the various subbands in a bit efficient manner.
- The channel is the medium to which the compressed signal is transferred. This could, for example, be a telephone link, or a hard disk.
- The decoder and the inverse quantizer produce an approximation to the subband signals based on the encoded representation.

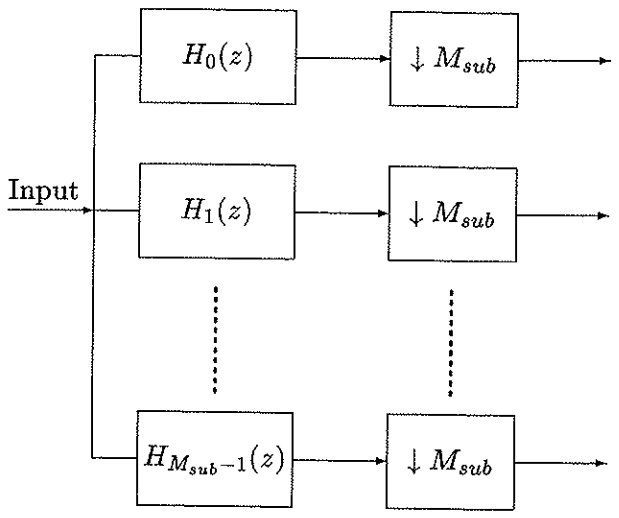


Figure 2.3: Parallel analysis filter bank. Sub-sampling by a factor of $M_{sub}$ is denoted by $\downarrow M_{sub}$.

- The synthesis filter bank reconstructs the decoded signal.

A complete system to be used for ECG collection, compression and storage would in addition have a data collection module, user interface as well as other utility modules.

### Bit efficient representation of the subbands

After the signal has been split into subbands it is in a form well suited for quantization and coding. In the system described here a uniform quantizer is used together with run-length and Huffman coding. Figure 2.5 illustrates the quantization and thresholding operation performed on each sample. As the figure shows, the input samples are represented by an amplitude selected from a discrete set of levels, each separated by $\Delta_Q$. If the sample amplitude has absolute value below a selected threshold $t_Q$ it is set equal to zero.

As will be evident shortly, it is convenient to reorganize the subband samples in the following way: The subband samples are put into vectors such that each element is taken from a separate subband. The sequence is such that the first entries in the vector are taken from the low frequency bands, whereas the latter entries are taken from the higher bands. We illustrate this with an example: Suppose we have a four subband split with 3 signal samples in each of the subbands. We can picture this as follows:

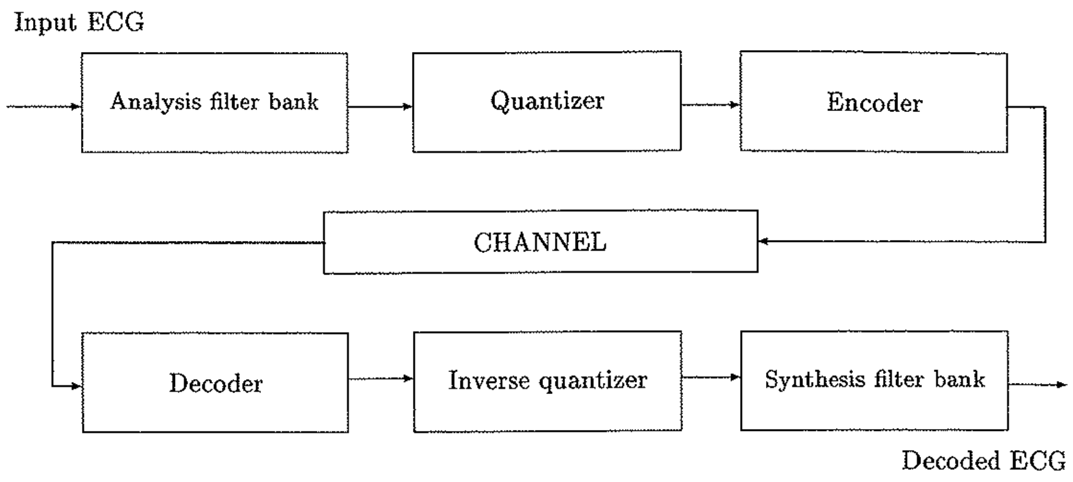


Figure 2.4: Building blocks in a subband coder system.

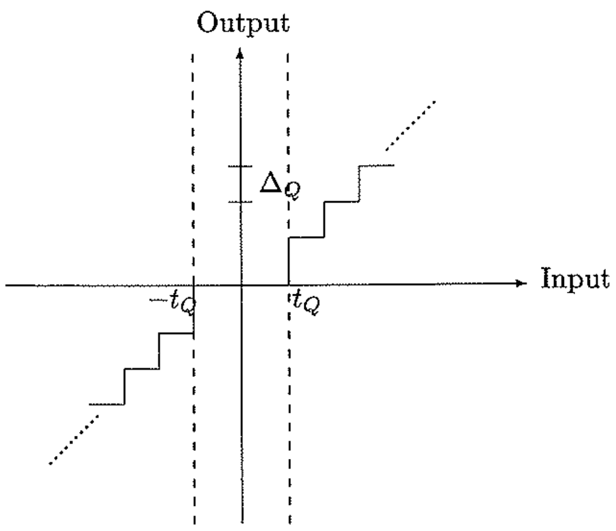


Figure 2.5: Uniform quantizer with thresholding.

```
abc def ghi jkl,
```

where *abc* are the 3 samples of subband 0, *def* are the 3 samples of subband 1 and so on. After reorganization, – or scanning, we get 3 vectors with elements given by:

```
adgj behk cfil
```

This results in a number of vectors equal to the total number of subband samples divided by the number of bands.

Given that most of the energy in the signal is in the lower subbands, it is reasonable to assume, – under most circumstances, that after quantization and thresholding a substantial number of higher band subband samples will be set to zero. Since these zeros tend to occur in clusters, – as a direct consequence of the way in which the data are organized in vectors, run-length coding of these zeros makes sense.

The run-length coding is done by representing the above mentioned vectors in the form *(Run, Level)*, where *Run* is the number of zeros before each non-zero sample, and *Level* is the amplitude of the *quantized* subband sample following a number of zeros given by *Run*. The event that the last samples of the vector are all zeros is represented with the special code *EOB* (end of block). We illustrate by an example:

```
before run length coding 01003200...0
after run length coding  (1,1) (2,3) (0,2) EOB
```

After the signal has been run-length coded it is finally encoded with a Huffman coder [21].

### ECG-optimized filter banks

In this subsection an attempt is made to exploit current knowledge of subband coding of images to design good filter banks for use with ECG signals. In [5] a flexible filter bank design method was presented. Using gradient search techniques, almost any kind of mathematical optimization criteria can be utilized.

When designing a filter bank for image compression at low bit rates, the following criteria are important [76]:

- Perfect, or near-perfect reconstruction in the absence of quantization noise.

- High coding gain.
- Zero DC-leakage.
- Short filter length to minimize ringing noise.
- Absence of blocking effects.
- Nonunitarity. The analysis filters may differ from the corresponding synthesis filters.

The majority of the given criteria are similarly important when compressing ECG signals. However, there are differences: When evaluating 2-dimensional grey-tone (or color) images, the human visual system responds differently than when evaluating 1-dimensional curves. For images, blocking effects in smooth image areas, and ringing noise, can be annoying. Ringing noise is characterized by over- and undershoots in the reconstructed signal close to abrupt transitions. Blocking effects result from the upsampling procedure in the synthesis filter bank.

Evaluation of reconstructed images and ECG signals is also different: In image compression the purpose often is that the decompressed images should "look good", i.e., it does not matter if there are minor degradations as long as they look natural. This is not always the case for ECG signals: It is important that no artifacts arise during compression.

A common optimization criterion for transform and subband coding is the so-called *coding gain*, defined by [47]

$$G_{SBC} = \frac{\sigma_{PCM}^2}{\sigma_{SBC}^2} = \frac{\sigma_y^2}{\left(\prod_{k=0}^{M-1} \sigma_{u_k}^2\right)^{\frac{1}{M_{sub}}}}, \tag{2.4}$$

where $\sigma_{PCM}^2$ and $\sigma_{SBC}^2$ are the variances of the reconstruction error associated with basic PCM and subband coding, respectively, and $M_{sub}$ is the total number of subband channels. The symbols $\sigma_y^2$ and $\sigma_{u_k}^2$ denote the input variance and the variance of subband channel no. $k$, respectively.

It is well known that images, as a rule, are of lowpass character. A common signal model is the autoregressive (AR) model of order one, with correlation factor $\rho = 0.95$ [45]. The associated power spectral density has lowpass shape. Using this model, the coding gain of Equation (2.4) can be maximized by adapting the subband filter coefficients [5].

### Decorrelation of a heartbeat signal

The subband variances can be found statistically as[3]:

$$\begin{aligned}\sigma^2_{u_k}(n) &= E[u_k(n)^2] \\ &= \sum_{j=0}^{L-1}\sum_{i=0}^{L-1} h_k(j)h_k(i)E[y(n-j)y(n-i)],\end{aligned} \tag{2.5}$$

where $u_k$ is subband signal in channel no. $k$, $h_k(\cdot)$ is the analysis filter in the same channel, and $L$ is the filter length. Due to the nonstationary nature of ECG signals, the variance will to a great extent depend on the time index $n$.

In the following we concentrate on the second order statistics of one normal heartbeat signal. A filter bank which performs good signal decorrelation in these regions will perform well. Figure 2.1 shows the basic shape of a heartbeat signal, where the QRS complex and also the P and T waves are indicated.

Substituting the expectation operator with time-averaging over $K$ consecutive samples,

$$E[u_k(n)^2] \rightarrow \frac{1}{K}\sum_{n=0}^{K-1} u_k(n)^2, \tag{2.6}$$

Equation (2.5) can be re-written as

$$\hat{\sigma}^2_{u_k}(n) = \sum_{j=0}^{L-1}\sum_{i=0}^{L-1} h_k(j)h_k(i)R_{yy}(i-j), \tag{2.7}$$

where $R_{yy}(\cdot)$ is the biased autocorrelation function (acf) estimate:

$$R_{yy}(k) = \frac{1}{K}\sum_{n=0}^{K-|k|-1} y(n)y(n+k). \tag{2.8}$$

An acf estimate was obtained in [35] using ensemble averaging over 18 heartbeats taken from different patients. The resulting function is shown in Figure 2.6.

[3]We tacitly assume all signals to have zero mean value. Any DC-component in the lowpass subband can be subtracted prior to coding.

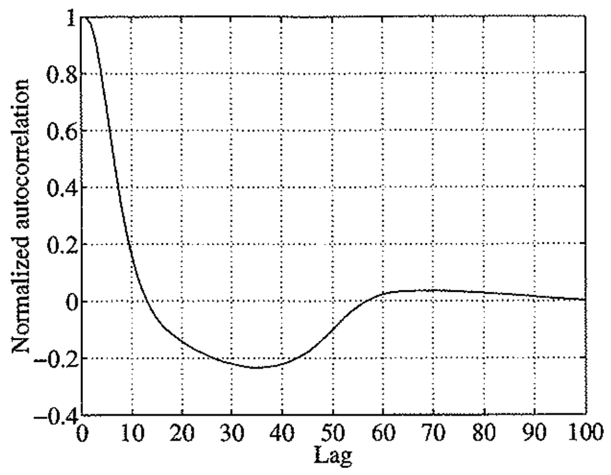


Figure 2.6: Estimated autocorrelation function for normal heartbeat.

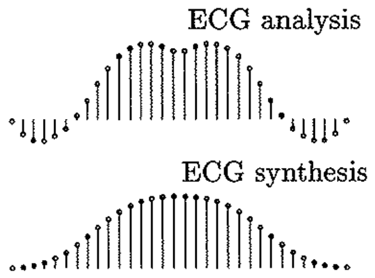


Figure 2.7: Lowpass unit pulse responses for the ECG-optimized nonunitary filter bank.

In [42], it was concluded that 16 channels was suitable for compression of ECG signals, although the exact number of channels did not significantly alter the coding results. We consider filter banks with 16 channels only in this section.

Using gradient-search techniques, a parallel (see Figure 2.3), uniform, nonunitary, 16-channel FIR filter bank was optimized using the criteria listed earlier. The coding gain was maximized assuming the signal acf of Figure 2.6, and the filter lengths were limited to 32 taps. The optimized filter bank is nonunitary. It is therefore interesting to highlight the differences between corresponding analysis and synthesis responses. The lowpass analysis and synthesis unit pulse responses of the optimized filter bank are shown in Figure 2.7.

There is a dramatic difference between the analysis and synthesis lowpass responses of the optimized filter bank: The analysis response is bimodal, and while it tapers off to zero at both ends, the decay is not as smooth as that of the synthesis response. The smoothness of the synthesis response is related to the blocking free reconstruction properties of this filter bank, and is in accordance with Malvar's approach [55]. An additional advantage of the smooth, monotonously decaying synthesis lowpass response, is that it does not produce ringing artifacts in the reconstructed signal. A signal edge reconstructed with this response, will be "smeared out", but without any over- or undershoots.

## 2.4 Summary

This chapter identifies two different main categories of ECG compression methods: Direct signal compression methods and other ECG compression methods. Performance evaluation of compression algorithms is discussed. The maximum error, the sum of squared error, the sample reduction ratio, the bit rate, the computational complexity and the execution time are defined the way they will be used throughout this dissertation. Some quality requirements specific to compression of ECG signal are stated. The FAN method, a heuristic ECG compression scheme much used in literature, is explained as an example of a direct signal compression method. Furthermore, a subband technique, more precisely an ECG-optimized filter bank [1, 2, 4, 3], which will be used for comparison to our new techniques in later chapters, is introduced.

# Chapter 3

# Cardinality constrained shortest path method

In this chapter we introduce the *Cardinality Constrained Shortest Path (CCSP)* method, a time domain signal compression scheme based on a rigorous mathematical model of the compression problem. By formulating the compression problem as a graph theory problem, optimization theory may be applied in order to achieve the best compression possible under the given constraints.

In Section 3.2 the problem is defined in strict mathematical terms. In Section 3.3 it is shown how the problem can be modeled using graph theory, making it suitable for solving with a shortest path algorithm as is shown in Section 3.4. Section 3.5 is devoted to the computational complexity of the algorithm.

## 3.1 Motivation

Compression of ECG signals has traditionally been tackled by heuristic approaches, such as the ones described in Section 2.2. Coding by these time domain methods is based on the idea of extracting a subset of significant signal samples to represent the signal. The key to a successful algorithm is a good rule for determining the most significant samples. Decoding is based on interpolation in this set. Despite the incorporation of intelligent sample selection rules, all heuristics suffer from lack of ability to extract signal samples in a manner that guarantees the smallest reconstruction error possible. However, by a rigorous mathematical model of the compression problem, and by a corresponding solution algorithm, the minimum set of samples may be achieved.

One such method, based on a rigorous mathematical model of the entire sample selection process, is the CCSP algorithm [37, 65, 66, 68, 69]. By modeling the signal samples as nodes in a directed graph, optimization theory is applied in order to achieve a maximum degree of compression, under the given constraints. Any pair of nodes are connected with an arc, the direction of which is given from the sample order. Including a particular arc in the solution corresponds to letting the end nodes of the arc constitute consecutive samples in the extracted subset of signal samples. The length of each arc in the digraph can be defined in a variety of ways, as will be shown in later chapters. As for now, let us denote the length of the arc between nodes $i$ and $j$ by $d_{ij}$. The problem is then to find a shortest path through the graph from the first to the last node, given an upper bound on the number of nodes that are allowed to be included in the path. Furthermore, the length of the path is given as a sum of arc lengths, $d_{ij}$, $i = 1, \ldots, N-1, j = i+1, \ldots, N$ along the path.

The problem described above is a special case of the *resource-constrained shortest path problem* [9, 14]. The resource in question is the number of vertices on the path. Unlike the general version of the problem, our model contains only one resource constraint [10, 34, 49, 78]. Omitting the resource constraint, we simply face the frequently studied *shortest path problem* [24].

In our case the particular choice of resource is the the number of nodes included in the path. For this reason, we term our problem the *cardinality constrained shortest path problem* (CCSP).

In an optimization approach, we define an *objective function* representing our ultimate goal. Other criteria present are represented as *constraints.* Our objective function will mainly be to minimize distortion of the reconstructed signal. Different constraints will be incorporated in our problem, such as an upper bound on the number of samples which may be part of the significant sample set, or an upper bound on the maximum error.

## 3.2 Optimization model

Denote the samples taken from an ECG signal at constant interval by $y(1), y(2), \ldots, y(N)$. Let $M$ denote the bound on the number of extracted samples. We seek an appropriate *compression set* $C = \{n_1, n_2, \ldots, n_M\} \subseteq \{1, 2, \ldots, N\}$ and

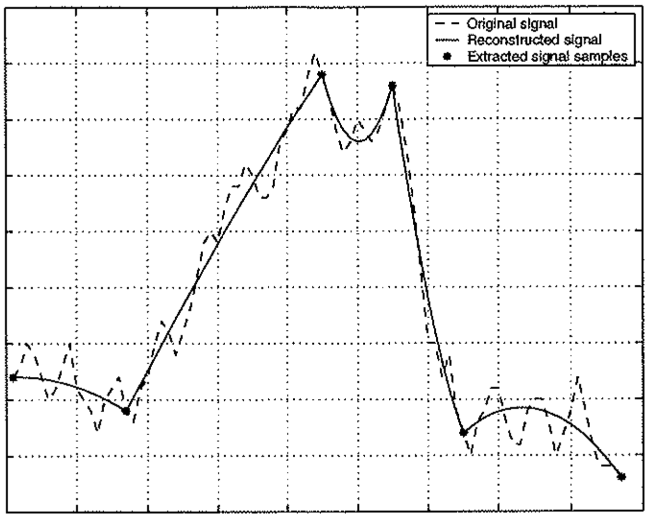


Figure 3.1: Example of short segment of original and reconstructed signal.

the corresponding sample values $y(n_1), \ldots, y(n_M)$ to represent the original signal. Assume $n_1 = 1$ and $n_M = N$. The approximation is then given by

$$\hat{y}(n) = \begin{cases} y(n) & \text{if } n \in C, \\ f_{n_k,n_{k+1}}(n) & \text{if } n \notin C, n \in \{1, 2, ..., N\} \text{ and } n_k < n < n_{k+1}. \end{cases} \tag{3.1}$$

Here $f_{n_k,n_{k+1}}(n)$ denotes a presumed reconstruction of $y(n)$ based on $y(n_k)$, $y(n_{k+1})$ and all the intermediate samples, used to extract parameters to describe the line piece connecting $n_k$ and $n_{k+1}$. The support of $f_{n_k,n_{k+1}}(n)$ is dependent on the particular definition of the function under consideration. The support given in Equation (3.1) is valid in the interpolating approximation case, where we insist on exact equality between the reconstructed and the original signal samples at the points of extraction. In the non-interpolating approximation case, where we relax the interpolation constraint, the support is slightly different, and will be given as $n \in [n_k, n_{k+1})$ if $n_{k+1} < N$ and $n \in [n_k, n_{k+1}]$ if $n_{k+1} = N$. By letting $f_{n_k,n_{k+1}}(n)$ be defined on the domain $[n_k, n_{k+1})$ as long as $n_{k+1} < N$, we represent the signal by functions, $f_{n_k,n_{k+1}}(n)$, which are not connected in the time indices. This means that if one segment ends in the index $j$, the next segment starts in $j + 1$. This is a natural way of representing a *non-interpolating* approximation when working on digitized signals. Figure 3.1 shows a short segment of an original and a reconstructed ECG signal. In this case $f_{n_k,n_{k+1}}(n)$ is a second order polynomial and an interpolating approximation approach is applied.

$f_{n_k,n_{k+1}}(n)$ may represent any kind of function. In its simplest form it is a straight line, but it may also be a polynomial of higher degree like the one in Figure 3.1. Which $f_{n_k,n_{k+1}}(n)$ gives the best approximation is a tradeoff between the quality of the approximation and the number of parameters needed to describe each polynomial uniquely. Clearly, we will be able to get a closer approximation to the original signal by applying a higher degree polynomial, but the improvement in performance may be lost in the cost of representing the polynomials. Investigation of which polynomials results in the overall best performance of the coder is one of the topics of this dissertation.

Defining the problem this way, we get a piecewise approximation to the original signal. Between two sample amplitudes corresponding to two succeeding elements of $C$, different functions are used in reconstruction of the signal. The choice of $C$ will thus have a vital importance for the quality of our approximation of the signal.

## 3.3 Graph formulation

Define the directed graph $G = (V, A)$ whose *vertex set* $V = \{1, 2, ..., N\}$ and *arc set* $A$ contains node pairs $(i, j)$ where $i, j \in V$ and $i < j$. If $n_1, n_M \in V$, the set $(n_1, n_2, ..., n_M)$ is said to be a *path* from $n_1$ to $n_M$ in $G$ if $n_1, ..., n_M \in V$ are distinct vertices and $n_1 < n_2 < \cdots < n_M$. Let $P_n$ denote a path from node 1 up to node $n$. The length of each arc $(i, j)$ in $A$ is given as the contribution to the total reconstruction error resulting from the elimination of samples between nodes $i$ and $j$. This can be expressed as

$$d_{ij} = \sum_{n=i}^{j-1} (\hat{y}(n) - y(n))^2 \quad \text{if } j < N, \tag{3.2}$$

$$d_{ij} = \sum_{n=i}^{j} (\hat{y}(n) - y(n))^2 \quad \text{if } j = N. \tag{3.3}$$

Equations (3.2) and (3.3) cover both the interpolating and the non-interpolating case. In the non-interpolating case, the expressions in Equations (3.2) and (3.3) follow as a natural consequence of the definition of the support of $f_{n_k,n_{k+1}}(n)$. In the interpolating case, $\hat{y}(n) = y(n)$ for $n = N$ by the very nature of the problem, and hence the distortion will be zero at this point.

The arc in a linear interpolation case is illustrated in Figure 3.2. Here $\epsilon(n) = \hat{y}(n) - y(n)$ and the length of the arc connecting nodes $i$ and $j$ are thus given by $d_{ij} = \sum_{n=i}^{j} \epsilon^2(n)$. Each arc $(i, j)$ in $A$ represents the possibility

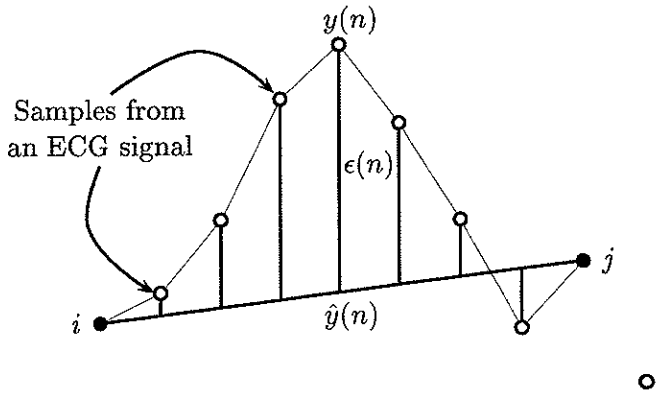


Figure 3.2: Arc length in the graph.

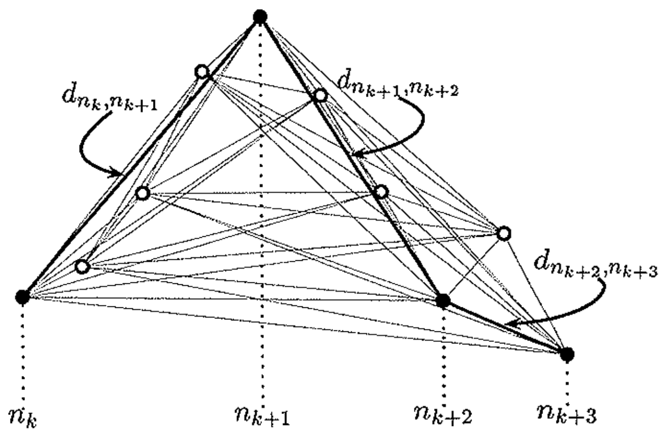


Figure 3.3: Path length in the graph.

of letting $i$ and $j$ be consecutive members of $C$. The length of $P_n$ will thus be the sum of the length of all arcs included in the path up to node $n$. The length of a given path $P$ is given by $D = \|P\| = \sum_{(i,j)\in P} d_{ij}$. This is illustrated for a linear interpolation case in Figure 3.3, where the path length from node $i = n_k$ to $j = n_{k+3}$, $d_{n_k n_{k+3}}$ equals $d_{n_k n_{k+1}} + d_{n_{k+1} n_{k+2}} + d_{n_{k+2} n_{k+3}}$.

Hence we are faced with the following problem : Minimize the length of $P_N$

under the constraint that $P_N$ contains no more than $M$ vertices, that is

$$\min_{P_N \subseteq A} \|P_N\| \quad \text{subject to } |P_N| \leq M. \tag{3.4}$$

We solve this problem by the dynamic programming algorithm described in Section 3.4.

## 3.4 Shortest path method

A dynamic programming algorithm solves the problem described in the previous section. The algorithm was originally developed by Haugland et al. [37]. However, since this algorithm is the heart of our compression scheme and since much of the work done in this dissertation has its offspring in this algorithm, its original version is included here.

In order to describe a solution scheme, we propose the following precise problem formulation. Define $P_{j,m}$ as the shortest path to $j$ visiting *exactly* $m$ vertices, and let $l(j,m)$ denote the length of $P_{j,m}$. We actually search for a $1 < m^* \leq M$ and the corresponding $P_{N,m^*}$ for which $l(N,m^*) = \min_{1<m\leq M} l(N,m)$. In our search, we compute all such paths $P_{N,m}$ in the order given by increasing values of $m$. The CCSP problem is hence solved when these quantities become available.

Consider the path $P_{j,m+1}$, and denote the second last vertex in $P_{j,m+1}$ by $i$. Obviously, $i < j$ and $i \geq m$. Hence $P_{j,m+1}$ contains a sub-path through $m$ vertices to $i$. But this sub-path has to be $P_{i,m}$, because otherwise we could find a shorter path through $m$ vertices to $i$. Augmenting this with vertex $j$ yields a path shorter than $P_{j,m+1}$ (through $m+1$ vertices) ending at $j$, contradicting the fact that $P_{j,m+1}$ is the shortest of all such paths.

Hence we have that $l(j,m+1) = l(i,m) + d_{ij}$ for some $i = m, m+1, \dots, j-1$. Furthermore, it is clear that $i$ must be the vertex *minimizing* the right hand side in this equation. Supplying the obvious condition that $l(j,2) = d_{1j}$, we arrive at the recursive equations

$$l(j,2) = d_{1j}, \tag{3.5}$$

$$l(j,m+1) = \min\{l(i,m) + d_{ij} \,|\, i = m, \dots, j-1\}. \tag{3.6}$$

When $j = 2, \dots, n$ and $m = 2, \dots, M-1$ are inserted in (3.5)-(3.6), $l(N,M)$ is uniquely determined.

Equations (3.5) - (3.6) constitute a *dynamic programming* formulation of CCSP. Similar formulations have been proposed for various constrained shortest path problems [14, 80, 82]. The formulation above resembles the one given in [82], but in (3.6) we exploit the fact that $(i,j) \in A$ only if $j > i$. Furthermore, we disregard $l(i,m)$ for all $i < m$ since all paths terminating at $i$ have at most $i$ vertices.

From the above formulation, the algorithm in Figure 3.4 suggests itself ($p(j,m)$ signifies the predecessor of $j$ in $P_{j,m}$). The compression set can thus be recorded by letting $n_{m^*} = N$ and $n_{m-1} = p(n_m, m)$ $(m = m^*, m^*-1, \dots, 2)$. This produces the interpolation points $(n_1, y(n_1)), (n_2, y(n_2)), \dots, (n_{m^*}, y(n_{m^*}))$.

```
Algorithm CCSP
for $j = 2, \dots, N$
begin
   $l(j,2) = d_{1j}$                        // Length of two-vertex path
   $p(j,2) = 1$                             // from 1 to $j$
end
$m^* = 2$                                   // Assume the two-vertex path
                                            // to $N$ is optimal
for $m = 2, \dots, M-1$                     // Find $m+1$-vertex paths
begin
   for $j = m+1, \dots, N$                  // Find the path to $j$
   begin
      $p(j, m+1) = m$                       // Assume $P_{j,m+1} = \{1, 2, \dots, m, j\}$
      $l(j, m+1) = l(m,m) + d_{m,j}$        // The length of this path
      for $i = m+1, m+2, \dots, j-1$        // $P_{j,m+1}$ may equal $P_{i,m} \cup \{j\}$
      begin
         if $l(i,m) + d_{ij} < l(j, m+1)$   // Shorter!
         begin
            $l(j, m+1) = l(i,m) + d_{ij}$   // Update the shortest length
            $p(j, m+1) = i$                 // Record the last step
         end
      end
      if $l(N, m+1) < l(N, m^*)$            // Shortest path to $N$ so far
      begin
         $m^* = m + 1$                      // Optimal number of vertices
                                            // in path to $N$
      end
   end
end
```

Figure 3.4: Algorithm for the CCSP problem.

## 3.5 Computational complexity

From the algorithm in Figure 3.4 it is seen that when all arc lengths are available, the computation of the shortest path involves $\mathcal{O}(MN^2)$ arithmetic operations. The computational complexity of the total algorithm will thus be dependent on how fast we can compute all the arc lengths. Computation of the arc lengths will, in turn, depend upon which model is applied in description of the arcs and which error measure we apply.

As the CCSP algorithm has a computational complexity of $\mathcal{O}(MN^2)$, it is of crucial importance that computation of all the arc lengths can be done without increasing the complexity any further. As N grows, there is a big difference between $\mathcal{O}(N^2)$ and $\mathcal{O}(N^3)$. When computing the arc lengths for the different piecewise approximations for the CCSP algorithm in the succeeding chapters, we will focus on doing this in a computational effective manner in order to keep the computational complexity as low as possible.

## 3.6 Summary

In this chapter, the ECG signal compression problem is defined in strict mathematical terms. It is shown how the problem may be mapped into a graph formulation. A solution algorithm based upon a shortest path algorithm, the Cardinality Constrained Shortest Path method, is introduced and the complexity of this approach is discussed. A crucial point in the CCSP algorithm is the computation of the arc lengths and this will be one of our main concerns in the succeeding chapters.

# Chapter 4

# Piecewise polynomial interpolating compression

With an interpolating approximation we insist on equality between the original and the reconstructed signal at the points of extraction. An example of this is illustrated in Figure 3.1. Between the extracted sample points, we apply different functions, as described in Section 3.2. Hence we arrive at a continuous function which interpolates $\{(n, y(n)) \,|\, n \in C\}$.

In this chapter we will investigate how different functions and error measures can be incorporated into the CCSP algorithm described in Chapter 3 applying an interpolating approximation. In section 4.1 we apply a first order polynomial approximation. In section 4.2 we incorporate two error measures, the maximum error and the sum of squared errors, into the CCSP algorithm. Section 4.3 is devoted to the development of an approach based on piecewise second order polynomial interpolation. We investigate the performance of these compression schemes applied to ECG signals.

A crucial point in the development of the different compression schemes is the investigation of the computational complexity of the algorithm. As more complex polynomials for the arcs are applied, we have to be able to do the computation of all the arc lengths in a way that ensures the lowest computational complexity possible.

## 4.1 First order polynomial compression

With first order polynomial interpolation, we apply straight lines between the extracted samples as shown in Figure 4.1. Remember that given the

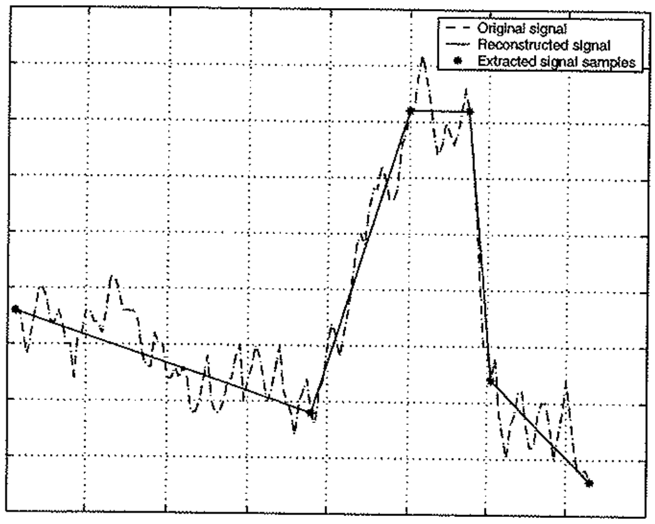


Figure 4.1: Example of short segment of original and reconstructed signal with linear interpolation applied.

signal samples, $y(1), y(2), ..., y(N)$, we seek an appropriate compression set $C = \{n_1, n_2, ..., n_M\} \subseteq \{1, 2, ..., N\}$ and the corresponding sample values. Assume $n_1 = 1$ and $n_M = N$ and let $i$ and $j$ be consecutive members of $C$. The approximation is then given by

$$\hat{y}(n) = \begin{cases} y(n) & \text{if } n \in C, \\ y(i) + \frac{y(j)-y(i)}{j-i}(n-i) & \text{if } i < n < j, \forall n \notin C, n \in 1, 2, \dots, N. \end{cases} \tag{4.1}$$

This way we get a piecewise approximation to the original signal based on linear interpolation.

We are searching for a path, $P_N$, from vertex 1 to vertex $N$ through the graph, $G = (V, A)$ as defined in Section 3.3. The length of each arc in the graph is given by

$$d_{ij} = \sum_{n=i+1}^{j-1} (\hat{y}(n) - y(n))^2. \tag{4.2}$$

What we are seeking is to minimize the length of $P_N$, i.e., $D = \|P_N\| = \sum_{(i,j)\in P_N} d_{ij}$, under the constraints that $P_N$ contains no more than $M$ vertices. When all the arc lengths are available this is solved with the shortest path method described in Chapter 3.4.

### 4.1.1 Computing the arc lengths

In order to keep the computational complexity low, we have to compute all the arc lengths in a careful manner. The graph $G$ consists of $\frac{N(N-1)}{2}$ arcs. This means that we have $\mathcal{O}(N^2)$ number of arcs for which we need to compute the length. From Equation (4.2) it is seen that the expression for $d_{ij}$ is a sum of $j-i-1$ terms, a number of order $N$. This means that straightforward computation of all these arc lengths will result in an algorithm with complexity $\mathcal{O}(N^3)$. Fortunately, this can be avoided by careful computation of the arc lengths.

The arc lengths are given by Equation (4.2). By substituting $\hat{y}(n)$ in Equation (4.2) by the expression given in Equation (4.1), we get

$$\begin{aligned} d_{ij} &= \sum_{n=i+1}^{j-1}\left(y(i)+\frac{y(j)-y(i)}{j-i}(n-i)-y(n)\right)^2 \\ &= \sum_{n=i+1}^{j-1} y(n)^2 - 2\frac{y(j)-y(i)}{j-i}\sum_{n=i+1}^{j-1} ny(n) + 2\left(\frac{i(y(j)-y(i))}{j-i}\right. \\ &\quad \left.-y(i)\right)\sum_{n=i+1}^{j-1} y(n) + \frac{(y(j)-y(i))^2}{(j-i)^2}\sum_{n=i+1}^{j-1} n^2 + 2\frac{y(j)-y(i)}{j-i}\left(y(i)\right. \\ &\quad \left.-\frac{i(y(j)-y(i))}{j-i}\right)\sum_{n=i+1}^{j-1} n + (j-i-1)\left(y(i)-\frac{i(y(j)-y(i))}{j-i}\right)^2 \end{aligned} \tag{4.3}$$

By expressing the terms including sums of powers of $n$ on closed forms (see Appendix B) we get

$$d_{ij} = \sum_{n=i+1}^{j-1} y(n)^2 + \alpha_{ij}\sum_{n=i+1}^{j-1} ny(n) + \beta_{ij}\sum_{n=i+1}^{j-1} y(n) + \eta_{ij}, \tag{4.4}$$

where

$$\begin{aligned} \alpha_{ij} &= -2\frac{y(j)-y(i)}{j-i}, \\ \beta_{ij} &= 2\left(\frac{i(y(j)-y(i))}{j-i} - y(i)\right), \end{aligned}$$

$$\begin{aligned}\eta_{ij} = {} & \frac{(y(j)-y(i))^2}{(j-i)^2}\frac{(j-i-1)(2j^2-j+2ij+i+2i^2)}{6} \\ & +2\frac{y(j)-y(i)}{j-i}\left(y(i)-\frac{i(y(j)-y(i))}{j-i}\right)\frac{(i+j)(j-i-1)}{2} \\ & +(j-i-1)\left(y(i)-\frac{i(y(j)-y(i))}{j-i}\right)^2.\end{aligned}$$

Each of the coefficients $\alpha_{ij}$, $\beta_{ij}$ and $\eta_{ij}$ depends only on $i$, $j$, $y(i)$ and $y(j)$, and computing all of them can be accomplished by $\mathcal{O}(N^2)$ operations. By defining $\Delta^1_{pj} = \sum_{n=1}^{j} n^p y(n), p = 0, 1$ and $\Delta^2_j = \sum_{n=1}^{j} y^2(n)$ we see that $\Delta^1_{p1}, \ldots, \Delta^1_{pN}$ and $\Delta^2_1, \ldots, \Delta^2_N$ are computed in $\mathcal{O}(N^2)$ time. We then compute

$$d_{ij} = \alpha_{ij}(\Delta^1_{0,j-1} - \Delta^1_{0i}) + \beta_{ij}(\Delta^1_{1,j-1} - \Delta^1_{1i}) + \Delta^2_{j-1} - \Delta^2_i + \eta_{ij}, \tag{4.5}$$

$1 < j < N$, involving $\mathcal{O}(N^2)$ operations. Hence, all the arc lengths $d_{ij}$ are available in $\mathcal{O}(N^2)$ time.

### 4.1.2 Coding scheme

Reconstruction of a signal coded by a linear interpolation time domain method requires two parameters for each retained sample of the signal; the amplitude and the position. Recall that the amplitudes of the signal samples extracted by the time domain coder are denoted $y(n_k)$, $k = 1, \ldots, M$ where $n_k$ is the sample index, corresponding to position.

We apply a simple predictive encoding scheme and encode the first order difference of both parameters (first order DPCM), that is, each segment of the signal is represented by the two parameters $\delta_{y(k)} = Q(y(n_k)) - Q(y(n_{k-1}))$ and $\delta_{n(k)} = n_k - n_{k-1}, k = 2, 3, \ldots, M$, where $Q$ denotes quantization. In addition, we need to encode the absolute amplitude of the first point, $y(n_1)$.

We thus have a pair of $(\delta_{y(k)}, \delta_{n(k)})$ to be encoded for each segment of the signal. Together these constitute a natural source word to be encoded. As the probabilities of occurrence varies for the different symbols at different bit rates, it is natural to employ a Variable Length Coder (VLC) scheme in encoding of the extracted signal samples. This encoder maps its input source signal into codewords of variable lengths. Compression is achieved by assigning short codewords to input symbols of high probability and longer codewords to the input symbols of low probability. We thus get a lossless compression of the symbols.

We have at least two possibilities when choosing between coding strategies:

1. Encoding of the symbols $\delta_{y(k)}$ and $\delta_{n(k)}$ by two separate encoders.
2. Encoding of the concatenated symbols $(\delta_{y(k)}, \delta_{n(k)})$ by one single encoder.

Alternative 2 implies a high number of possible source symbols. If we assume that the test signals in Section 4.1.3 results in a $\delta_{y(k)}$ with a dynamic range of 512 and that no run is longer than 256, we arrive at $512 \cdot 256 = 131072$ possible different source symbols for alternative 2, as opposed to $512 + 256 = 768$ different possibilities for alternative 1. Experiments show that in the case of alternative 2, only a small fraction of the possible source symbols are actually used, but nevertheless, this alternative will result in a much higher number of source symbols than alternative 1. This indicates that if we are to transmit the VLC tables[1], alternative 1 will result in the most efficient encoding scheme. However, by having an indexed set of VLC tables in both encoder and decoder we can get around this problem. By choosing alternative 2 we can utilize the correlation that exists between $\delta_{y(k)}$ and $\delta_{n(k)}$. This correlation originates from the fact that in high frequency regions of the signal, we will extract more points. We thus have many $\delta_{n(k)}$, each of which has a low numerical value, accompanied by $\delta_{y(k)}$ with a high dynamic range due to the abrupt changes in the waveform. In low frequency regions of the signal it is the other way around: We can represent the signal with fewer points, and will thus have fewer $\delta_{n(k)}$ with a higher dynamic range than in the high frequency case, accompanied by $\delta_{y(k)}$ with a lower dynamic range due to small variations in amplitude. However, experiments show that there is only a marginal gain obtained by applying alternative 2 as opposed to alternative 1 in this context. For these reasons we choose to use alternative 1 and encode $\delta_{y(k)}$ and $\delta_{n(k)}$ by two separate coders. Note however, that alternative 2 is viable in our subband coders, see Section 2.3.1, and in some of our coders where we apply other arc models, see Section 4.3.3. The structure of the encoding system in this case where we apply linear interpolation is as illustrated in Figure 4.2. The two VLC's are the variable length coders, Q is the quantizer and MUX is the multiplexer used to combine the two bit streams into one for storage or transmission.

The original signal samples are represented with a resolution of 12 bits per sample. With the dynamic range of the ECG test signals used in this dissertation, this corresponds to a quantization of the amplitudes, $y(n)$, $n = 1, \dots, N$ using a uniform quantizer. Let us denote the step size of the original quantizer by $\Delta_Q$. The extracted signal samples, $y(n_k)$, $k = 1, \dots, M$, will thus be

[1]Tables indicating the mapping of source symbols into codewords.

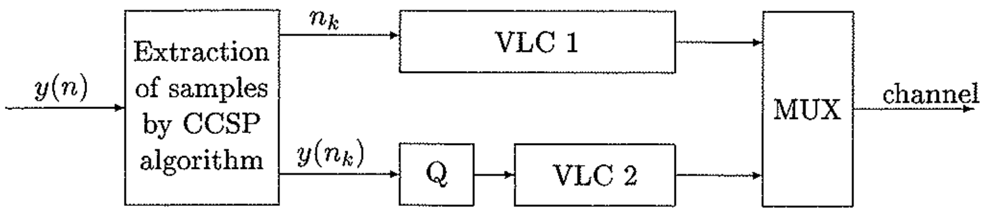


Figure 4.2: Structure of encoding system in the linear interpolating case.

represented in the same way. By applying a quantizer with a different step size than $\Delta_Q$ to these extracted signal samples we may be able to lower the rate-distortion curves. This is shown for test signal mit100_1000 in Figure 4.3 where we report results from the complete coding scheme, i.e., both approximation of the original signal and encoding of the extracted signal samples, for quantizers of different step size. We see that by increasing the quantization step from size $\Delta_Q$ to $2\Delta_Q$ and from $2\Delta_Q$ and $3\Delta_Q$, the overall rate-distortion performance is improved, for the area of bit rates below 1.8 bits per sample. However, when the quantizer step size is increased beyond $3\Delta_Q$, we reduce the performance in some areas while we gain in others. The reason for this is that at low bit rates, we have few extracted samples and thus we introduce a smaller error by quantizing these samples than we do for high bit rates where there are more extracted samples. Which quantization step gives the overall best performance is dependent on the target bit rate, the signal to be compressed and the original sampling rate. We have chosen to use a uniform quantizer with a step size of $3\Delta_Q$ in the experimental results from the different versions of the CCSP algorithm and the FAN algorithm, except in the experimental results in Section 4.2.3 where we apply a uniform quantizer with a step size of $\Delta_Q$. The results from the complete coding system are presented and discussed in Section 4.1.3.

### 4.1.3 Numerical experiments and discussion

For quantitative evaluation of the performance of the coders, the commonly used *Percentage Root-mean-square Difference* distortion measure, given by Equation (2.3) is applied in addition to the maximum error, given by Equation (2.1). We evaluate these error measures as a function of bit rate.

The two error performance measures are useful for testing the relative performance of the various ECG coding techniques. However, as each compression

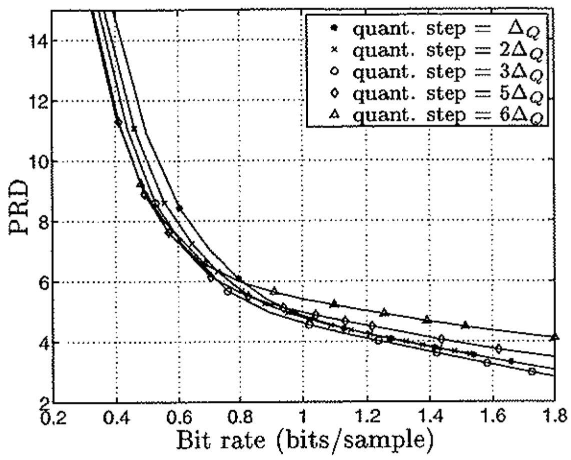


Figure 4.3: PRD versus bit rate for test signal mit100_1000 with uniform quantizers with different step sizes.

method has its own distortion characteristics, the quantitative evaluation measures should be supplemented by visual inspection of the reconstructed signal. We will also report some benchmarks, i.e., the execution time of the different algorithms on a given machine. This is to evaluate the real time performance of the coders.

Several recordings taken from the MIT/BIH Arrythmia CD-ROM database, second edition [60], were used in the coding experiments. The first 10 seconds of the test signals we use are plotted in Appendix A.

The test signals are coded using two different compression algorithms: The CCSP algorithm based on linear interpolation presented in this chapter, and the FAN algorithm, a traditional time domain algorithm presented in Section 2.2.1. In comparison with other time domain coders, the FAN algorithm has been reported to give high compression ratios, in addition to producing reconstructed signals with high fidelity [46], and it can thus be viewed as a reference case for the whole class of traditional heuristic time domain ECG compression schemes.

The complexity of the CCSP algorithm based on linear interpolation is $\mathcal{O}(MN^2)$ where $N$ is the number of original signal samples and $M$ is the upper bound on the number of extracted samples. To keep the execution time down, the input signal is processed in blocks of 500 samples when extracting samples by the CCSP algorithm. The total record of extracted signal samples and their corresponding positions are then encoded by a VLC as described in Section 4.1.2.

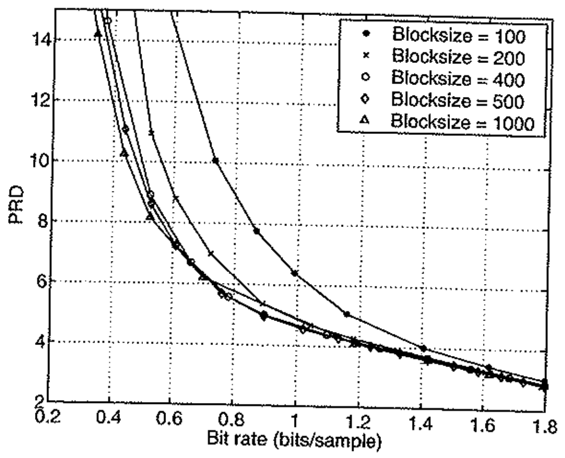


Figure 4.4: PRD versus bit rate for test signal mit100_10000 evaluated for different block sizes.

The choice of block size is based upon a tradeoff between optimality of solution and execution time. Naturally, the shorter the blocks, the faster the algorithm. However, short blocks will lead to loss of optimality due to the fact that the end points of each block are always extracted as significant signal samples. From Figure 4.4 we can see PRD versus bit rate for different block sizes for test signal mit100_1000. We see that as the block size increases from 100 to 200, the performance increases significantly in terms of PRD. As the block size is further increased, the performance also increases further, but the difference between rate-distortion curves are smaller for larger block sizes. There is little difference between the curve obtained with a block size of 500 and the one obtained with a block size of 1000. For this reason we have chosen to use a block size of 500.

To achieve an even more efficient implementation of the coder, the method described in [38] can be applied. By dividing the signal into short blocks (i.e., 100 or 150 original signal samples) before extraction of samples, the CCSP algorithm satisfies real time requirements as is shown in Table 4.1 on page 48. As stated above, short blocks will lead to loss of optimality. Presenting an effective implementation of the CCSP algorithm, the method in [38] accounts for this. The idea is to divide the input signal into partly overlapping segments, and process each segment independently. It is shown that an optimal compression algorithm can be implemented by a windowing technique, processing 100 samples at a time with only slightly reduced coding performance.

We compare the results from the FAN algorithm to the CCSP technique. The samples extracted by the FAN algorithm are encoded in the same manner as the samples extracted by the CCSP algorithm based on linear interpolation, see Figure 4.2.

### Evaluation based on the PRD measure

Figure 4.5 presents obtained PRD's for the coders for bit rates between 0.2 and 1.8 bits per sample (bps).

For all test signals, the FAN method is outperformed by the CCSP method by a wide margin. At low bit rates (around 0.6 bps) the FAN algorithm has from 20% to 130 % higher PRD than the CCSP algorithm based on linear interpolating approximation. At higher rates (around 1.0 bps) the difference is smaller, but still significant.

Generally, the test signals mit203_1100 are the hardest one to compress using time domain algorithms. This is due to the fact that this is a more rapidly varying signal, or in other words, it contains more high frequency components than the other test signals. This can be seen from the power density spectrum plots of the test signals in Figure A.2 in Appendix A.

Side information, i.e., the overhead necessary due to transmission or storage of the VLC table, has not been taken into account in the rate-distortion curves of Figure 4.5. The amount of side information will be approximately the same for both the CCSP and the FAN coder as the extracted samples are encoded in the same way. Experiments show that side information will add less than 9% to the bit rate shown in Figure 4.5[2]. Side information constitutes a bigger part for low bit rates, and is decreasing as the bit rate increases.

### Evaluation based on the maximum error

Figure 4.7 presents obtained maximum errors for the coders. The CCSP based on linear interpolation is outperformed by the FAN method for all test signals and all bit rates. It is expected that FAN would perform better than the CCSP method in terms of maximum error, as this error measure is bounded in the FAN algorithm. But what happens in the CCSP algorithm causing these big maximum errors? The answer is illustrated in Figure 4.6, showing a short segment of test signal mit203_0100, both original and reconstructed signal, at a bit rate of 1 bps. We see that in areas of high frequency activity, the CCSP algorithm cannot "afford" to include all peaks in the solution as

[2]This is valid under the assumption that $\delta_{y(k)}$ and $\delta_{n(k)}$ are encoded by separate VLC coders and that a quantizer with step size of $3\Delta_Q$ is applied.

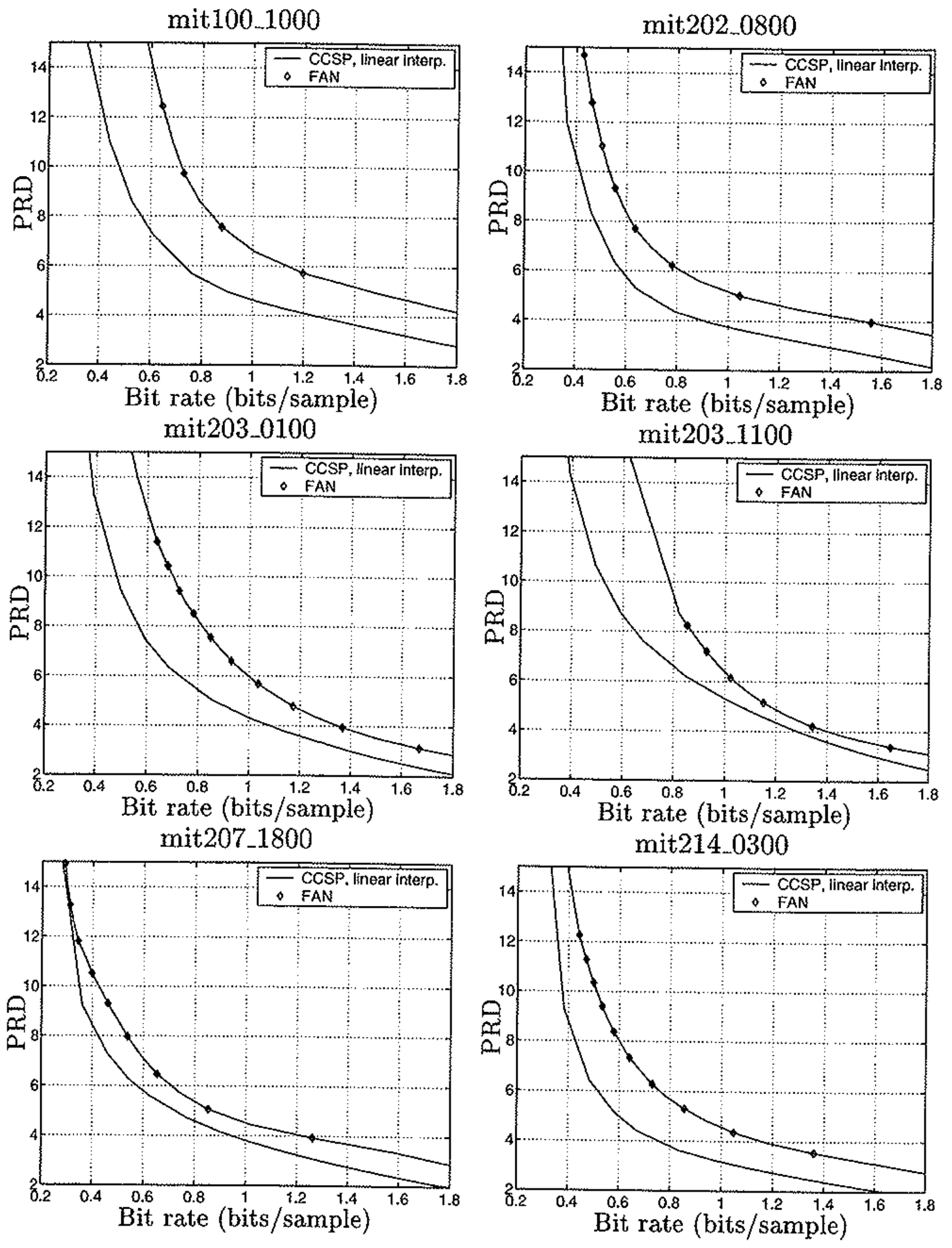


Figure 4.5: PRD versus bit rate for the different coders and test signals. Solid line: CCSP linear interpolation. Solid line with diamonds: FAN.

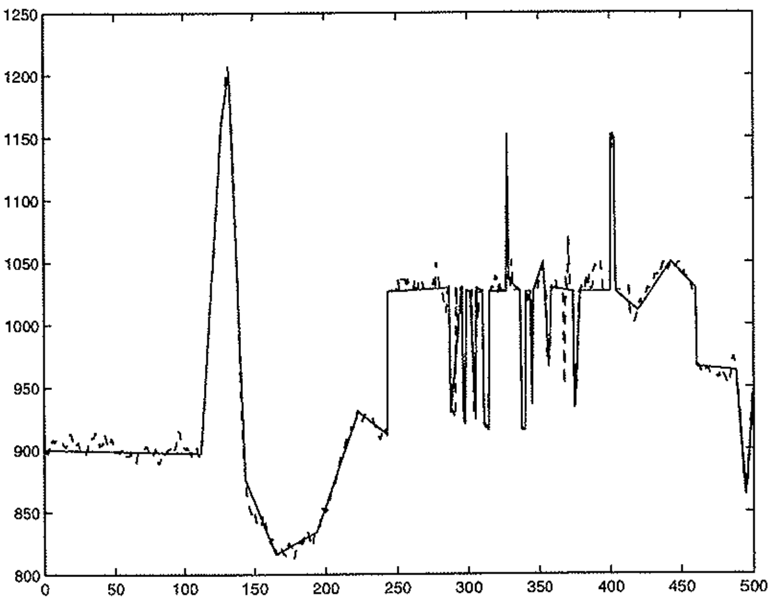


Figure 4.6: Short segment of test signal mit203_0100, decoded at a bit rate of 1 bps. Dotted line: Original signal. Solid line: Reconstructed signal.

it is only allowed to extract a defined number of samples from each block. Thus the peaks resulting in the lowest overall cost will be excluded from the solution, causing the maximum error to rise. One such peak excluded from the solution in a long signal sequence will cause a high maximum error. One possible way to account for this is to allow the CCSP algorithm to extract an unequal number of samples from each block of the signal. Classifying the signal in high- and low-frequency blocks, we can extract more samples from the high-frequency blocks and thus account for the high maximum error.

Another way to account for the high maximum error is by incorporating the maximum error into the CCSP algorithm as is done in Section 4.2. By restricting how big the maximum error is allowed to be, we can tune the algorithm to a suitable trade-off between PRD, maximum error and bit rate. This approach offers control over both the maximum error and the sum of squared errors.

### Evaluation based on visual inspection

Evaluation of the performance of the different coders should be accompanied by visual inspection of the reconstructed signals. This is to show coding artifacts as they appear for the different coders. We have chosen a short segment of

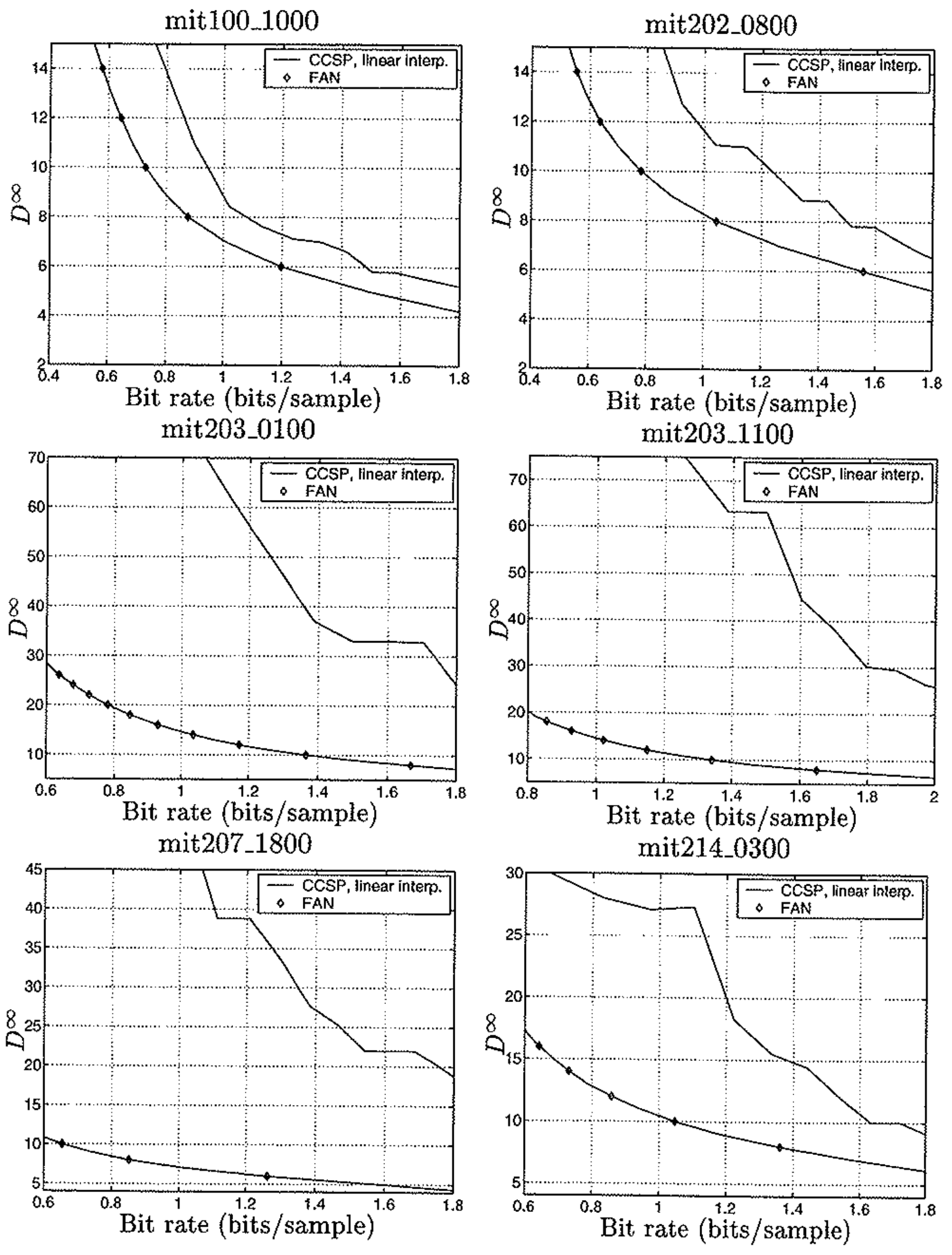


Figure 4.7: Maximum error, $D^\infty$, versus bit rate for the different coders and test signals. Solid line: CCSP linear interpolation. Solid line with diamonds: FAN.

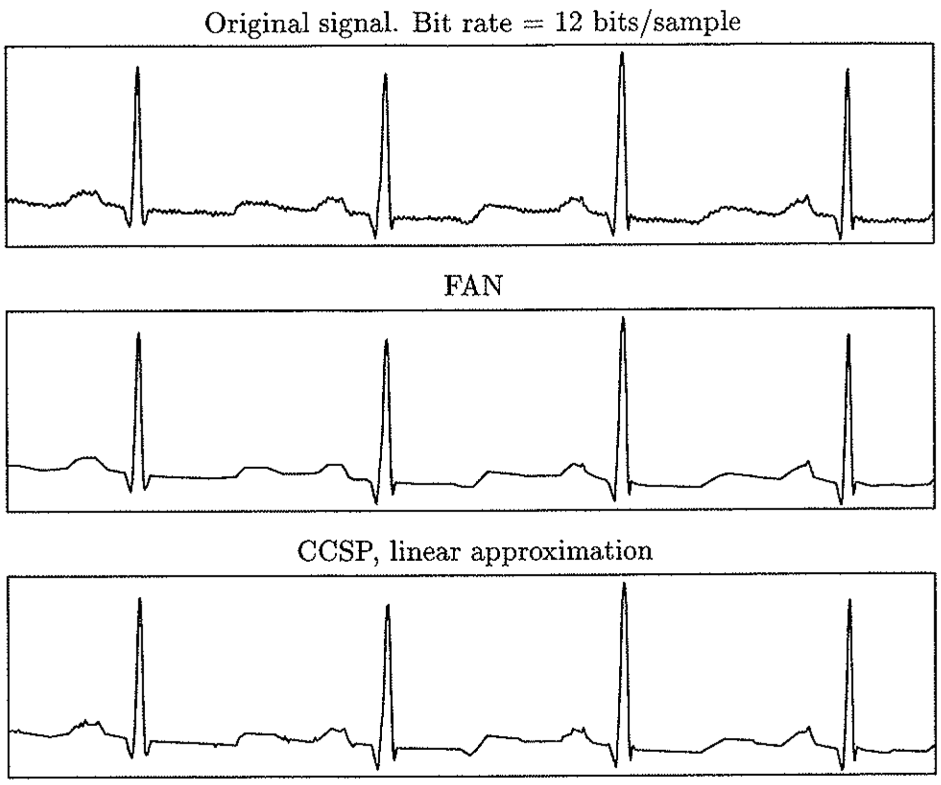


Figure 4.8: Short segment of reconstructed signal (taken from mit100_1000) at 1.0 bits per sample.

the mit100_1000 signal representing regular sinus rhythm. The reconstructed signal segment is shown at a bit rate of 1.0 bit per sample in Figure 4.8 and 0.5 bits per sample in Figure 4.9. The original signal is also included. We see that both coders smooth out some of the details in the original signal. This is particularly evident with the FAN coder, where the line pieces are also most prominent in the reconstructed signal.

Both the CCSP and the FAN method smooth out the ripple noise which can be seen in the original signal. This is undesirable noise, and will often be removed by a filter before transmission or storage of the ECG signal.

### Evaluation based on execution time

In real life, ECG compression algorithms will often be implemented on small hand-held devices, like an ECG recorder, or on portable devices like the defib-

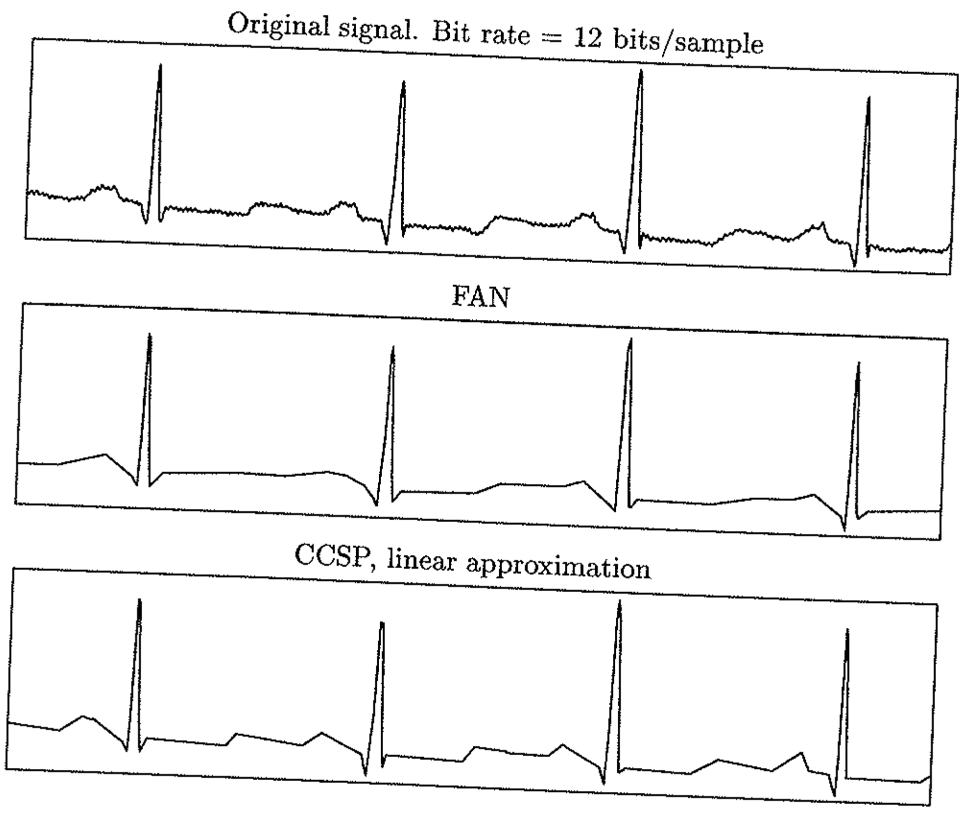


Figure 4.9: Short segment of reconstructed signal (taken from mit100_1000) at 0.5 bits per sample.

rillator[3]. In these cases it is important that the compression algorithm runs in real time, i.e., that it is able to analyze the signal *as it comes in* without causing the incoming signal to accumulate.

Table 4.1 shows execution times for the CCSP algorithm based on linear interpolation for different block sizes and different number of extracted samples. The experiments are run on an HP9000 C360 work station with a 367 MHz processor and the test signal is extracted from the beginning of mit100_1000. The execution times are based on the average of 10 runs. We have extracted different number of samples for the different block sizes, in order to cover approximately the same area of the rate-distortion curves shown in Figure 4.5.

From Table 4.1 it can be seen that for the CCSP algorithm based on linear

[3]An electronic apparatus used to counteract atrial or ventricular fibrillation by the application of brief electroshock to the heart, either directly or through electrodes placed on the chest wall.

interpolation with a sampling frequency of 360 Hz and block sizes of 100 and 200, the real time requirements are fulfilled for all values of the number of extracted samples, $M$. For larger block sizes, real time requirements are violated as $M$ grows. In order to cope with real time requirements, the technique presented in [38] can be implemented. In this case, the CCSP based on linear interpolation will run in real time on an HP9000 C360.

As for the FAN algorithm, it processes one signal sample at a time as it comes in and will thus fulfill real time requirements for all cases.

As the CCSP algorithm exploits all spare time to computations, it has an advantage over the FAN algorithm with respect to execution time. Depending on processor power available or optimization of the code with respect to execution time, block size can be matched to utilize the processor capacity. This will result in a reconstructed signal with even higher fidelity. As for the FAN algorithm this is not an issue, it will spend the time available waiting for the next signal sample.

## 4.2 Multiple error measures in ECG signal compression

As stated in Section 2.1.1, two different error measures have traditionally been applied to the compression of ECG signals: The maximum error and the sum of squared errors given in Equations (2.1) and (2.2), respectively. Which error measure gives better result is an open problem. When considering the sum of squared error, we assure that the overall error will be as small as possible, however, we do not guarantee anything about the error at each point of the signal.

Instead of choosing between the maximum error and the sum of squared error, we have developed an approach where we incorporate both error measures into one compression algorithm. An algorithm for solving the compression problem in the case of sum of squared error is given in Chapter 3. It is shown that the algorithm converges in cubic time, and in [38] it is demonstrated how the idea can be implemented in order to comply with the constraints that apply to execution time. Unlike heuristics like [13] and [22], which both are guided by a bound on the maximum reproduction error, the exact CCSP method has so far been unable to deal with such a bound. In the following sections we show how this can be overcome.

| **CCSP, linear interpolation** | | |
|---|---|---|
| Block size | Extracted samples, M | Execution time in sec. |
| 100 | 5 | 0.02 |
| | 10 | 0.03 |
| | 30 | 0.08 |
| 200 | 5 | 0.06 |
| | 10 | 0.11 |
| | 30 | 0.33 |
| | 50 | 0.51 |
| 300 | 5 | 0.11 |
| | 10 | 0.25 |
| | 30 | 0.73 |
| | 50 | 1.16 |
| | 70 | 1.52 |
| 400 | 5 | 0.18 |
| | 10 | 0.41 |
| | 30 | 1.30 |
| | 50 | 2.08 |
| | 70 | 2.84 |
| | 90 | 3.44 |
| 500 | 5 | 0.30 |
| | 10 | 0.66 |
| | 30 | 2.04 |
| | 50 | 3.41 |
| | 70 | 4.69 |
| | 90 | 5.94 |

Table 4.1: Execution times for the CCSP algorithm based on linear interpolation run on an HP9000 C360.

### 4.2.1 Problem definition

As before, let $N$ denote the total number of samples, and let $M$ be an upper bound on the number of extracted samples. Let $y(i)$ still denote the amplitude of sample $i$. Define

$$\epsilon_{nij} = y(i) + \frac{y(j) - y(i)}{j - i}(n - i) - y(n) \quad \text{for all } i \leq n \leq j. \tag{4.6}$$

With each arc $(i,j)$ we associate the *length parameters*

$$d_{ij} = \sum_{n=i}^{j} \epsilon_{nij}^2, \tag{4.7}$$

$$d_{ij}^{\infty} = \max\left\{|\epsilon_{nij}| : i \leq n \leq j\right\}. \tag{4.8}$$

Correspondingly, we define the *path lengths*

$$D = \sum_{(i,j)\in P} d_{ij}, \tag{4.9}$$

$$D^{\infty} = \max\left\{d_{ij}^{\infty} : (i,j) \in P\right\}, \tag{4.10}$$

where $P$ is the set (path) of pairs (arcs) of consecutive retained samples. Let $|P|$ denote the cardinality of $P$.

The set $P$ defines the approximation uniquely, and we define the restored signal values $\hat{y}(1), \ldots, \hat{y}(N)$ by letting $\hat{y}(n) = y(n)$ if sample $n$ is retained. Otherwise, we let

$$\hat{y}(n) = y(i) + \frac{y(j) - y(i)}{j - i}(n - i), \tag{4.11}$$

where $i$ and $j$ are the consecutive retained samples closest to $n$, for which $i < n < j$. Hence $\epsilon_{nij}$ signifies the local error introduced when replacing $y(n)$ by its approximation $\hat{y}(n)$. It is required that the absolute value of the difference between original and restored signal nowhere violates the upper bound $\bar{d}^{\infty}$. That is, the *maximum error* is to be bounded from above.

The problem thus amounts to extract a sample selection to be represented by $P$, satisfying $D^{\infty} \leq \bar{d}^{\infty}$ and $|P| \leq m - 1$, and such that $D$ is minimized. When $\bar{d}^{\infty}$ is sufficiently large, this essentially becomes the problem addressed in Section 4.1.

### 4.2.2 Computing the infinity norm error

In [36], it is proven that only minor changes to the cardinality constrained shortest path problem are necessary in order to cope with bounds on the maximum reproduction error. Actually, this can be accomplished by reducing the graph, i.e. eliminating all arcs that would contribute to a violation of this bound. Next the algorithm in Section 3.4 can be applied to the reduced graph.

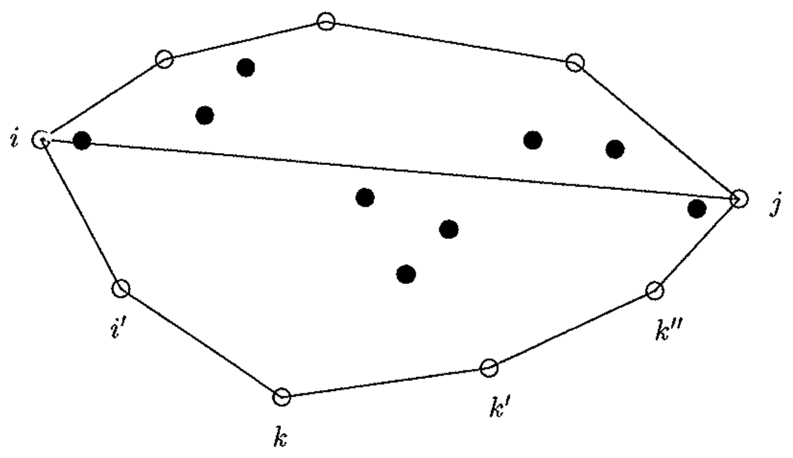


Figure 4.10: The convex hull of a set of $2D$ points.

The main effort in reducing the graph, is to compute $d_{ij}^{\infty}$. This is easily accomplished provided that the *convex hull*[4] of all sample points from $i$ to $j$ are available. By executing a *binary search*[5] along the boundary vertices of the convex hull, $d_{ij}^{\infty}$ can be computed for any combination of $i$ and $j$. In the case depicted in Figure 4.10 the maximum error occurs at vertex k.

The problem of reducing the graph can thus be solved by the following major steps:

- For each pair of vertices $i$ and $j, i = 1, \ldots, N-1, j = i+1, \ldots, N$, compute the convex hull containing all samples from $i$ to $j$.
- Compute $d_{ij}^{\infty}$ by *binary search* along the boundary vertices of the corresponding convex hull.
- Compare $d_{ij}^{\infty}$ to $\bar{d}^{\infty}$. If $d_{ij}^{\infty} > \bar{d}^{\infty}$, eliminate the corresponding arc from the graph.

An important part of the problem of constraining the maximum error, is the computation of a dynamic convex hull of a sorted set of $2D$ points. A general method for this is given in [95]. Denote the convex hull of all samples from $i$ to $j$ by $H(i,j)$. From Figure 4.10 it is clear that $H(i,j)$ can be uniquely

[4]The smallest convex set containing the points. Can be thought of as a rubber band wrapped around the extreme points.

[5]The process of examining a middle value of a sorted array to see which half contains the value in question and continuing to halve until the value is located.

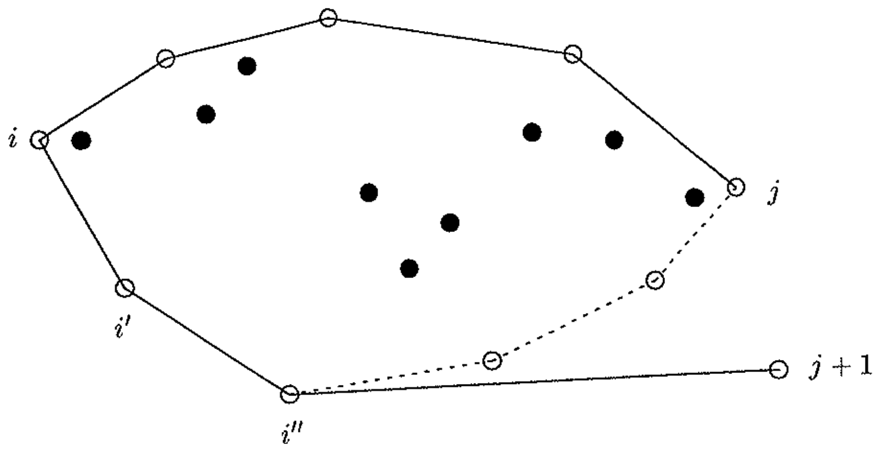


Figure 4.11: Insertion of a new point in an existing convex hull.

represented by its vertices. The boundary of $H(i,j)$ is described by one convex and one concave function, and is thus naturally sub-divided into two parts. When computing $H(1,j)$, $j = 2,\dots,N$, we verify whether sample $i$ $(1 < i < j)$ is a vertex of the convex part, a vertex of the concave part, or an interior point. Such *boundary relations* are stored in a dynamic data structure, and exploited in the computation of $H(i,j), i = 2,\dots,N-1, j = i+1,\dots,N$. In the following discussion we focus on how to find the vertices of the convex part of the boundary, but equivalent operations apply to the concave part.

Assume $H(i,j)$ is found. This hull is to be updated to $H(i,j+1)$ by taking one new sample, $j+1$, into account. We find $H(i,j+1)$ by backtracking along the boundary of $H(i,j)$, and insert the new sample in such a way that convexity is maintained as shown in Figure 4.11.

To analyze how $H(i,j+1)$ will change compared to $H(i,j)$, we must examine the slope of the straight line joining sample number $j+1$ and one vertex in $H(i,j)$. Let $\gamma_{ij}$ denote the slope of the straight line joining samples number $i$ and $j$, that is

$$\gamma_{ij} = \frac{y(j) - y(i)}{j - i}. \tag{4.12}$$

To decide where to insert sample $j+1$ in $H(i,j)$, we backtrack until we find two succeeding vertices $i' < i''$ such that $\gamma_{i'i''} \leq \gamma_{i''j+1}$, and insert sample $j+1$ after vertex $i''$ as illustrated in Figure 4.11. For the concave part it is the other way around: We backtrack the boundary vertices in $H(i,j)$ until we find two succeeding vertices $i' < i''$ such that $\gamma_{i'i''} \geq \gamma_{i''j+1}$ and insert sample $j+1$ after vertex $i''$.

Once $H(i,j), j = i+1, \ldots, N$ are found, the next step is to compute $H(i+1,j), j = i+2, \ldots, N$. We know that interior points of $H(i,j)$ may be on the boundary of $H(i+1,j)$. However, if sample $k$ is a vertex of both $H(i,j)$ and $H(i+1,j)$, then all vertices $k' > k$ of $H(i,j)$ are vertices of $H(i+1,j)$ as well. In order to be computationally efficient, the algorithm has to exploit this information. In the computation of $H(i,j), j = i+1, \ldots, N$, we therefore keep track of all boundary relations for later to be utilized in the computation of $H(i+1,j), j = i+2, \ldots, N$.

A total of $\frac{N(N-1)}{2}$ convex hulls have to be made available in order to restrict $D^\infty$. Each of these are represented by the vertices of the convex and concave part of the boundary. When the vertices are found, we make binary search in both vertex sets in order to find the points where the maximum error occurs. The binary search can be applied in this case because the line segments on the boundary are sorted by slope. By stepping through the vertices of the convex part, we will reach a point where the slope of the straight line connecting two vertices gets larger than $\gamma_{ij}$. This is the point where the maximum error occurs. This is illustrated in Figure 4.10. We see that $\gamma_{ij} > \gamma_{i'k}$ and that $\gamma_{ij} < \gamma_{kk'}$. Thus $k$ is the point of maximum error. By making binary search in both vertex sets of the boundary of $H(i,j)$, $d_{ij}^\infty$ can be found in $\mathcal{O}(\log N)$ time.

Computing all $\frac{N(N-1)}{2}$ convex hulls along the lines outlined above, requires $\mathcal{O}(N^2)$ operations [36]. Hence the graph reduction is performed in $\mathcal{O}(N^2 \log N)$ time. Furthermore, the complexity of the optimization algorithm in Chapter 3 is $\mathcal{O}(MN^2)$, where the dependency upon $N^2$ is explained by the fact that an unreduced graph consists of $\frac{N(N-1)}{2}$ arcs. If the number of arcs after reduction is $K$, the complexity of the new version of the algorithm becomes $\mathcal{O}(N^2 \log N + MK)$. Note however that in the worst case, $K = \mathcal{O}(N^2)$.

### 4.2.3 Numerical experiments and discussion

To get a quantitative evaluation of the performance of the coders, the PRD distortion measure given in Equation (2.3) is applied. We evaluate PRD as a function of bit rate for different numerical values of the maximum error bound. Despite the incorporation of the maximum error bound, we still evaluate the maximum error as a function of bit rate. This is due to the fact that the *actual* maximum error may differ from the maximum error *bound*.

We will not present a visual reconstruction of the decoded signals in this section. The reconstructed signals will be very similar to the ones presented in Section 4.1.3, so we refer the reader to this section to get a visual impression of the reconstructed signals. However, we will report some benchmarks, i.e., the execution time of the different algorithms on a given machine. This is to evaluate the real time performance of the coders.

To keep the execution time down, the input signal is processed in blocks of 500 samples when extracting samples by the CCSP algorithm. The total record of extracted signal samples are encoded in the way described in Section 4.1.2, that is, we use a simple predictive encoding scheme and encode the first order difference for position and amplitude by two separate VLC's. Thus the extracted signal samples are encoded according to Figure 4.2. Unlike the results presented in Section 4.1.3, where we use a quantizer with a step size of $3\Delta_Q$, we apply a quantizer with a step size of $\Delta_Q$ to the extracted signal samples in this section. The reason for this is that by applying a quantizer with a step size different from the original one, i.e, $\Delta_Q$, to the extracted signal samples in this case, we loose the complete control this method offers over the maximum error. By quantizing the extracted signal samples, we introduce quantization noise and thus we can no longer guarantee that the maximum error will be below the predescribed bound on the maximum error. An example of this is shown in Figure 4.12 where we show complete coding experiments of mit100_1000 with a quantizer step size of $3\Delta_Q$. We see that that maximum error is *not* below the predescribed error bounds of 4, 6 and 8, respectively, but has been increased above these due to quantization noise introduced.

The test signals applied in this section are presented in Appendix A.

### Evaluation based on the PRD measure

The PRD as a function of bit rate is shown in Figure 4.13. Input to the CCSP algorithm are $\bar{d}^{\infty}$ and $M$, while only $\bar{d}^{\infty}$ is input to the FAN algorithm. We plot several curves for the CCSP algorithm based on several maximum error bounds and evaluate this towards the FAN algorithm. The plots of Figure 4.13 shows the same tendency as was shown in Figure 4.5: The FAN algorithm is outperformed by the CCSP algorithm in terms of PRD for all bit rates and all test signals applied.

From Figure 4.13 we see that if the bounds on the bit rate and $\bar{d}^{\infty}$ are too strict, there exists no solution. This reflects the fact that given a desired compression ratio, it is not possible to get an arbitrarily small infinity norm error because there exists no path through the graph which satisfies these

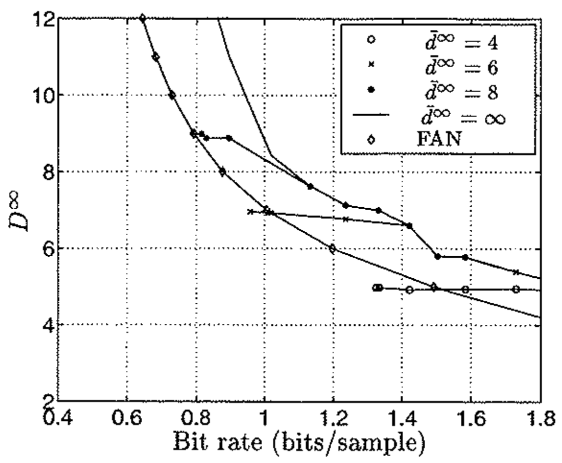


Figure 4.12: PRD versus bit rate for test signal mit100_1000 with uniform quantizers with step size of $3\Delta_Q$.

constraints. The rate-distortion curves of Figures 4.13 and 4.14 reflect this fact. The leftmost point in each of the curves reports the smallest feasible bit rate for the different $\bar{d}^\infty$. An example : For test signal mit100_1000 with $\bar{d}^\infty = 8$, the smallest possible value for the bit rate is 0.88 bps. We thus have to make a tradeoff between $\bar{d}^\infty$ and the sample reduction ratio in order to find a solution.

Restricting the sample selection by introducing $\bar{d}^\infty$ implies only a marginal increase in PRD as compared to excluding the maximum error bound. This can be seen from Figure 4.13, indicating that the CCSP algorithm based on linear interpolation still is superior to conventional time-domain heuristics in the sense that it exhibits much less PRD, especially for low bit rates.

### Evaluation based on the maximum error

Figure 4.14 shows obtained maximum errors as a function of bit rate for the different coders and test signals. For each signal we report experiments based on different maximum error bounds. We see that we are able to lower the maximum error for the CCSP algorithm by introducing $\bar{d}^\infty$. However, the FAN algorithm still generally performs better in terms of maximum error than the CCSP algorithm with an exception of a few areas for test signal mit100_1000.

The reason why we cannot lower the maximum error further for the CCSP algorithm, is that we have to pay attention to the problem of cases without

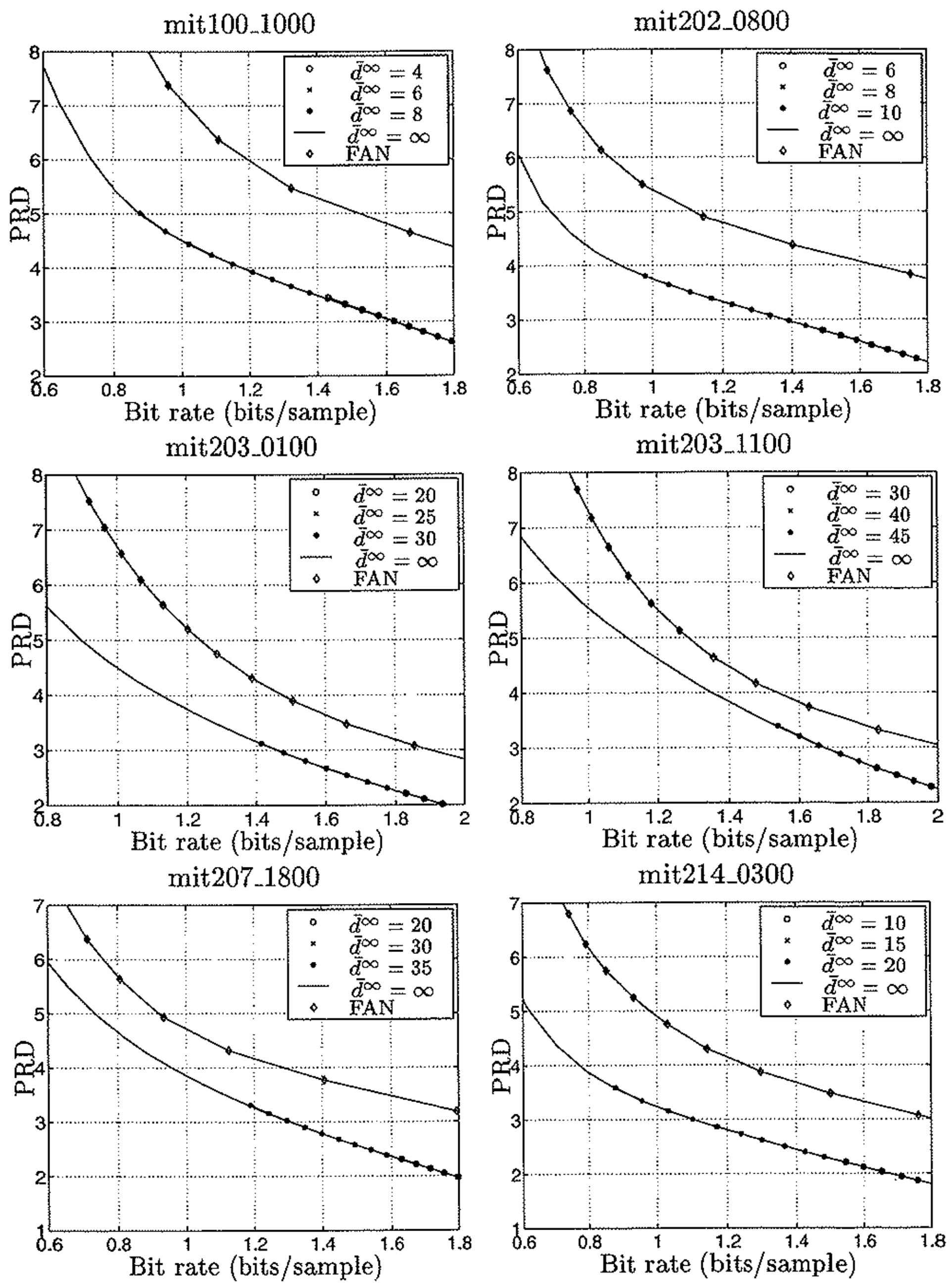


Figure 4.13: PRD versus bit rate for the different coders and test signals.

solution. The plots in Figure 4.14 show the range of bit rates for which we obtain lower maximum errors than in the case without the maximum error incoporated for the different test signals. The leftmost points in the plots reflect the smallest possible bit rate for the different values of $\bar{d}^{\infty}$. We see that as the bit rate increases for one particular $\bar{d}^{\infty}$, the problem essentially becomes the one solved in Section 4.1.

The CCSP algorithm codes the signal block by block, considering a block size of 500 samples at a time. From each block it is only allowed to extract an upper number of $M$ samples. This has severe impact on the maximum error. If one block contains much high frequency information, the CCSP algorithm will not be able to find a solution for this block with a tight bound on the maximum error. For the FAN algorithm this is not an issue as it does not processes the signal using a block-based approch, but rather considers one sample at a time. One way to lower the maximum error even further and avoid the problem of non-existing solutions when coding with the CCSP algorithm is by allowing a variable number of samples to be extracted from each block of the signal as discussed in Section 4.1.3.

### Evaluation based on execution time

Another important aspect by restricting the infinity norm error, is the total execution time. As pointed out in the previous section, the complexity of the algorithm becomes $\mathcal{O}(N^2 \log N + MK)$ when a bound on the maximum error is introduced, as opposed to $\mathcal{O}(MN^2)$ in the original version. In cases where $K \ll \frac{N(N-1)}{2}$, this may be of major importance. For the first block (the first 500 samples) of test signal mit100_1000 we found that $K$ attains values 8210, 11285, 21648 and 29189 when $\bar{d}^{\infty}$ is put equal to 5.05, 6.00, 13.10 and 16.90, respectively, whereas the unreduced graph has 124750 arcs. This implies considerable reduction factors (ranging from 8 to 30).

Table 4.2 shows execution times for the CCSP algorithm for different block sizes and different number of extracted samples. The experiments are run on the same machine with, and the same test signals as in Section 4.1.3. The first entry in the table for each combination of block size and number of extrcted samples, reflects the minimal feasible integer $\bar{d}^{\infty}$ for which a solution exists.

By comparing the numbers in Table 4.2 to the ones in Table 4.1 it can be seen that incorporating the maximum error bound into the CCSP algorithm lowers the execution time drastically for large block sizes and large number of extracted samples. For a block size of 500 with 70 samples extracted, the execution time is reduced by a factor of 5.

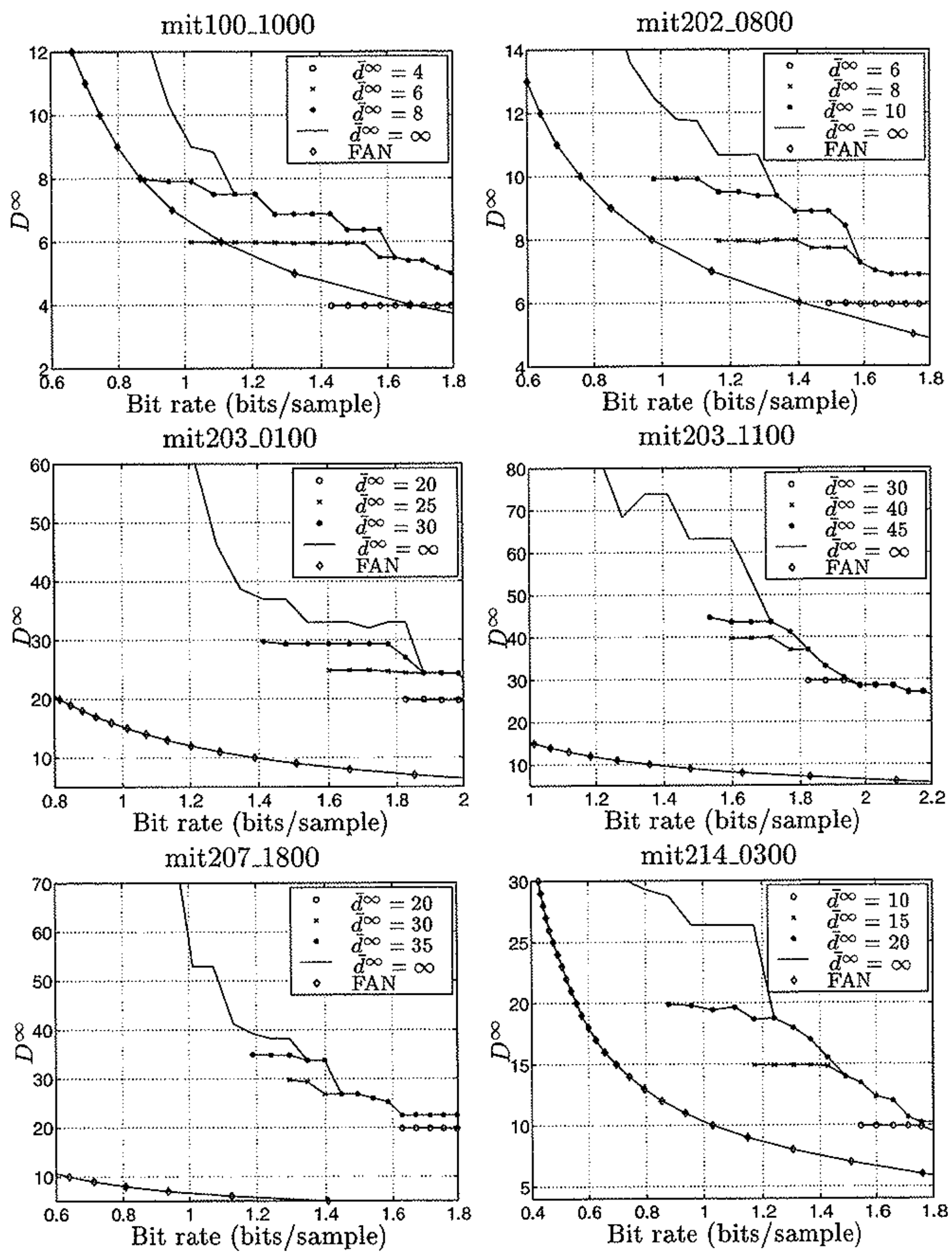


Figure 4.14: Maximum error, $D^\infty$, versus bit rate for the different coders and test signals.

From Table 4.2 it can be seen that for the CCSP algorithm based on linear interpolation with the maximum error bound incorporated with a sampling frequency of 360 Hz, the real time requirements are fulfilled for all block sizes and all values of the number of extracted samples, $M$. Even as the maximum error bound, $\bar{d}^{\infty}$, increases, the execution times will still be lower than the ones reported in Table 4.1, as even a high $\bar{d}^{\infty}$ will elimiate the need of taking arcs with a much to high maximum error into consideration when searching for a shortest path through the graph. For instance, with a block size of 500, with $M = 50$ and $\bar{d}^{\infty} = 100$, the execution time equals 2.41 sec. as opposed to 3.41 sec. in the linear interpolating case.

## 4.3 Second order polynomial compression

Reconstruction of a signal compressed by any of the time-domain algorithms mentioned so far is done by linear interpolation between the elements of the extracted subset of signal samples. This is a simple, but computationally effective way of reconstructing the signal. However, an ECG signal is not linear in its nature, but rather more curvaceous. It would therefore be interesting to investigate if it is possible to get a better approximation to the original signal, under the same compression ratio, by using a polynomial of higher degree in reconstruction of the signal. In this section we demonstrate how the CCSP algorithm can be further developed in order to reconstruct the signal by second order polynomials.

Applying linear interpolation in the reconstruction phase, the CCSP algorithm is proven to converge in cubic time. In [38] it is demonstrated how to cope with real time constraints of the algorithm. We now show that the CCSP algorithm can be applied to the case where polynomial approximation is used in reconstruction of the signal without increasing the computational complexity of the algorithm.

### 4.3.1 Optimization model

We still apply the definition of $M$, $N$ and $C$ given in Section 3.2. Assume $n_1 = 1$ and $n_M = N$. The approximation is then given by

$$\hat{y}(n) = \begin{cases} y(n) & \text{if } n \in C, \\ f_{n_k,n_{k+1}}(n) & n_k < n < n_{k+1} \forall n \notin C, n \in \{1, 2, ..., N\}. \end{cases} \tag{4.13}$$

| **CCSP, linear interpolation with bound on maximum error** | | | |
|---|---|---|---|
| Block size | Extracted samples, $M$ | Maximum error bound | Execution time in sec. |
| 100 | 5 | 5 | 0.04 |
| | | 8 | 0.07 |
| | 10 | 4 | 0.04 |
| | | 8 | 0.07 |
| | 30 | 3 | 0.04 |
| | | 8 | 0.11 |
| 200 | 10 | 18 | 0.21 |
| | | 25 | 0.24 |
| | 30 | 5 | 0.14 |
| | | 8 | 0.20 |
| | 50 | 3 | 0.11 |
| | | 6 | 0.19 |
| 300 | 10 | 18 | 0.36 |
| | | 25 | 0.39 |
| | 30 | 5 | 0.24 |
| | | 8 | 0.35 |
| | 50 | 4 | 0.23 |
| | | 8 | 0.41 |
| 400 | 10 | 18 | 0.52 |
| | | 25 | 0.62 |
| | 30 | 6 | 0.40 |
| | | 10 | 0.56 |
| | 50 | 4 | 0.33 |
| | | 8 | 0.61 |
| 500 | 30 | 13 | 0.83 |
| | | 18 | 1.01 |
| | 50 | 6 | 0.68 |
| | | 10 | 0.91 |
| | 70 | 5 | 0.65 |
| | | 8 | 0.94 |

Table 4.2: Execution times for the CCSP algorithm based on linear interpolation with bound on maximum error incoporated run on an HP9000 C360.

Here $f_{n_k,n_{k+1}}(n)$ denotes a presumed reconstruction of $y(n)$ based on $y(n_k)$, $y(n_{k+1})$ and all the intermediate samples and will be given a precise definition in the next section. In this way we get a piecewise approximation to the original signal. Between two sample amplitudes corresponding to two succeeding elements of $C$, different functions are used in reconstruction of the signal. The choice of $C$ will thus have a vital importance for the quality of our approximation of the signal.

Again, we are searching for a shortest path from vertex 1 up to vertex $N$ through the directed graph $G = (V, A)$. As before, the length of each arc $(i, j)$ in $A$ is given as the contribution to the total reconstruction error by eliminating all vertices between $i$ and $j$. This can be expressed as

$$d_{ij} = \sum_{n=i+1}^{j-1} (\hat{y}(n) - y(n))^2. \tag{4.14}$$

The length of $P_N$ will thus be the sum of the length of all arcs included in the path up to vertex $N$. Each arc $(i, j)$ in $A$ represents the possibility of letting $i$ and $j$ be consecutive members of $C$. Including an arc $(i, j)$ in $C$ has the effect of increasing the total path length by $d_{ij}$.

Hence we are faced with the following problem : Minimize the length of $P_N$, i.e., $D = \|P_N\| = \sum_{(i,j)\in P_N} d_{ij}$, under the constraint that $P_N$ contains no more than $M$ vertices. With the reconstruction method applied here, this is a modified version of the problem presented in Section 4.1.

### 4.3.2 Solution method

We now go on to show that the algorithm presented in Section 3.4 can be modified such that it handles second order reconstruction polynomials and still preserves its computational complexity of $\mathcal{O}(MN^2)$.

To be able to extract samples from the original signal in an optimal way, we need to know the contribution to reconstruction error introduced by including any two samples as consecutive members of $C$. The aim now is therefore to fit a function $f_{ij}$ to the data set $\{(n, y(n)) : n = i, ..., j\}$ in such a way that the reconstruction error is minimized. We wish to use a second order polynomial in this context, that is we let $f_{ij}(n) = a_{0ij} + a_{1ij}n + a_{2ij}n^2, n \in [i, j]$. If we interpolate between two end points $i$ and $j$, we have for each arc $(i, j)$ :

$$d_{ij} = \sum_{n=i+1}^{j-1} \left(a_{0ij} + a_{1ij}n + a_{2ij}n^2 - y(n)\right)^2, \tag{4.15}$$

$$a_{0ij} + a_{1ij}i + a_{2ij}i^2 = y(i), \tag{4.16}$$

$$a_{0ij} + a_{1ij}j + a_{2ij}j^2 = y(j). \tag{4.17}$$

The optimal parameters $a_{0ij}, a_{1ij}$ and $a_{2ij}$ are found by minimizing (4.15) under the constraints given in (4.16) and (4.17). By inserting these optimal parameters into (4.15), the minimal $d_{ij}$ is found for each arc.

The graph $G$ consists of $\frac{N(N-1)}{2}$ arcs. The expression for $d_{ij}$ is a sum of $j-i-1$ terms. Straightforward computation of all $a_{0ij}$'s, $a_{1ij}$'s, $a_{2ij}$'s and all $d_{ij}$'s will thus result in an algorithm with a complexity of $\mathcal{O}(N^3)$. Fortunately, this can be avoided by careful computation of the arc lengths.

The arc lengths are given by (4.15). Assume that we express (4.15) in terms of $a_{2ij}$ by the use of (4.16) and (4.17). By putting the derivative of this expression with respect to $a_{2ij}$ equal to zero we arrive at an expression of the following form :

$$a_{2ij} = \frac{\sum_{n=i+1}^{j-1}(\alpha_{0ij} + \alpha_{1ij}n + \alpha_{2ij}n^2)y(n) + \eta_{1ij}}{\eta_{2ij}}, \tag{4.18}$$

where

$$\eta_{1ij} = \sum_{n=i+1}^{j-1} \beta_{0ij} + \beta_{1ij}n + \beta_{2ij}n^2 + \beta_{3ij}n^3, \tag{4.19}$$

$$\eta_{2ij} = \sum_{n=i+1}^{j-1} \gamma_{0ij} + \gamma_{1ij}n + \gamma_{2ij}n^2 + \gamma_{3ij}n^3 + \gamma_{4ij}n^4. \tag{4.20}$$

All $\alpha_{ij}$'s, $\beta_{ij}$'s and $\gamma_{ij}$'s are expressions in $i$, $j$, $y(i)$ and $y(j)$ (see Appendix B) and hence all these coefficients are computed in $\mathcal{O}(N^2)$ time. The sums of powers of $n$ are evaluated by closed form formulas, and hence all $\eta_{1ij}$ and $\eta_{2ij}$ are computed in $\mathcal{O}(N^2)$ time. By defining $\Delta_{pj} = \sum_{n=1}^{j} n^p y(n)$, $p = 0, 1, 2$, we see that $\Delta_{p1}, \ldots, \Delta_{pN}$ are computed in $\mathcal{O}(N^2)$ time. Next, we compute

$$a_{2ij} = \frac{\sum_{p=0}^{2} \alpha_{pij}(\Delta_{p,j-1} - \Delta_{pi}) + \eta_{1ij}}{\eta_{2ij}}, \tag{4.21}$$

$1 < j < N$, involving $\mathcal{O}(N^2)$ operations.

When the optimal parameters are computed in the way described above, they are inserted into (4.15) in order to find the minimum arc lengths. This will lead to an expression of the form

$$d_{ij} = \sum_{n=i+1}^{j-1} \left(a_{0ij}^2 + 2a_{0ij}a_{1ij}n + \left(2a_{0ij}a_{2ij} + a_{1ij}^2\right)n^2 + \right.$$
$$2a_{1ij}a_{2ij}n^3 + a_{2ij}^2n^4 - 2y(n)\left(a_{0ij} + a_{1ij}n + \right.$$
$$\left.\left. a_{2ij}n^2\right) + y^2(n)\right).$$

By applying the same technique as in computation of $a_{2ij}$, all arc lengths are available in $\mathcal{O}(N^2)$ time. The detailed mathematical computations can be found in Appendix B.

When all arc lengths are available, the actual sample extraction takes place. This is accomplished by the dynamic programming algorithm described in Section 3.4.

### 4.3.3 Coding scheme

In order to be able to reconstruct a signal coded by a second order polynomial interpolating approach, we need three parameters for each retained sample. We represent each second order polynomial between vertices $n_k$ and $n_{k+1}$ by the sample amplitude $y(n_k)$, the distance, $n_{k+1} - n_k$ and the approximate sample value $\hat{y}\left(\frac{n_k+n_{k+1}}{2}\right)$. This data set determines the piecewise polynomial reconstruction uniquely.

As in the linear interpolating CCSP algorithm described in Section 4.1, we use a quantizer with a step size of $3\Delta_Q$ to the extracted signal samples. Subsequently, we apply a simple predictive encoding scheme and encode the first order difference of both parameters (first order DPCM), that is, each segment of the signal is represented by the three parameters $\delta_{y(k)} = Q_1(y(n_k)) - Q_1(y(n_{k-1}))$, $\delta_{\hat{y}(k)} = Q_2\left(\hat{y}\left(\frac{n_k+n_{k+1}}{2}\right)\right) - Q_2\left(\hat{y}\left(\frac{n_{k-1}+n_k}{2}\right)\right)$ and $\delta_{n(k)} = n_k - n_{k-1}, k = 2, 3, \ldots, M$, where $Q_1$ and $Q_2$ denotes quantization. In addition, we need to encode the absolute amplitude of the first sample point, $y(n_1)$.

We thus have triples of $(\delta_{y(k)}, \delta_{\hat{y}(n_k)}, \delta_{n(k)})$ to be encoded for each segment of the signal. Here we have the choice of several different coding strategies. As $\delta_{y(k)}$ and $\delta_{\hat{y}(k)}$ are in the same numerical range, while $\delta_{n(k)}$ is in a different numerical range, we consider two different coding strategies:

1. Two separate encoders, one for the amplitudes, $\delta_{y(k)}$ and $\delta_{\hat{y}(k)}$, and one for the distance, $\delta_{n(k)}$.
2. One encoder for the concatenated symbol $(\delta_{y(k)}, \delta_{n(k)})$ and one for the symbol $\delta_{\hat{y}(k)}$.

By investigating the joint probability distribution of $\delta_{y(k)}$ and $\delta_{n(k)}$ we find that it has sharp peaks representing frequently used combinations of $(\delta_{y(k)}, \delta_{n(k)})$ as can be seen in Figure 4.15. This indicates that there is a profit in coding the symbols according to alternative 2. This coding strategy will lead to a

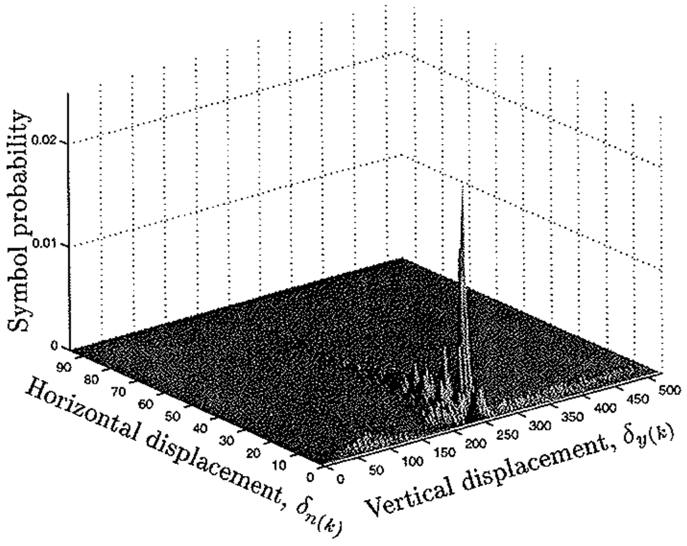


Figure 4.15: Symbol probability distribution of $\delta_{y(k)}$ and $\delta_{n(k)}$ at a bit rate of 1 bps.

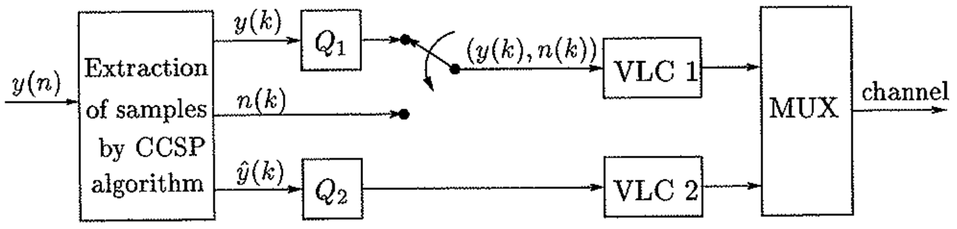


Figure 4.16: Structure of encoding system.

higher number of source symbols than alternative 1, as discussed in Section 4.1.2. However, experiments show that in the case of alternative 2 only a small fraction of the possible number of source symbols are actually used. In order to utilize the dependency between $\delta_{y(k)}$ and $\delta_{n(k)}$ we choose to use alternative 2 and encode the concatenated symbol $(\delta_{y(k)}, \delta_{n(k)})$ by one single encoder and the symbol $\delta_{\hat{y}(k)}$ by a separate encoder in this context. The structure of the coder is shown in Figure 4.16 where $Q_1$ and $Q_2$ denote quantizers, VLC denotes variable length coder and MUX is the multiplexer assembling the two bit streams into one for storage or transmission .

### 4.3.4 Numerical experiments and discussion

As before, we evaluate the coders in terms of PRD (given in Equation (2.3)), maximum error (given in Equation (2.1)), visual impression and execution times.

The test signals applied are the ones whose 10 first seconds are plotted in Appendix A. As we would like to compare the performance from the different coders, the test signals are coded using three different coders: The CCSP coder based on polynomial interpolation presented in this section, the CCSP coder based on linear interpolation presented in Section 4.1 and the FAN method presented in Section 2.2.1. For the CCSP algorithm based on polynomial interpolation, the extracted signal samples are encoded according to the procedure described in Section 4.3.3. The CCSP method and the FAN algorithm encodes the extracted signal samples as described in Section 4.1.2.

The input signal is divided into blocks of 500 samples before processing by the CCSP algorithm in order to keep the execution time down.

#### Evaluation based on the PRD measure

Figure 4.17 presents PRD versus bit rate for the six test signals for bit rates between 0.2 and 1.8 bps. From the plots we can see that both the CCSP algorithm based on linear interpolation and the CCSP algorithm based on second order polynomials generally perform much better than the FAN algorithm in terms of PRD, especially at low bit rates.

The CCSP method based on polynomial interpolation performs better than the CCSP method based on linear interpolation for low bit rates, i.e., below approximately 0.8 bps, while for higher bit rates the CCSP method based on linear interpolation performs better or similar. One possible explanation for this crossing of the rate-distortion curves is that given the number of retained samples, the CCSP algorithm based on polynomial interpolation extract samples in an optimal way with respect to reconstruction error. However, as the approximation has to be encoded, some quantization noise is introduced, and the solution may thus be slightly suboptimal. This is due to the the fact that the CCSP polynomial interpolation algorithm interpolates each arc between two end points (samples), but the third point used to represent $f_{n_k,n_{k+1}}$ is approximated on basis of the optimal polynomial coefficients. This approximated signal sample has to be quantized and this may cause the actual distortion to differ from the optimal distortion with some amount. This problem is not an issue in the linear interpolation case as the representation points corresponds to samples which are quantized in advance. In the case of test signal mit100_1000 at a bit rate of 1 bps, the deviation from the optimal distortion amounts to 0.78 % of the optimal distortion. For the same test signal at a bit rate of 0.5 bps, the deviation amounts to 0.14 %. The deviation grows larger for higher bit rates, as more samples are retained and thus we get a higher contribution to the total quantization error.

### Evaluation based on the maximum error

Figure 4.18 presents obtained maximum errors for the different coders as a function of bit rate. We see that the FAN algorithm performs best for all test signals. The CCSP method based on polynomial interpolation performs better than the CCSP method based on linear interpolation for test signals mit100_1000 for bit rates below 1 bps, for test signal mit202_0800 for bit rates between 1.17 bps 1.3 bps, mit203_0100 for bit rates above 1.7 bps and for an area of test signal mit214_0300 for bit rates between 0.85 bps and 1.2 bps.

Incorporation of the maximum error into the CCSP algorithm as was done in Section 4.2 can also be done in the polynomial interpolation case. This will lower the maximum error and most likely reduce the execution time of the algorithm.

### Evaluation based on visual inspection

We accompany the performance evaluation of the coders with a visual inspection of the reconstructed signal. This is to show coding artifacts as they appear for the different coders. We have chosen a short segment of the mit100_1000 signal representing regular sinus rhythm. The reconstructed signal segment is shown at a bit rate of 1.0 bit per sample in Figure 4.19 and 0.5 bits per sample in Figure 4.20. The original signal is also included. From the Figures it can be seen that both the FAN and the CCSP compression algorithms smooth out some of the details in the original signal, especially for low bit rates (see Figure 4.20). The FAN algorithm produces a reconstructed signal in which less details are obtained than what is the case for the the CCSP methods. The CCSP coder based on polynomial interpolating approximation produces a reconstructed signal which is less rough than the other methods.

### Evaluation based on execution time

An important aspect of a compression algorithm is the ability to run in real time. In order to shed some light upon the real time performance of the CCSP

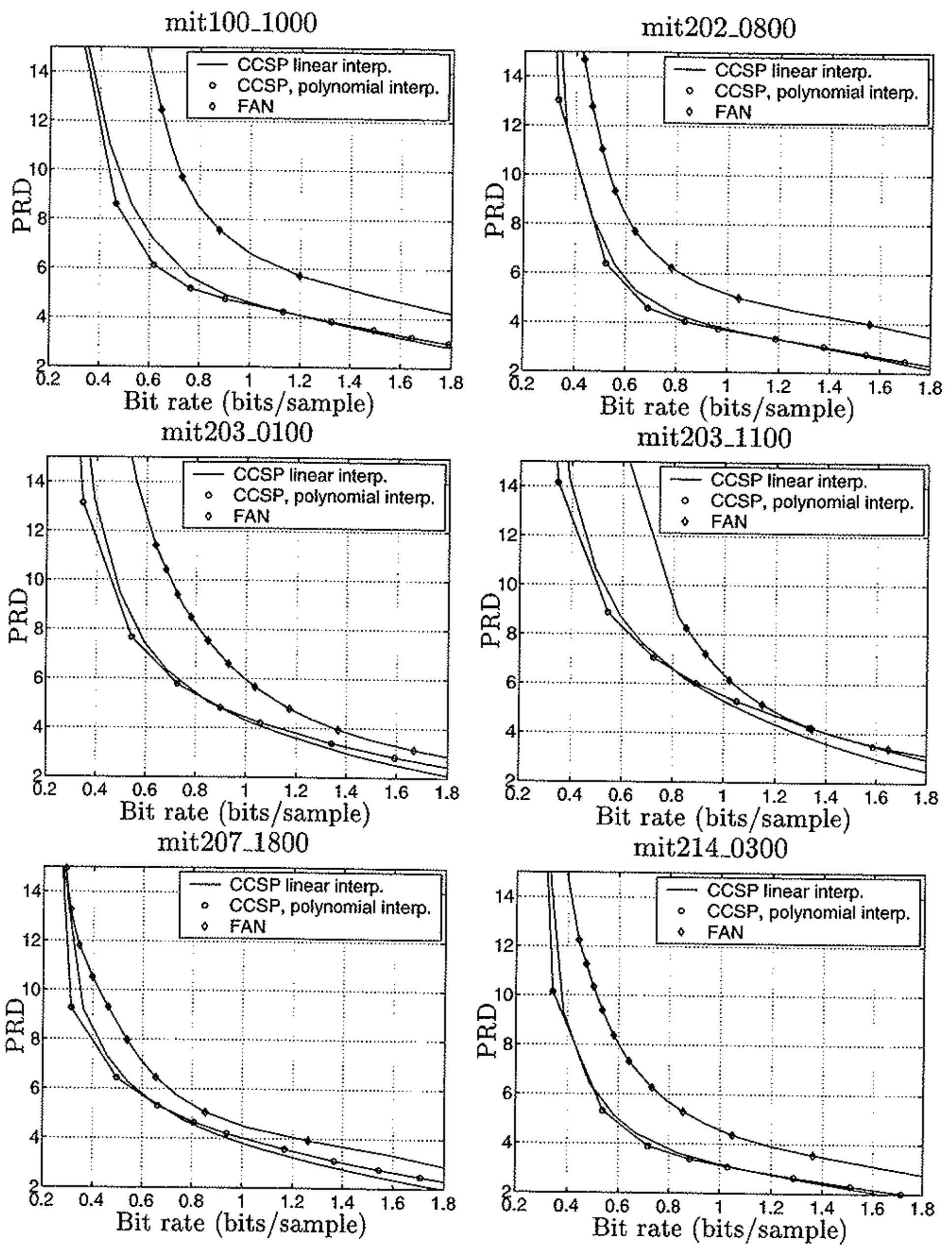


Figure 4.17: PRD versus bit rate for the different coders and test signals. Solid line: CCSP linear interpolation. Solid line with circles: CCSP polynomial interpolation. Solid line with diamonds: FAN.

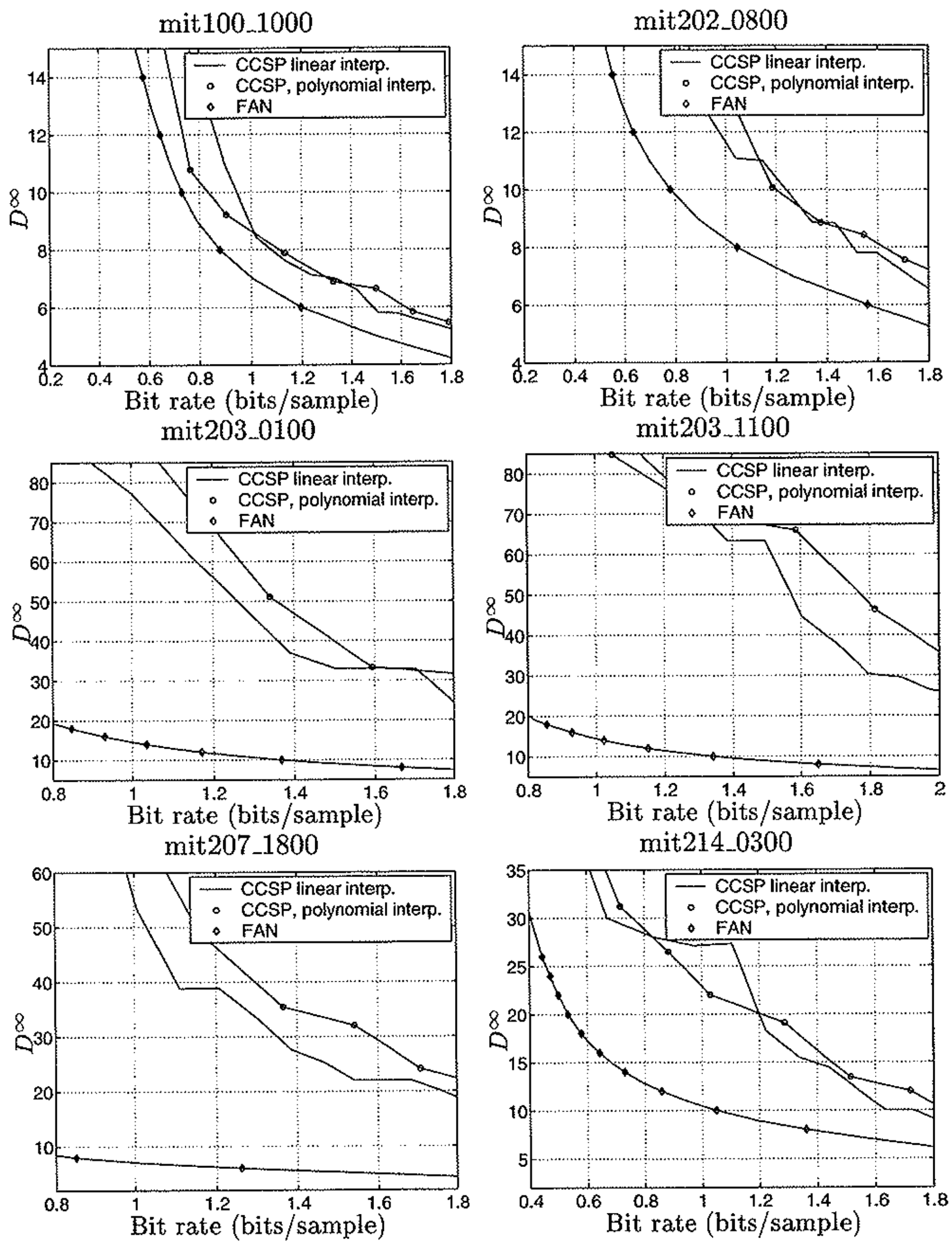


Figure 4.18: Maximum error, $D^\infty$, versus bit rate for the different coders and test signals. Solid line: CCSP linear interpolation. Solid line with circles: CCSP polynomial interpolation. Solid line with diamonds: FAN.

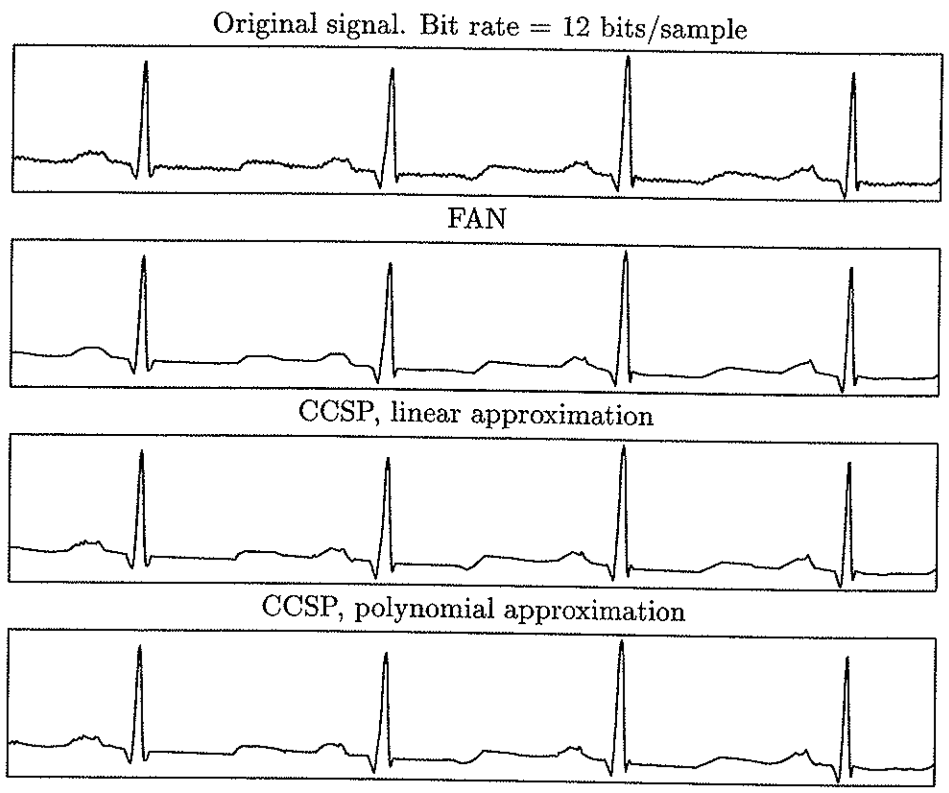


Figure 4.19: Short segment of reconstructed signal (taken from mit100_1000) at 1.0 bits per sample.

compression algorithm presented in this section, we present some benchmarks for the algorithm here, i.e., some execution times for the algorithm for different compression ratios run on a specific machine.

The complexity of the algorithm for polynomial interpolation is of $\mathcal{O}(MN^2)$ as in the linear case, but the execution time is a bit longer in the polynomial interpolation case. This is due to the computations necessary to find the optimal polynomial coefficients. In order to cope with real time constraints, the techniques presented in [38] may be applied.

Table 4.3 shows execution times for the CCSP algorithm based on polynomial interpolation for different block sizes and different number of extracted samples. The experiments are run on the same machine, and with the same test signals as in the previous sections. We have extracted different number of samples for the different block sizes, in order to cover approximately the same area of the rate-distortion curves shown in Figure 4.17.

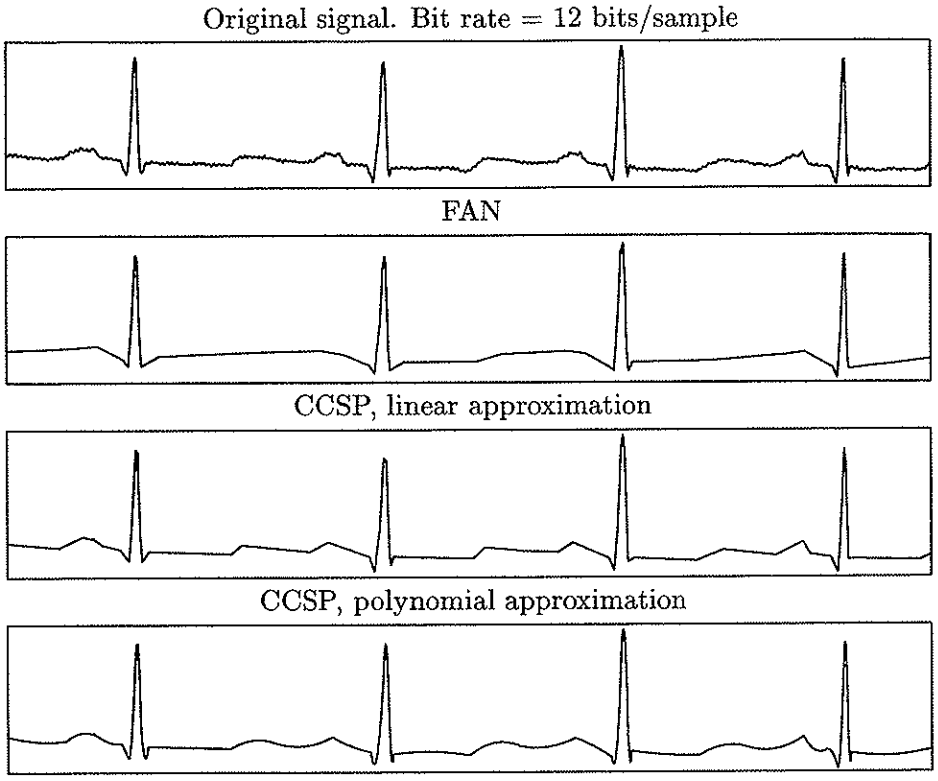


Figure 4.20: Short segment of reconstructed signal (taken from mit100_1000) at 0.5 bits per sample.

From Table 4.3 it can be seen that for the CCSP algorithm based on polynomial interpolation with a sampling frequency of 360 Hz, the real time requirements are fulfilled for a block size of 100. For a block size of 200, the real time requirements are fulfilled for a number of extracted samples, $M$, below 15. For larger block sizes, real time requirements are violated. Keep in mind that the implementation of the coder used in all experiments throughout this dissertation has *not* been optimized with respect to efficiency. The execution times reported here are generated using straightforward C++ implementation of the coder.

Having investigated the use of second order polynomials in reconstruction of the signal, a natural question is: What about higher order polynomials? Will polynomials of higher order than two result in a closer approximation to the original signal at even lower bit rates than the techniques described so far?

These questions were investigated in the work done in [43]. The linear inter-

| **CCSP with polynomial interpolation** | | |
|---|---|---|
| Block size | Extracted samples, $M$ | Execution time in sec. |
| 100 | 5 | 0.10 |
| | 10 | 0.12 |
| | 30 | 0.16 |
| 200 | 5 | 0.41 |
| | 10 | 0.47 |
| | 30 | 0.72 |
| | 50 | 0.92 |
| 300 | 5 | 0.92 |
| | 10 | 1.07 |
| | 30 | 1.59 |
| | 50 | 2.06 |
| | 70 | 2.47 |
| 400 | 5 | 1.65 |
| | 10 | 1.88 |
| | 30 | 2.82 |
| | 50 | 3.77 |
| | 70 | 4.62 |
| | 90 | 5.32 |
| 500 | 5 | 2.59 |
| | 10 | 2.96 |
| | 30 | 4.63 |
| | 50 | 6.03 |
| | 70 | 7.43 |
| | 90 | 8.72 |

Table 4.3: Execution times for the CCSP algorithm based on polynomial interpolation run on an HP9000 C360.

polating algorithm presented in Section 4.1 was further developed to include reconstruction of the signal based on polynomials of arbitrary order. Experimental results from this work all showed the same tendency: Polynomials of a higher order than 2 results in little or no gain in terms of PRD. As the order of the polynomial is increased beyond 3, the performance is lowered in terms of PRD.

Due to the limited success in terms of rate-distortion performance of this approach, the idea of increasing the order of polynomials beyond two has not been investigated further. However, this approach has given us valuable insight into the problem. More details can be found in [43].

## 4.4 Summary

In this chapter we have developed three new coders based on graph theoretic approach to the problem of compressing ECG signals. We consider a digraph where each vertex represents a sample point in the original ECG signal and each arc signifies inclusion of the samples corresponding to its connecting vertices as consecutive retained samples. We then seek a selection of arc and vertices in the digraph which is to constitute a path from the first to the last vertex in the graph, corresponding to a path from the first to the last sample point.

Between the extracted vertices of the digraph, we apply two different arc models:

- Linear interpolating approach, i.e., applying straight lines between the elements of the extracted sample set.
- Second order polynomial interpolation, i.e., applying second order polynomials between the elements of the extracted sample set. This is done without increase in the computational complexity of the algorithm.

We also develop a method for incorporating two error measures into the CCSP algorithm, the maximum error and sum of squared errors. The goal is to minimize the total error while respecting bounds on the maximum error and the size of the compressed signal. We outline the idea behind efficient computation of the maximum error, and show how to incorporate a bound on this error in an existing algorithm for minimizing the total error. Unlike previously known methods, the suggested algorithm enables us to control both the maximum error and the sample reduction ratio.

Three algorithms are implemented based on the theoretic basis of this chapter: A linear interpolating CCSP algorithm, a CCSP algorithm where both the maximum error and sum of squared errors is incorporated and a CCSP algorithm based on second order polynomial interpolation.

The complexity of the algorithms implemented is no worse than cubic in the number of samples, and numerical experiments show that the algorithms can be run in real time with minor modifications.

Compared to traditional time-domain algorithms, our approaches contribute to a drastic reduction in the total error of the decoded signal, depending on the compression rate. From this we learn that frequently applied time domain methods suffer from significant deviations from the theoretical optimum. In terms of maximum error, the traditional method performs well. However, visual inspection of the reconstructed signal indicates that the traditional methods is outperformed by our new, optimal methods.

When comparing different arc models applied in reconstruction of the signal, we see that we obtain less total error by increasing the order of the polynomials from one to two. Visual inspection of the signal confirms this. By increasing the order of the polynomial further, we gain marginally from the order of two to three in terms of total error, but by increasing the order of the polynomial further, we lose in terms of total error. However, the increased execution time occurring as a consequence of increasing the order of the polynomial from two to three, indicates that there is little profit in this.

# Chapter 5

# Piecewise polynomial non-interpolating compression

So far, we have looked at signal compression by the use of interpolating approximation between the elements of the extracted subset of signal samples. However, by applying interpolation between the extracted signal samples we insist on exact equality between the reconstructed and the original signal samples at the points of extraction while allowing the approximation to deviate from the original signal at all other points. This imposes restrictions on the algorithm. Removing this restriction will give us a higher degree of freedom in extraction of signal samples, and thus hopefully a better representation of the original signal. In this chapter we therefore demonstrate how the optimization algorithm presented in Chapter 3 can be developed into an approach where we apply piecewise *non-interpolating* approximation.

We look into two different arc descriptions in the non-interpolating case. Section 5.1 is concerned with *linear* non-interpolating approach, where we use straight lines in reconstruction of the signals, whereas Section 5.2 is devoted to *polynomial* non-interpolating approach. In this case we apply second order polynomials in reconstruction of the signal.

We still have to keep in mind that the computational complexity of the algorithm is a crucial point and develop the new approaches in a way that ensures a computational complexity that will be as low as possible.

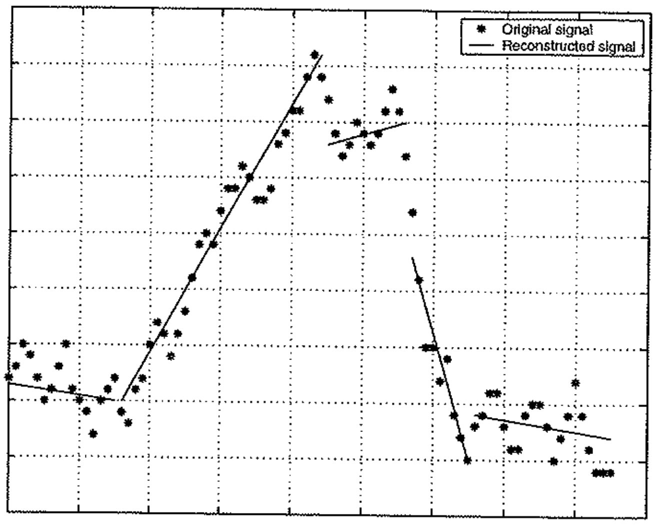


Figure 5.1: Example of a short sequence of original and reconstructed signal with a piecewise linear non-interpolating approach.

## 5.1 First order non-interpolating polynomial compression

Applying linear interpolation in the reconstruction phase, the algorithm in Section 4.1 is proven to converge in cubic time. In this section it is shown how the idea presented there can be further developed in order to handle piecewise linear non-interpolating approximation in reconstruction of the signal. This is obtained without increasing the computational complexity of the algorithm.

An example of an original and a reconstructed signal with the piecewise linear non-interpolating approach is shown in Figure 5.1. We see that the idea is that if one segment ends in sample $j$, the next segment starts in sample $j+1$.

### 5.1.1 Problem definition

We apply the definitions of signal samples, $M$ and $N$ given in Section 3.2. Let the approximated signal be denoted by $\hat{y}(k)$. In general $\hat{y}(k) \neq y(k)$. As before, we seek an appropriate compression set $C = \{n_1, ..., n_M\}$ and the corresponding approximated sample values $\hat{y}(n_1), \dots, \hat{y}(n_M)$ to represent the original signal. Assume $n_1 = 1$ and $n_M = N$, and let $i$ and $j$ be consecutive members of $C$. The approximation is then given by

$$\hat{y}(n) = \hat{y}(i) + \frac{\hat{y}(j) - \hat{y}(i)}{j - i}(n - i) \begin{array}{ll} \forall n \in [i, j) & \text{if } j < N, \\ \forall n \in [i, j] & \text{if } j = N. \end{array} \tag{5.1}$$

In this way we get a piecewise linear non-interpolating approximation to the original signal starting in $n_1 = 1$ and ending in $n_M = N$.

Again, we are searching for a shortest path, $P_N$, from node 1 to node $N$ through the graph $G = (V, A)$ defined in Section 3.3. The length of each arc $(i, j)$ in $A$ is given as the contribution to the total reconstruction error by eliminating all nodes between $i$ and $j - 1$ if $j < N$ and between $i$ and $j$ if $j = N$. This length can be expressed as

$$d_{ij} = \begin{cases} \sum_{n=i}^{j-1} (\hat{y}(n) - y(n))^2 & \text{if } j < N, \\ \sum_{n=i}^{j} (\hat{y}(n) - y(n))^2 & \text{if } j = N. \end{cases} \tag{5.2}$$

Notice that we let the piecewise reconstructed signal be given by a left-continuous function. That is, we compute each arc $(i, j)$ for all legal combinations of $i$ and $j$, but we let its length $d_{ij}$ be based on $i, i + 1, \ldots, j - 1$ as long as $j < N$. This way the functional value of the right index, $j$, is discarded.

Defining the problem this way, we arrive at the same problem as in Section 3.2: Minimize the length of $P_N$ under the constraint that $P_N$ contains no more than $M$ vertices. When all arc lengths are available this problem is solved with the shortest path algorithms described in Section 3.4.

### 5.1.2 Solution method

The parameters describing each straight line segment of the approximation to the original signal is found by minimizing the expression for $d_{ij}$ with respect to these parameters. We do this by rewriting Equation (5.1) as

$$\hat{y}(n) = a_{0ij} + a_{1ij}(n - i) \begin{array}{ll} \forall n \in [i, j) & \text{if } j < N, \\ \forall n \in [i, j] & \text{if } j = N. \end{array} \tag{5.3}$$

We still have to pay attention to the computational complexity in the computation of the arc lengths, as straightforward computation of all the arc lengths will result in an algorithm with a complexity of $\mathcal{O}(N^3)$. We apply a procedure similar to the one described in Section 4.3.2. We express $d_{ij}$ in terms of

$a_{0ij}$ and $a_{1ij}$ and then put the derivative of $d_{ij}$ with respect to these parameters equal to zero. After inserting closed form expressions for the expressions involving finite sums of powers of n, we arrive at an expression of the form

$$d_{ij} = \tag{5.4}$$

$$(j-i)a_{0ij}^2 + (j-i)(j-i-1)a_{0ij}a_{1ij} + 2(ia_{1ij} - a_{0ij})\sum_{n=i}^{j-1} y(n) +$$

$$\frac{1}{6}(j-i)(j-i-1)(2j-2i-1)a_{1ij}^2 - 2a_{1ij}\sum_{n=i}^{j-1} ny(n) + \sum_{n=i}^{j-1} y(n)^2,$$

where

$$a_{0ij} = \frac{2(2j+i-1)\sum_{n=i}^{j-1} y(n) - 6\sum_{n=i}^{j-1} ny(n)}{(j-i)(j-i+1)}, \tag{5.5}$$

$$a_{1ij} = \frac{12\sum_{n=i}^{j-1} ny(n) - 6(i+j-1)\sum_{n=i}^{j-1} y(n)}{(j-i)(j-i-1)(j-i+1)}. \tag{5.6}$$

Equations (5.4), (5.5) and (5.6) are valid for the case where $j < N$. When $j = N$, we have to substitute $j+1$ for $j$. We will thus get similar expressions to Equations (5.4), (5.5) and (5.6) for the case where $j = N$.

We have to compute $d_{ij}$, and thus $a_{0ij}$ and $a_{1ij}$, for every legal combination of $i$ and $j$. By applying a similar procedure to the one used in Section 4.1.1 all the arc lengths $d_{ij}$ are available in $\mathcal{O}(N^2)$ time.

### 5.1.3 Encoding scheme

In general we need three parameters for each segment of the signal in order to describe the piecewise linear non-interpolating approximation uniquely. The exact optimization algorithm described in Section 5.1.1, represents each linear segment of the signal between nodes $n_k$ and $n_{k+1}$ by the approximated sample values $\hat{y}(n_k)$ and $\hat{y}(n_{k+1}-1)$ in addition to $\delta_{n(k)} = n_{k+1} - n_k$. This is shown in Figure 5.2. Before encoding, we change the representation of the line segments slightly. The approximated sample values representing the end of one line segment and the start of the next one is replaced by its mean, i.e.,

$$\hat{y}_{mid}(n_{k+1}) = \frac{\hat{y}(n_{k+1}-1) + \hat{y}(n_{k+1})}{2}, \tag{5.7}$$

and the distance between the mean and the approximated sample values, i.e.,

$$\delta\hat{y}_{mid}(k+1) = Q\left(\hat{y}(n_{k+1}-1) - \hat{y}_{mid}(n_{k+1})\right) = Q\left(\hat{y}_{mid}(n_{k+1}) - \hat{y}(n_{k+1})\right), \tag{5.8}$$

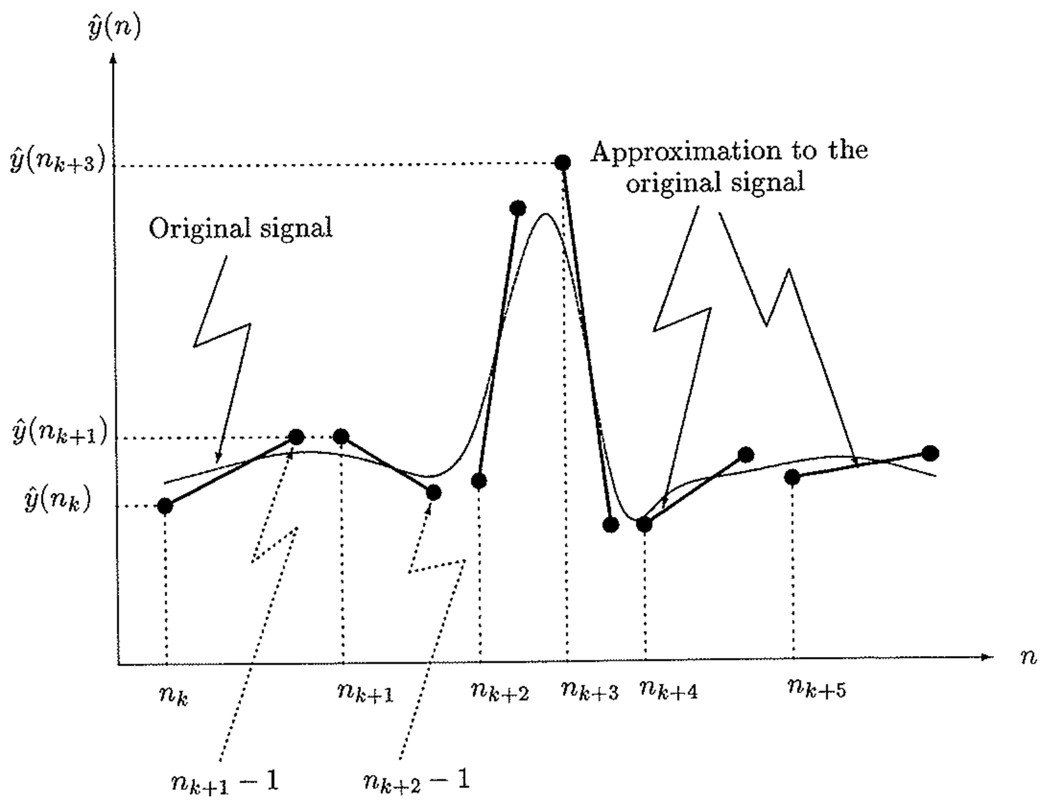


Figure 5.2: Example of a short sequence of original and reconstructed signal.

where $Q$ denotes quantization. We use a uniform quantizer with a step size of $3\Delta_Q$, in the numerical experiments in the next section. This manipulation of the parameters will lower their dynamic range as compared to encoding all the approximated sample values directly.

We thus have three parameters to be encoded for each segment of the signal: $\delta_{n(k)}, \hat{y}_{mid}(n_k)$ and $\delta\hat{y}_{mid}(k)$. We apply separate VLC's for each of these parameters.

### 5.1.4 Numerical experiments and discussion

For evaluation of the performance of the coders we apply the PRD error measure given in Equation (2.3), the maximum error given in Equation (2.1), visual inspection of the reconstructed signal as well as reporting some benchmarks of the algorithm.

We apply the same test signals as before, see Appendix A, and compare results from the linear non-interpolating CCSP algorithm described in this section to the linear interpolating CCSP algorithm descibed in Section 4.1 as well as the FAN algorithm descibed in Section 2.2.1. Both the linear interpolation CCSP algorithm and the FAN algorithm approximate the signal by linear line segments represented by two parameters per segment of the signal. These parameters are encoded according to the description in Section 4.1.2.

### Evaluation based on the PRD measure

Figure 5.3 presents obtained PRD's for the coders for bit rates between 0.2 and 1.8 bits per sample (bps). The FAN method is outperformed by both the CCSP methods by a wide margin for nearly all test signals. An exception is seen for test signal mit203_1100 fot bit rates above 1 bps and for test signal mit207_1800 for bit rates below 0.35 bps, where the FAN method actually outperforms the CCSP method based on linear non-interpolating approch.

The CCSP method based on linear interpolation gives lower PRD's than the non-interpolating approach for all test signals and all bit rates. The reason why the interpolating approach performs best is that it represents each arc of the signal with two parameters as opposed to three in the non-interpolating case. Thus the increased quality of the reconstructed signal which we hoped would be the result of the extra degree of freedom obtained by releasing the interpolation restriction is lost in the rate-distortion tradeoff.

### Evaluation based on the maximum error

Figure 5.4 presents obtained maximum errors for the coders. We see that the FAN algorithm performs best for all test signals. Generally the CCSP algorithm based on linear interpolation gives lower maxium error than does the non-interpolating approach. An exception is seen for test signals mit203_1100 for bit rates below 1.35 bps and for test signal mit214_0300 for bit rates between 0.9 bps and 1.2 bps. As before, we can also incorporate a maximum error bound into the non-interpolating approch as is done in Section 4.2 and thus lower the maximum error for the CCSP algorithm based on non-interpolating approach.

### Evaluation based on visual inspection

Figures 5.5 and 5.6 show a short segment of a reconstructed signal taken from mit100_1000, at bit rates of 1.0 bps and 0.5 bps, respectively. The original

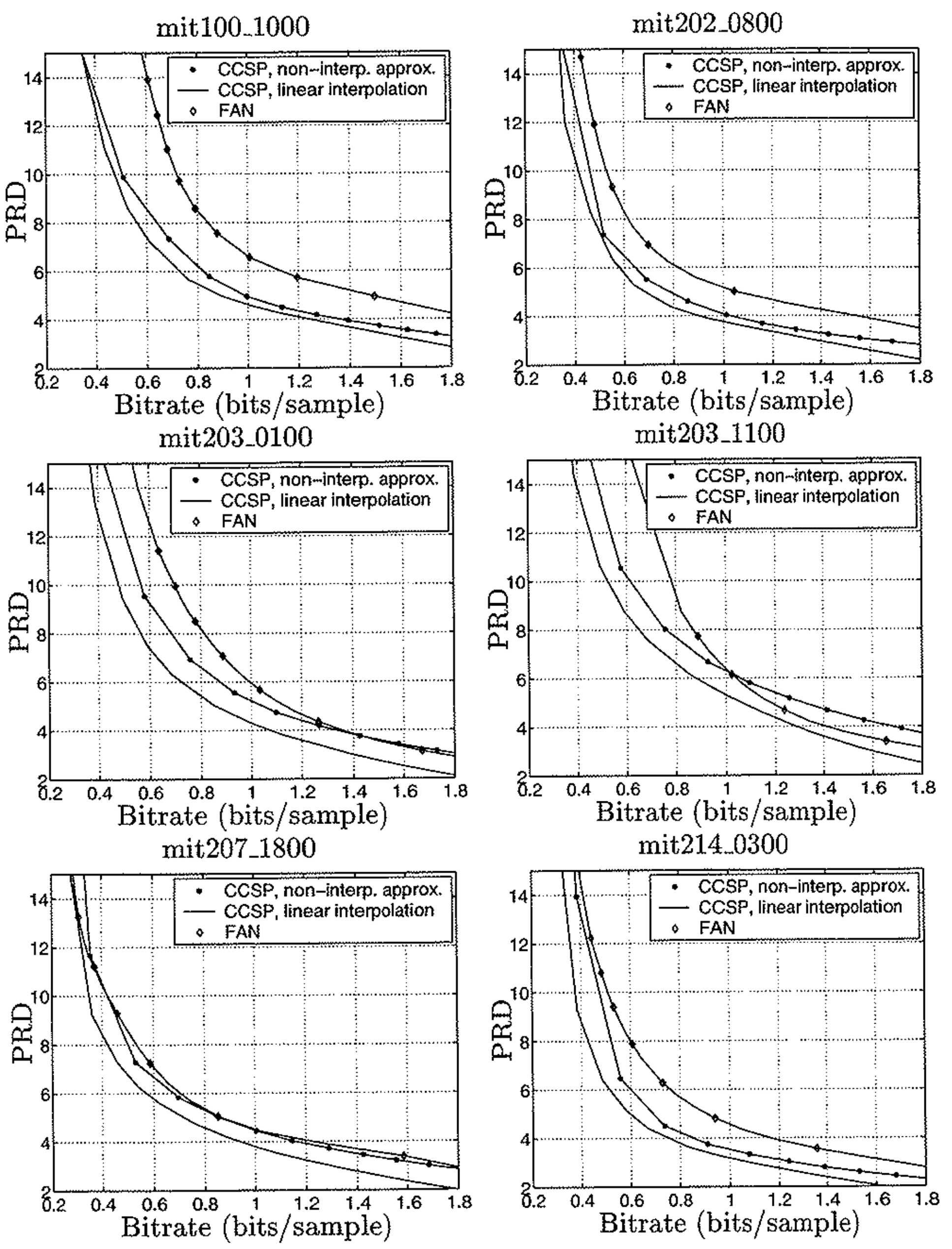


Figure 5.3: PRD versus bit rate for the different coders and test signals. Solid line: CCSP linear interpolation. Solid line with stars: CCSP non-interpolating approach. Solid line with diamonds: FAN.

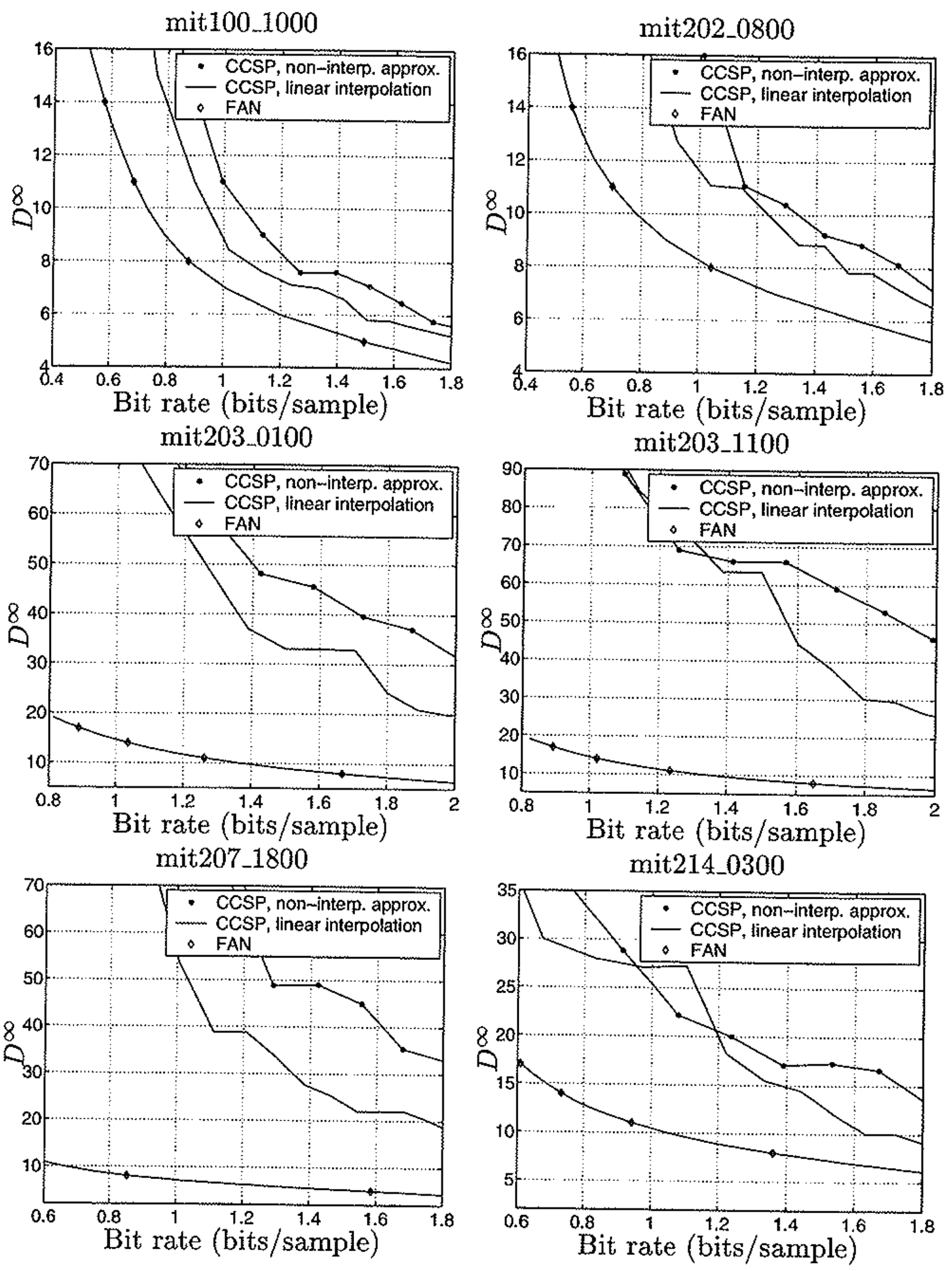


Figure 5.4: Maximum error, $D^{\infty}$, versus bit rate for the different coders and test signals. Solid line: CCSP linear interpolation. Solid line with stars: CCSP non-interpolating approach. Solid line with diamonds: FAN.

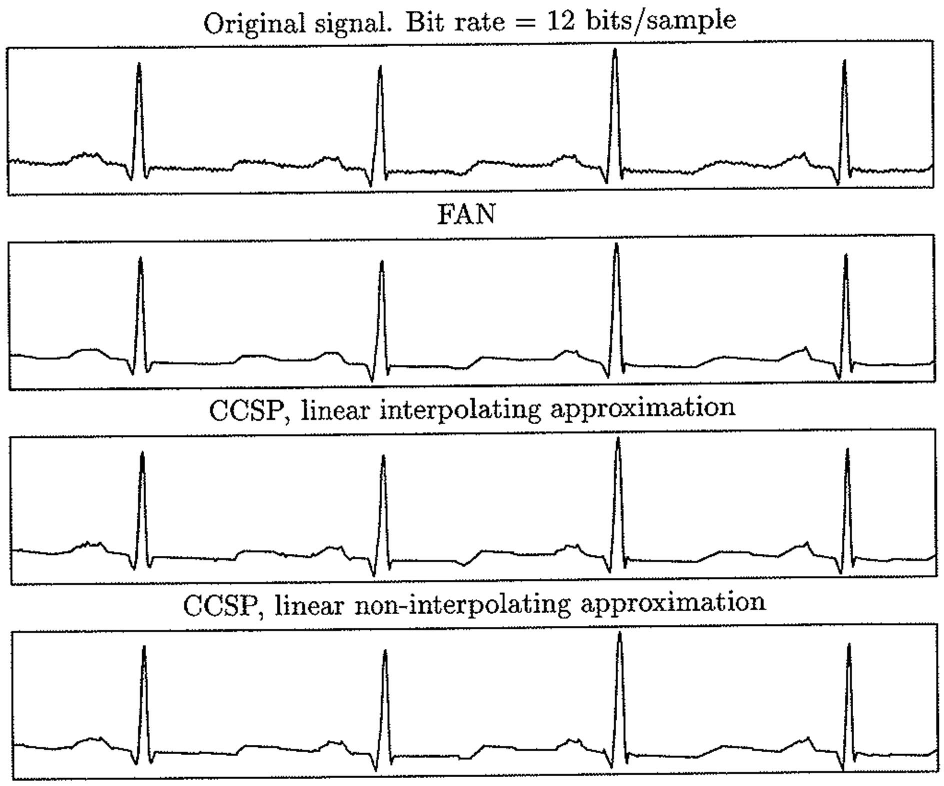


Figure 5.5: Short segment of reconstructed signal (taken from mit100_1000) at 1.0 bits per sample.

signal is also included. It is hard to say which method performs best for someone who is not an expert in the area of interpreting ECG signals. However, we can see from the plots in Figures 5.5 and 5.6 that it seems like the CSSP non-interpolating approach has a bit more jagged look than the CCSP interpolating approach. The FAN method seems to produce a reconstructed signal where less details are retained.

### Evaluation based on execution time

To get an impression of the real time performance of the CCSP algorithm based on a non-interpolating linear approximation, we present some benchmarks for the algorithm here, i.e., some execution times for the algorithm for different compression ratios run on a specific machine.

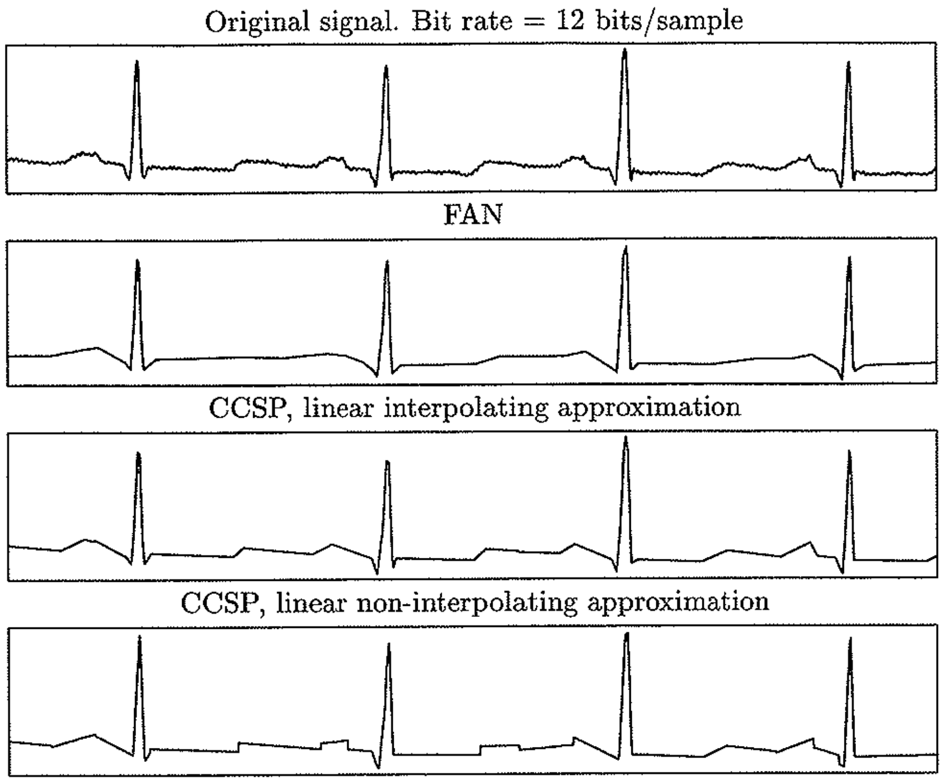


Figure 5.6: Short segment of reconstructed signal (taken from mit100_1000) at 0.5 bits per sample.

The complexity of the algorithm for linear non-interpolating interpolation is $\mathcal{O}(MN^2)$. This is the same complexity as for the linear interpolating case and the polynomial interpolating case. However, execution time may vary a bit between the different approaches, due to different number of computations necessary inside the loops of the algorithms.

Table 5.1 shows execution times for the CCSP algorithm based on linear non-interpolating approach for different block sizes and different number of extracted samples. The experiments are run on an HP9000 C360 work station with a 367 MHz processor and the test signal is extracted from the beginning of mit100_1000. The execution times are based on the average of 10 runs. We have extracted different number of samples for the different block sizes, in order to cover approximately the same area of the rate-distortion curves shown in Figure 5.3.

From Table 5.1 it can be seen that for the CCSP algorithm based on linear non-interpolating approach with a sampling frequency of 360 Hz, the real time requirements are fulfilled for a block size of 100. For a block size of 200, the real time requirements are fulfilled for all values of $M$ except $M = 70$. For larger block sizes, real time requirements are fulfilled for small values of the number of extracted samples, $M$. As $M$ grows, real time requirements are violated. In order to cope with real time requirements, the technique presented in [38] can be implemented.

Comparing Table 5.1 to Table 4.1 we see that, although the algorithms are identical with respect to complexity, the non-interpolating approximation is generally a bit slower than the interpolating one. This is due to the fact that in the non-interpolating case, there are more computations to be performed, as each arc of the signal is represented with three parameters as opposed to two in the interpolating case.

## 5.2 Second order non-interpolating polynomial compression

In Section 5.1 we apply straight lines in reconstruction of the signal. We saw that the results were not as good as we hoped for. However, as we saw in Chapter 4.3, second order polynomials generally resulted in a more efficient compression scheme in terms of PRD, than applying straight lines. To resume the thread of Chapter 4, it would therefore be interesting to investigate if it is possible to get a better approximation to the original signal, at the same bit rate, by using a polynomial of higher degree in reconstruction of the signal. This is similar to the idea implemented in Section 4.3, but in the present case we remove the interpolation restriction. This will give us another degree of freedom, and thus hopefully a better representation of the original signal. In this section we demonstrate how the algorithm in Section 4.3 can be further developed in order to apply to *non-interpolating* approximation by the use of second order polynomials. This is obtained without increasing the computational complexity of the algorithm.

### 5.2.1 Problem definition

We keep the notation for original sample values, compression set and number of original and extracted samples as defined in Section 3.2, and define the polynomial approximation

$$\hat{y}(n) = f_{n_k,n_{k+1}}(n) \quad \begin{array}{ll} \forall n \in [n_k, n_{k+1}) & \text{if } n_{k+1} < N, \\ \forall n \in [n_k, n_{k+1}] & \text{if } n_{k+1} = N. \end{array} \tag{5.9}$$

| **CCSP, linear non-interpolating approach** | | |
|---|---|---|
| Block size | Extracted samples, $M$ | Execution time in sec. |
| 100 | 5 | 0.02 |
| | 10 | 0.04 |
| | 30 | 0.10 |
| 200 | 5 | 0.08 |
| | 10 | 0.16 |
| | 30 | 0.50 |
| | 50 | 0.76 |
| 300 | 5 | 0.18 |
| | 10 | 0.38 |
| | 30 | 1.15 |
| | 50 | 1.81 |
| | 70 | 2.39 |
| 400 | 5 | 0.32 |
| | 10 | 0.65 |
| | 30 | 2.08 |
| | 50 | 3.38 |
| | 70 | 4.49 |
| | 90 | 5.53 |
| 500 | 5 | 0.51 |
| | 10 | 1.01 |
| | 30 | 3.23 |
| | 50 | 5.34 |
| | 70 | 7.31 |
| | 90 | 9.18 |

Table 5.1: Execution times for the CCSP algorithm based on linear non-interpolating approach run on an HP9000 C360.

Here $f_{n_k,n_{k+1}}(n)$ denotes a presumed reconstruction of $y(n)$ based on $\hat{y}(n_k)$, $\hat{y}(n_{k+1})$ and all intermediate samples. We wish to use a second order polynomial in this context, that is we let

$$f_{ij}(n) = a_{0ij} + a_{1ij}n + a_{2ij}n^2 \begin{array}{ll} \forall n \in [i,j) & \text{if } j < N, \\ \forall n \in [i,j] & \text{if } j = N. \end{array} \tag{5.10}$$

The parameters of $f_{ij}$ are computed for all possible indices $i$ and $j$, $j > i$ and this will give us a piecewise non-interpolating approximation to the original signal.

In the same way as the *linear* non-interpolating approach, we let $f_{ij}(n)$ be defined on the domain $n \in [i, j)$ as long as $j < N$. That is, we represent the signal by second order line pieces, $f_{ij}$, which are not connected in the time indices. This means that if one segment ends in the index $j$, the next segment starts in $j+1$. This is a natural way of representing a *non-interpolating* approximation when working on digitized signals.

Working on the directed graph $G = (V, A)$, we have to compute the length of all the arcs in $G$. These lengths can be expressed as

$$d_{ij} = \sum_{n=i}^{j-1} (f_{ij}(n) - y(n))^2 \quad \text{if } j < N,$$
$$d_{ij} = \sum_{n=i}^{j} (f_{ij}(n) - y(n))^2 \quad \text{if } j = N.$$

By calculating the distortion, $d_{ij}$, between any two points $i$ and $j$, $j > i$, up to but not including $j$, as long as we have not reached the last sample point, $N$, we ensure that the distortion is additive. This means that the distortion for the total reconstructed signal is made up of a sum of the segment distortions, which is an important property of the distortion measure in our application.

We are searching for the shortest path from vertex 1 to vertex $N$ through the graph. When all arc lengths are available, we are therefore faced with the same problem we have solved before: Minimize the length of $P_N$ under the constraint that $P_N$ contains no more than $M$ vertices. This problem is solved by the dynamic programming algorithm thoroughly described in Section 3.4.

The parameters $a_{0ij}$, $a_{1ij}$ and $a_{2ij}$ describing each line segment of the approximation to the original signal are found by minimizing the expression for $d_{ij}$ with respect to these parameters, in the same way as in the linear non-interpolating case (only we have one more parameter in this case). We still have to keep in mind that straightforward computation of all the arc lengths will result in an algorithm with a complexity of $\mathcal{O}(N^3)$. Fortunately, this can be avoided by careful computation of the arc lengths similar to the procedure shown in Appendix B. We then arrive at expressions of the following form:

$$d_{ij} =$$

$$(j-i)\Bigg(a_{0ij}^2 + (j-i-1)a_{0ij}a_{1ij} + \frac{1}{6}(2a_{0ij}a_{2ij} + a_{1ij}^2)(j-i-1)(2j-2i-1)$$
$$+\frac{1}{2}a_{1ij}a_{2ij}(j-i)(j-i-1)^2 + \frac{a_{2ij}^2}{30}(j-i-1)(2j-2i-1)(3j^2-3j-6ij$$
$$+3i^2+3i-1)\Bigg) + 2(a_{1ij}i - a_{0ij} - a_{2ij}i^2)\sum_{n=i}^{j-1} y(n) + 2(2a_{2ij}i - a_{1ij})\sum_{n=i}^{j-1} ny(n)$$
$$-2a_{2ij}\sum_{n=i}^{j-1} n^2y(n) + \sum_{n=i}^{j-1} y(n)^2, \tag{5.11}$$

where

$$a_{0ij} = \frac{3}{(j-i)(j-i+1)(j-i+2)}\Bigg((3j^2-3j+6ij+i^2-3i+2)\sum_{n=i}^{j-1} y(n)$$
$$-2(6j+4i-3)\sum_{n=i}^{j-1} ny(n) + 10\sum_{n=i}^{j-1} n^2y(n)\Bigg), \tag{5.12}$$

$$a_{1ij} = \frac{6}{(j-i)(j-i+1)(j-i+2)(j-i-2)}\Bigg(\frac{1}{j-i-1}(-6j^3-14ij^2$$
$$+21j^2+16i^2j+18ij-21j+4i^3-9i^2-i+6)\sum_{n=i}^{j-1} y(n) + \frac{2}{j-i-1}$$
$$(16j^2-30j-2ij-14i^2+11)\sum_{n=i}^{j-1} ny(n) - 30\sum_{n=i}^{j-1} n^2y(n)\Bigg), \tag{5.13}$$

$$a_{2ij} = \frac{30}{(j-i)(j-i+1)(j-i-1)(j-i+2)(j-i-2)}\Bigg((j^2-3j+4ij$$
$$+i^2-3i+2)\sum_{n=i}^{j-1} y(n) - 6(j+i-1)\sum_{n=i}^{j-1} ny(n) + 6\sum_{n=i}^{j-1} n^2y(n)\Bigg) \tag{5.14}$$

Equations (5.11), (5.12), (5.13) and (5.14) are valid for the case where $j < N$. When $j = N$, we have to substitute $j+1$ for $j$. We will thus get similar expressions to Equations (5.11) - (5.14) for the case where $j = N$.

### 5.2.2 Encoding scheme

In general we need four parameters for each segment of the signal, three to define $f_{ij}$ and one to define its domain, in order to describe the piecewise

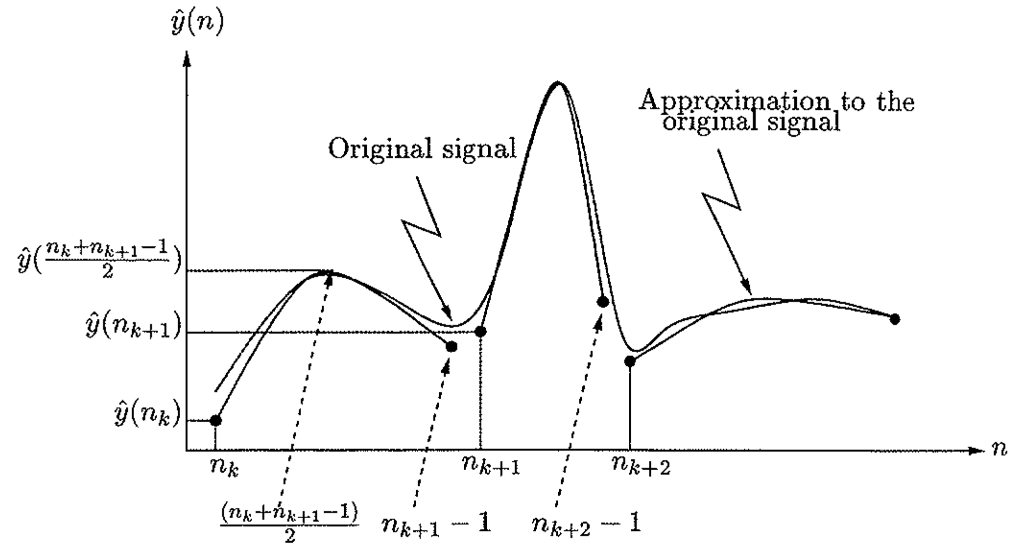


Figure 5.7: Example of a short sequence of original and reconstructed signal.

non-interpolating approximation with second order polynomials uniquely. We have several possible ways of representing these segments. We could encode the parameters $a_{0n_k,n_{k+1}}, a_{1n_k,n_{k+1}}, a_{2n_k,n_{k+1}}$ and the distance, $\delta_{n(k)} = n_{k+1} - n_k$ directly. However, this will lead to a need for four separate VLC's as each of these parameters have different domains. In addition the sensitivity with respect to quantization errors is poor in this representation form.

In the optimization algorithm described in Section 5.2.1, we represent each segment of the signal between vertices $n_k$ and $n_{k+1}$ by the approximated sample values $\hat{y}(n_k)$, $\hat{y}(n_{k+1}-1)$ and $\hat{y}(\frac{n_k+n_{k+1}-1}{2})$ in addition to the distance, $\delta_{n(k)} = n_{k+1} - n_k$, as illustrated in Figure 5.7. It is clear that these three interpolation points uniquely define $a_{0n_k,n_{k+1}}, a_{1n_k,n_{k+1}}$ and $a_{2n_k,n_{k+1}}$. This way we avoid the problem with the sensitivity regarding quantization error and we can apply one VLC for the three interpolation points.

We apply a simple predictive encoding scheme and encode the first order difference of the amplitudes. We thus have four parameters to be encoded for each segment of the signal:

$$\begin{aligned}
\delta_{n(k)} &= n_{k+1} - n_k, \\
\delta_{\hat{y}_1(k)} &= Q_1\left(\hat{y}\left(\frac{n_k + n_{k+1} - 1}{2}\right)\right) - Q_1\left(\hat{y}\left(n_k\right)\right),
\end{aligned}$$

$$\begin{aligned}\delta_{\hat{y}_2(k)} &= Q_2\left(\hat{y}\left(n_{k+1}-1\right)\right) - Q_2\left(\hat{y}\left(\frac{n_k+n_{k+1}-1}{2}\right)\right),\\ \delta_{\hat{y}_3(k)} &= Q_3\left(\hat{y}\left(n_{k+1}\right)\right) - Q_3\left(\hat{y}\left(n_{k+1}-1\right)\right),\end{aligned}$$

with $Q_k, k = 1,2,3$ denoting quantization. The choice of quantizer will be dependent on the scaling and sampling rate of the signal. In the experiments presented in Section 5.2.3 we use a uniform quantizer with quantization steps of size $3\Delta_Q$. In addition we need the absolute value of the first sample point. We apply separate VLC's for the runs and the quantized differential amplitudes, i.e., we use two different VLC's in this context and encode $\delta_{n(k)}$ by one encoder and $\delta_{\hat{y}_1(k)}, \delta_{\hat{y}_2(k)}, \delta_{\hat{y}_3(k)}$ by a different encoder.

### 5.2.3 Numerical experiments and discussion

We use the same evaluation criteria as before: The PRD distortion measure given in Equation (2.3), the maximum error given in Equation (2.1), visual inspection of the reconstructed signal and overview of execution times on a given machine.

For a description of the test signals, see Appendix A.

We compare the results from the FAN algorithm described in Section 2.2.1 and three different versions of the CCSP algorithm to the non-interpolating polynomial CCSP algorithm described on the preceding pages. The three versions of the CCSP algorithm included are the CCSP algorithm based on linear interpolation described in Section 4.1, the CCSP algorithm based on polynomial interpolation described in Section 4.3 and the CCSP algorithm based on non-interpolating linear approximation described in Section 5.1.

The FAN algorithm and the CCSP algorithm based on a linear interpolating approach approximate the signal by linear line segments represented by one extracted sample value and one position index per segment of the signal. These parameters are encoded by two separate VLC's as described in Section 4.1.2. The polynomial interpolating CCSP method and the linear non-interpolating CCSP method both approximate the signal by line segments, each of which is represented with three parameters. These parameters are encoded as described in Section 4.3.3 and Section 5.1.3, respectively.

We then have five different coders, representing the compressed signal in different ways. The FAN algorithm and the linear interpolating CCSP method use two parameters for each segment of the signal, the polynomial interpolating and linear non-interpolating CCSP algorithms, represent each segment of the signal with three parameters, and the polynomial non-interpolating optimization algorithm described on the preceding pages uses four parameters to represent each segment of the signal. It is clear that the more parameters we have, the higher the bit rate will be. However, the performance is a trade-off between rate and distortion, and the methods using more parameters in representing the signal may also be able to get a closer approximation to the original signal and thereby result in a higher overall performance.

### Evaluation based on the PRD measure

Figure 5.8 presents obtained PRD's for the coders for bit rates between 0.2 and 1.8 bits per sample (bps). The FAN method is outperformed by all the CCSP methods by a wide margin for nearly all test signals, except an area of test signal mit203_1100 for bit rates above 1 bps and an area of test signals mit207_1800 for bit rates below 0.35 bps where the FAN method performs better than the CCSP method based on linear non-interpolating approach. At low bit rates (around 0.6 bps) the FAN algorithm has from 50 % to 155 % higher PRD than the CCSP algorithm based on piecewise polynomial non-interpolating approximations. At higher rates (around 1.0 bps) the difference is smaller, but still significant.

Between the different versions of the CCSP algorithm, it seems like the CCSP method based on polynomial interpolation performs best for low bit rates (below 0.8 bps) while the CCSP method based on linear interpolation has best performance for higher bit rates, in terms of PRD.

### Evaluation based on the maximum error

Figure 5.9 presents obtained maximum errors for the different coders. We see that the FAN method performs best for all test signals and all bit rates. Incorporating the bound on maximum error as was done in Section 4.2, we can limit the maximum error.

Which of the CCSP methods that exhibits least maximum error is dependent on the particular signal and target bit rate. However, it seems as the CCSP method based on linear non-interpolating approach suffers from large maximum error for many test signals. The non-interpolating approaches may suffer from larger maximum errors than the interpolating ones due to end-of-block effects. If the signal has abrupt changes at the end of a block, this will be reflected in large maximum errors. This can be accounted for by taking signal characteristics into consideration before splitting the signal into blocks.

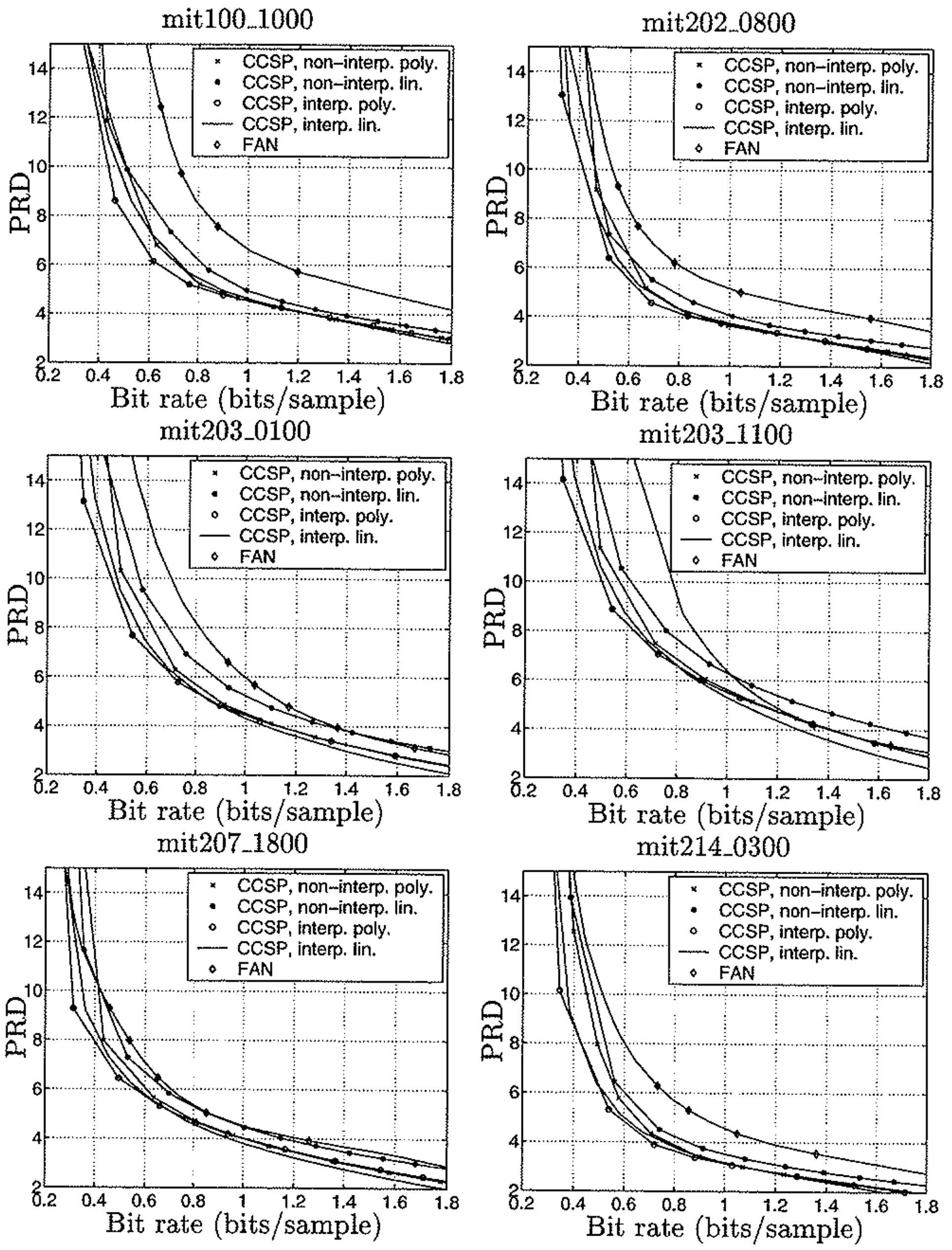


Figure 5.8: Coding performance for the different coders and test signals. Solid line with crosses: CCSP, non-interpolating, second order polynomials. Solid line with stars: CCSP, non-interpolating, linear. Solid line with circles: CCSP, interpolating, polynomial. Solid line: CCSP, interpolating, linear. Solid line with diamonds: FAN.

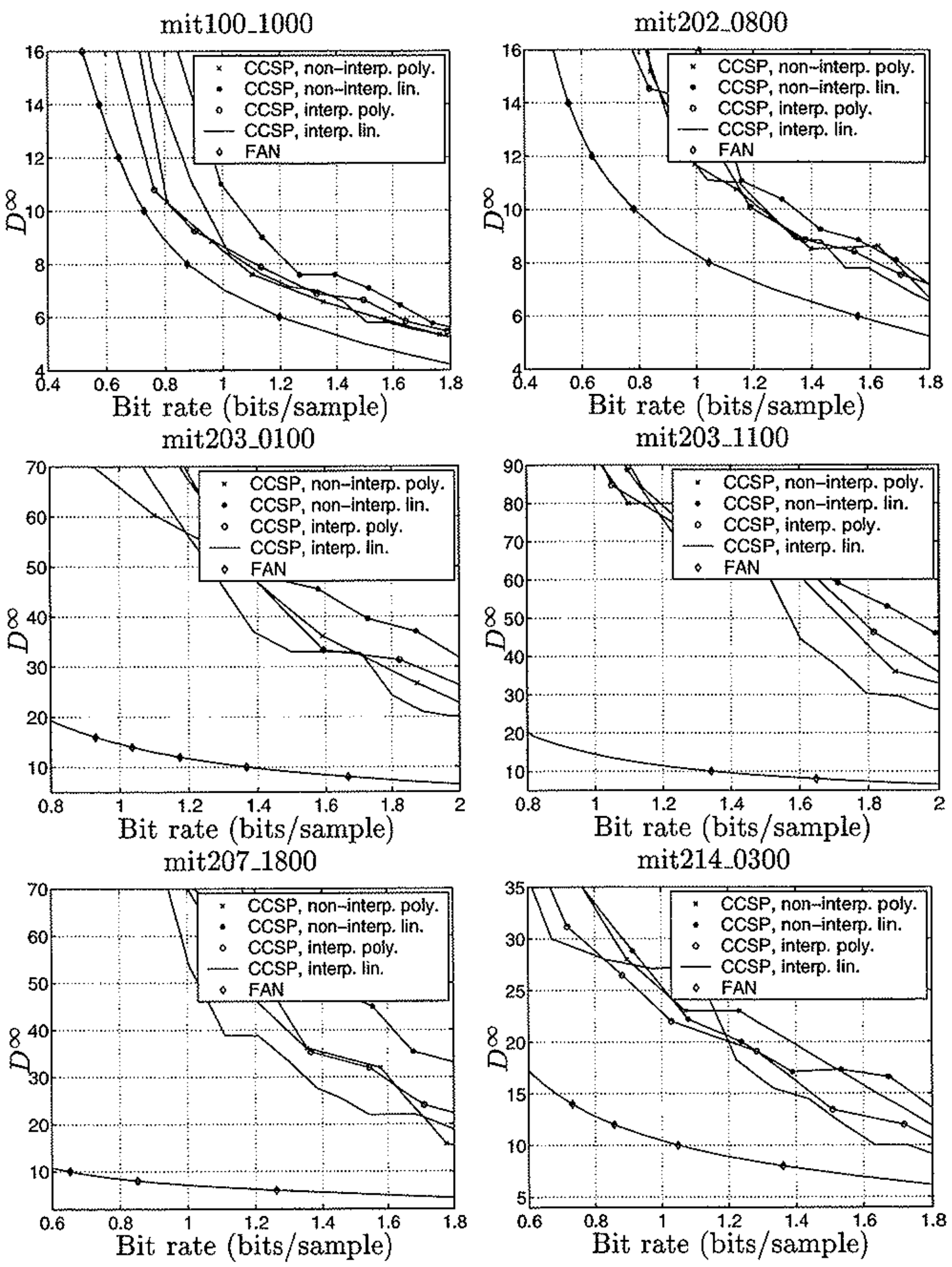


Figure 5.9: Maximum error, $D^\infty$, versus bit rate for the different coders and test signals. Solid line with crosses: CCSP, non-interpolating, second order polynomials. Solid line with stars: CCSP, non-interpolating, linear. Solid line with circles: CCSP, interpolating, polynomial. Solid line: CCSP, interpolating, linear. Solid line with diamonds: FAN.

### Evaluation based on visual inspection

Figures 5.10 and 5.11 show a short segment of test signal mit100_1000 reconstructed at bit rates of 1 bps and 0.5 bps, respectively. The original signal is also included. We see that the methods based on polynomial approximation, both the interpolating and non-interpolating approach, result in a reconstructed signal with less obvious straight lines. In the non-interpolating linear interpolating approach, the segmentation of the signal is obvious in the reconstructed signal, especially at low bit rates (see Figure 5.11).

Which compression method performs best in terms of producing the reconstructed signal with the highest fidelity based on visual inspection is hard to judge for an unexperienced eye. It will also be dependent on the application at hand.

### Evaluation based on execution time

Table 5.2 shows execution times for the CCSP algorithm based on polynomial non-interpolating approach for different block sizes and different number of extracted samples. The experiments are run on the same machine with the same test signal as in previous sections. From this table we conclude that the CCSP algorithm based on polynomial non-interpolating approach with a sampling frequency of 360 Hz, the real time requirements are not fulfilled. However, by optimizing the code and implementing the method described in Section 4.2 and/or the technique described in [38] this can be accounted for.

## 5.3 Summary

In this chapter we have shown how the optimal graph-theoretic approach presented in Chapter 3 can be further developed in order to release the interpolation restriction and apply a piecewise *non-interpolating* approach. This is done without increase in the computational complexity of the algorithm.

We develop two new coding strategies in this chapters

- An approach based on piecewise linear non-interpolating approximation of the signal.
- An approach based on piecewise second order non-interpolating approximation of the signal.

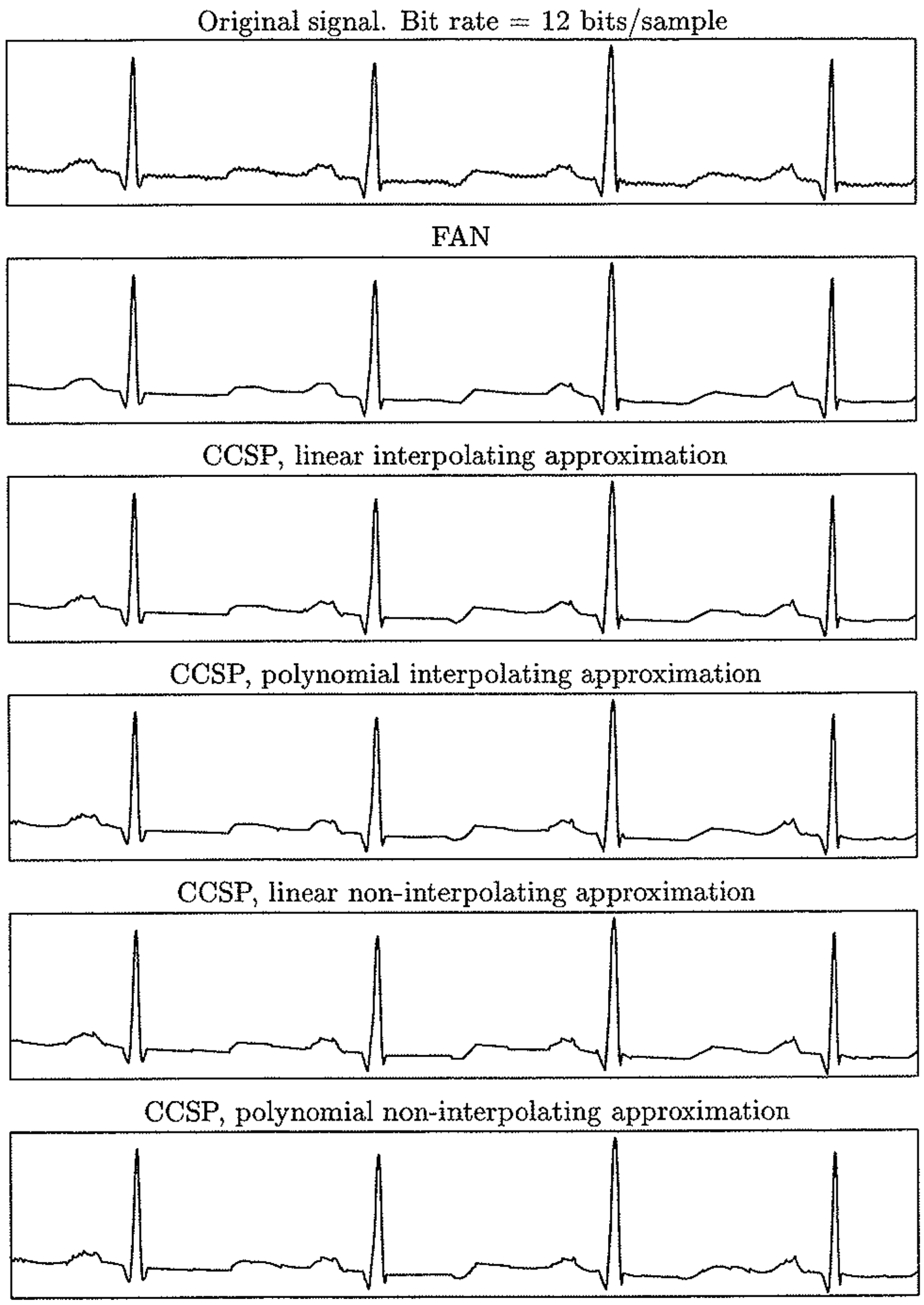


Figure 5.10: Short segment of reconstructed signal (taken from mit100_1000) at 1.0 bits per sample.

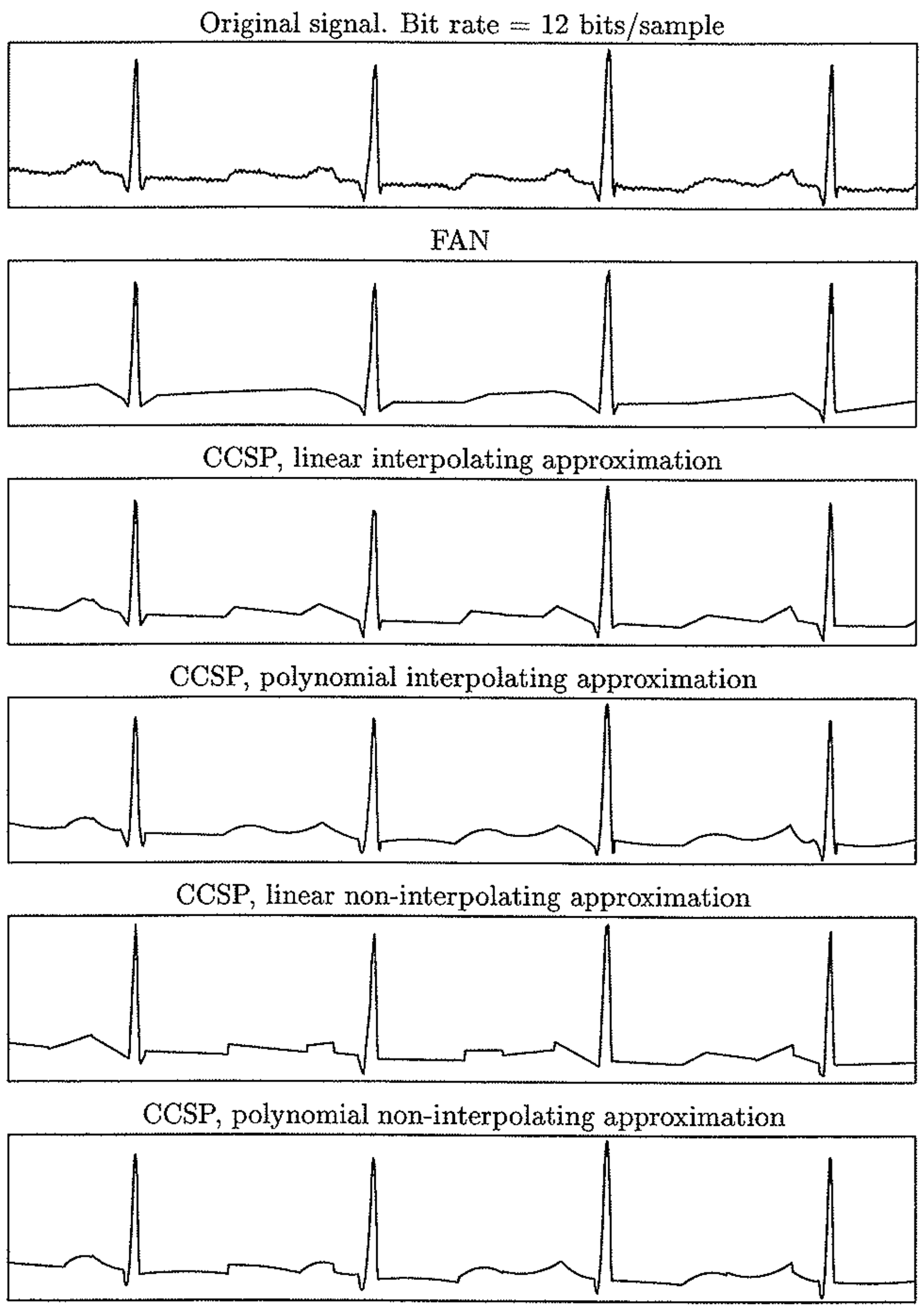


Figure 5.11: Short segment of reconstructed signal (taken from mit100_1000) at 0.5 bits per sample.

| **CCSP, polynomial non-interpolating approach** | | |
|---|---|---|
| Block size | Extracted samples, $M$ | Execution time in sec. |
| 100 | 5 | 0.30 |
| | 10 | 0.31 |
| | 30 | 0.38 |
| 200 | 5 | 1.20 |
| | 10 | 1.29 |
| | 30 | 1.62 |
| | 50 | 1.90 |
| 300 | 5 | 2.73 |
| | 10 | 2.92 |
| | 30 | 3.72 |
| | 50 | 4.42 |
| | 70 | 4.98 |
| 400 | 5 | 4.92 |
| | 10 | 5.24 |
| | 30 | 6.66 |
| | 50 | 8.00 |
| | 70 | 9.21 |
| | 90 | 10.29 |
| 500 | 5 | 7.76 |
| | 10 | 8.35 |
| | 30 | 10.60 |
| | 50 | 12.80 |
| | 70 | 14.73 |
| | 90 | 16.61 |

Table 5.2: Execution times for the CCSP algorithm based on polynomial non-interpolating approach run on an HP9000 C360.

Coding experiments show that the optimal time domain coders have significantly higher performance in terms of PRD than the traditional FAN algorithm. This is verified by visual inspection of the reconstructed signal. In terms of maximum error, the FAN method performs well. However, by incorporating the maximum error into the CCSP algorithm as was done in Section 4.2, the maximum error can be bounded.

Between the different CCSP methods developed in this dissertation, there are no obvious winner in terms of performance. The CCSP method based on polynomial interpolation generally performs best in terms of PRD for low bit rates. For higher bit rates, above approximately 1 bps, the CCSP method based on linear interpolation seems to perform better or at least similar to the polynomial interpolating approach. The linear non-interpolating CCSP approach cannot compete with the other CCSP methods in terms of PRD, maximum errors, execution times or by visual inspection of the reconstructed signal. The extra parameter necessary within this approach as compared to the interpolating one is too expensive when it comes to encoding, and we thus loose in the overall performance. The polynomial non-interpolating approach yields better performance than the linear non-interpolating one, and achieves in some cases, results similar to those of the interpolating approaches. However, considering the extra execution time for this algorithm as compared to the interpolating ones, the interpolating approaches still exhibit the highest overall performance. When it comes to execution times, the CCSP algorithm with the maximum error incorporated is superior to the other CCSP methods. Real time requirements are fulfilled for all block sizes for the sampling rate in our system for this version of the CCSP algorithm.

The FAN method is outperformed by the CCSP methods in terms of PRD, but it is superior in terms of maximum error. This is probably due to the fact that CCSP processes the signal on a block based approach, and is only allowed to extract an upper number of $M$ samples per block. This limits the flexibility of the algorithm. By allowing a variable number of samples to be extracted from each bloc as discussed in Section 4.1.3 this problem can be accounted for. Incorporating a bound on the maximum error into the algorithm as was done in Section 4.2 is another way of coping with this problem.

# Chapter 6

# Compression of image contours

In this chapter we develop our compression scheme further, to include the compression of image contours. This is an important problem in for instance the MPEG-4 standard [12]. The MPEG-4 standard has been developed through several stages and was conformance tested in November 1999. The discussion around the different approaches to shape coding in connection with this standard, indicates that this is a fruitful research area.

Section 6.1 gives a short introduction to the area of 2-D shape coding. In Section 6.2 we define the problem we aim to solve in a precise mathematical manner. Section 6.3 is devoted to the solution method and finally in Section 6.5 we report some numerical experiments.

## 6.1 Background

A *digital image* is an image which has been discretized both in spatial coordinates and brightness. Such an image can be considered a matrix whose row and column indices identify a point in the image and the corresponding matrix element value identifies the gray level at that point. The elements of such a matrix are called *image elements*, *picture elements*, *pixels* or *pels*.

An enormous amount of data is produced when a 2-D light intensity function is sampled and quantized to create a digital image. It is impractical to deal with these amounts of data for storage, processing and communication requirements. Image compression is therefore a huge research area, and interest in this area dates back more than 30 years. It has developed from bandwidth compression by the means of analog methods in the early years to todays efficient digital compression standards.

There are many ways to approach the problem of image compression. We may view the image as a "black box" without considering what objects the image contains in particular or without discriminating between different regions of the image. Many image compression algorithms are based on this point of view like the JPEG standard [48]. However, with new applications like digital libraries and content-based storage and retrieval, new approaches to image compression have evolved. Second generation image coding techniques segment an image into regions and describe each region by texture and shape [53]. Hence, the problem of representing the shape of an object in an efficient way arises. This is the problem we are concerned with in this chapter.

A significant part of the literature on shape coding deals with coding of binary shapes, i.e., images where all pixels are classified as belonging to one of the two categories *object* or *background*. There are two classes of 2-D binary shape coders [50]:

1. Bitmap-based coders where all pixels are encoded whether they belong to an object or not.

2. Contour-based coders which encode the outline of an object.

We will focus on category 2 and develop an approach for compression of a given contour, assuming that the contour is given to us. We will not be concerned with the extraction of a contour from a given image.

There has been significant research activity in the area of compressing digital planar curves, as documented for example in [40], [57], and [86]. The methods described in these papers are either based upon some heuristics or they suffer from restrictions making them suboptimal in some sense. As opposed to this, we present a method that guarantees the smallest reconstruction error possible when applying linear interpolation, given the number of retained curve points.

Many algorithms for data compression are based on the idea of extracting a subset of points from the original data and linearly interpolating among them in reconstruction of the curve. Which data points to be extracted depend on the underlying criterion for the point selection process. To get a high performance compression algorithm, much effort should be put into designing intelligent point selection strategies.

In this context the points of the original curve are modeled as nodes in a directed graph (digraph). Any pair of nodes are connected with an arc, the direction of which is given from the point order. We label the starting point of our algorithm 1, and succeeding points are numbered in increasing order. In the case of a closed contour, the first point is identical to the last one. We will associate a subset of selected points with a *path* in the graph. Including a particular arc in the path corresponds to letting the end nodes of the arc constitute consecutive points in the extracted subset of points. We thus assume that the contours to be compressed are given as an ordered set of discrete points.

The length of each arc in the digraph can be defined in a variety of ways. Here the length of the arc connecting two points $(x_i, y_i)$ and $(x_j, y_j)$ is defined as the contribution to the reconstruction error from eliminating all points recorded between $(x_i, y_i)$ and $(x_j, y_j)$, and approximating them with linear interpolation. Defining the problem in this way, minimization of the reconstruction error can be recognized as solving the cardinality constrained shortest path problem defined on the graph.

Applying linear interpolation in the reconstruction phase, the algorithm in Section 4.1 is proven to converge in cubic time. In [38] it is demonstrated how to implement the algorithm in a computationally efficient way. We will now show that the idea presented in Section 4.1, can be applied to the problem of compressing planar curves without increasing the computational complexity of the original algorithm.

## 6.2 Optimization model

Denote the points constituting a planar curve by $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$. Let each point be denoted by $\mathbf{p}_k = (x_k, y_k)$, $k = 1, 2, \ldots, N$. Let $M$ denote the bound on the number of extracted points and $S$ denote the *point set* $S = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_N\}$. We seek an appropriate *compression set* $C = \{n_1, n_2, \ldots, n_M\}$ and the corresponding points $K = \{\mathbf{p}_{n_1}, \mathbf{p}_{n_2}, \cdots, \mathbf{p}_{n_M}\} \subseteq S$.

As is illustrated in Figure 6.1, the approximation is given by

$$(\hat{x}_n, \hat{y}_n) = \begin{cases} (x_n, y_n) & \text{if } n \in C, \\ (x_{n_k} + \Delta x_n, y_{n_k} + \Delta y_n) & \text{if } n \notin C. \end{cases} \tag{6.1}$$

The quantities $\Delta x_n$ and $\Delta y_n$ are the $x$-component and $y$-component, respectively, of the Euclidean distance between $(x_{n_k}, y_{n_k})$ and $(\hat{x}_n, \hat{y}_n)$. In this way we get a piecewise linear approximation to the original curve.

Define the directed graph $G = (V, A)$ whose *vertex set* $V = \{1, 2, ..., N\}$ and *arc set* $A = \{(i, j), i, j \in V, i < j\}$. The set $(n_1, n_2, \ldots, n_M)$ is said to be a

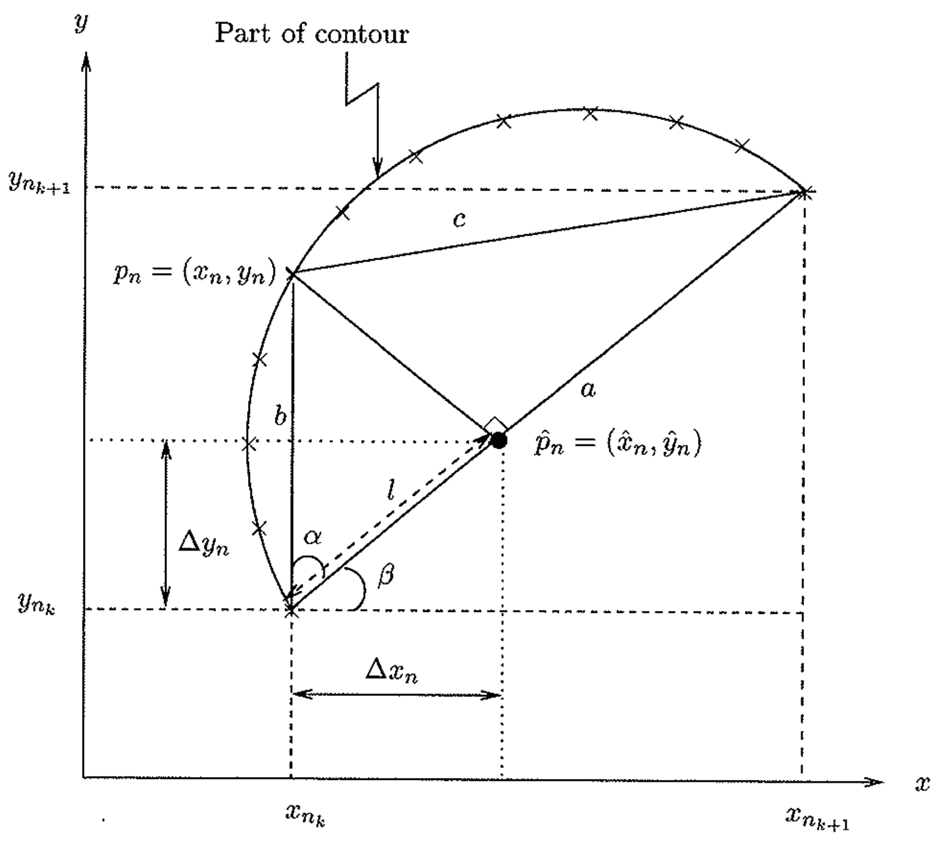


Figure 6.1: Linear interpolation of a point on a contour based on two other points.

*path* from $n_1$ to $n_M$ in $G$ if $n_1, \ldots, n_M \in V$ are distinct vertices and $n_1 < n_2 < \cdots < n_M$. Each arc $(i, j) \in A$ represents the possibility of letting $\mathbf{p}_i$ and $\mathbf{p}_j$ be consecutive members of $K$. Hence we actually explore $G$ for a suitable path from $n_1$ to $n_N$. The set of vertices traversed by this path will constitute $C$. The length of each arc $(i, j) \in A$ is given as the contribution to the total reconstruction error by eliminating all curve points between $\mathbf{p}_i$ and $\mathbf{p}_j$, approximating them with linear interpolation, and evaluating the Euclidean distance between the original and reconstructed curve points. This can be expressed as

$$d_{ij} = \sum_{n=i+1}^{j-1} \|\hat{\mathbf{p}}_n - \mathbf{p}_n\|^2. \tag{6.2}$$

Denoting the path from vertex 1 up to vertex $n$ by $P_n$, the length of this path will be given as the sum of the length of all arcs included in the path up to vertex $n$.

Hence we are faced with the following problem : Minimize the length of $P_N$ under the constraint that $P_N$ contains no more than $M$ vertices. With the problem at hand here, this is a modified version of the problem presented in Section 4.1.

## 6.3 Solution method

We now go on to show that the algorithm in Chapter 3 can be modified such that it handles planar curves and still preserves its computational complexity of $\mathcal{O}(MN^2)$. As before, $M$ is the bound on the number of extracted curve points and $N$ is the number of original curve points.

To be able to extract points from $S$ in an optimal way, we need to know the contribution to the reconstruction error introduced by including any two points as consecutive members of $K$. The aim now is therefore to compute the reconstruction error introduced by letting any two points be consecutive members of $K$. We consider the point set $S$ an ordered set and only allow two points $\mathbf{p}_i$ and $\mathbf{p}_j$ to be consecutive members of $K$ if $i < j$.

The graph $G$ consists of $\frac{N(N-1)}{2}$ arcs. From Equation (6.2) it is seen that the expression for $d_{ij}$ is a sum of $j-i-1$ terms. Straightforward computation of all these arc lengths will thus result in an algorithm with a complexity of $\mathcal{O}(N^3)$. Fortunately, this can be avoided by careful computation of the arc lengths. The arc lengths are given by

$$\begin{aligned} d_{ij} &= \sum_{n=i+1}^{j-1} \|\hat{\mathbf{p}}_\mathbf{n} - \mathbf{p}_\mathbf{n}\|^2 \qquad (6.3)\\ &= \sum_{n=i+1}^{j-1} \|(\hat{x}_n, \hat{y}_n) - (x_n, y_n)\|^2 \\ &= \sum_{n=i+1}^{j-1} \left((\hat{x}_n - x_n)^2 + (\hat{y}_n - y_n)^2\right) \\ &= \sum_{n=i+1}^{j-1} (\hat{x}^2 - 2x_n\hat{x}_n + x_n^2 + \hat{y}^2 - 2y_n\hat{y}_n + y_n^2). \end{aligned}$$

In general, indices $n_k$ and $n_{k+1}$ can be denoted as $i$ and $j$, respectively. The expressions for $\hat{x}_n$ and $\hat{y}_n$ are then given by

$$\hat{x}_n = x_i + \Delta x_n, \qquad (6.4)$$

and

$$\hat{y}_n = y_i + \Delta y_n, \tag{6.5}$$

which can easily be seen from Figure 6.1. By looking at Figure 6.1 we can obtain expressions for $\Delta x_n$ and $\Delta y_n$ by the use of some elementary trigonometry. We observe that

$$\Delta x_n = l\cos\beta, \tag{6.6}$$
$$\Delta y_n = l\sin\beta. \tag{6.7}$$

The expressions for $\cos\beta$ and $\sin\beta$ can be deduced directly by looking at Figure 6.1

$$\cos\beta = \frac{x_j - x_i}{\sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}}, \tag{6.8}$$
$$\sin\beta = \frac{y_j - y_i}{\sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}}. \tag{6.9}$$

We then need an expression for $l$. From the cosine theorem we know that

$$c^2 = a^2 + b^2 - 2ab\cos\alpha,$$

and thus

$$\cos\alpha = \frac{a^2 + b^2 - c^2}{2ab}. \tag{6.10}$$

By looking at Figure 6.1 we observe that

$$l = b\cos\alpha = \frac{a^2 + b^2 - c^2}{2a}, \tag{6.11}$$
$$a^2 = (x_j - x_i)^2 + (y_j - y_i)^2, \tag{6.12}$$
$$b^2 = (x_n - x_i)^2 + (y_n - y_i)^2, \tag{6.13}$$
$$c^2 = (x_j - x_n)^2 + (y_j - y_n)^2. \tag{6.14}$$

Inserting Equations (6.10), (6.12), (6.13) and (6.14) into Equation (6.11) we arrive at

$$l = \frac{(x_j - x_i)^2 + (y_j - y_i)^2 + (x_n - x_i)^2 + (y_n - y_i)^2 - (x_j - x_n)^2 - (y_j - y_n)^2}{2\sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}}. \tag{6.15}$$

By inserting Equations (6.8), (6.9) and (6.15) into Equations (6.6) and (6.7) we find the expressions for $\Delta x_n$ and $\Delta y_n$ we are searching for. We can then insert these expressions for $\Delta x_n$ and $\Delta y_n$ into Equations (6.4) and (6.5) and obtain the following expressions for $\hat{x}_n$ and $\hat{y}_n$:

$$\begin{aligned}
\hat{x}_n &= \\
x_i &+ \frac{x_i^2 - x_i x_j + (x_j - x_i)x_n + y_i^2 - y_i y_j + (y_j - y_i)y_n}{(x_j - x_i)^2 + (y_j - y_i)^2}(x_j - x_i), \quad (6.16)\\
\hat{y}_n &= \\
y_i &+ \frac{x_i^2 - x_i x_j + (x_j - x_i)x_n + y_i^2 - y_i y_j + (y_j - y_i)y_n}{(x_j - x_i)^2 + (y_j - y_i)^2}(y_j - y_i). \quad (6.17)
\end{aligned}$$

We have thus arrived at expressions for $\hat{x}_n$ and $\hat{y}_n$ as functions of $x_i$, $x_j$, $y_i$, $y_j$, $x_n$ and $y_n$. By substituting the expressions for $\hat{x}_n$ and $\hat{y}_n$ into the expression for $d_{ij}$ in Equation (6.3), we arrive at an expression of the form

$$\begin{aligned}
d_{ij} = {} & \zeta_{1ij} \sum_{n=i+1}^{j-1} x_n^2 + \zeta_{2ij} \sum_{n=i+1}^{j-1} x_n + \zeta_{3ij} \sum_{n=i+1}^{j-1} x_n y_n \\
& + \zeta_{4ij} \sum_{n=i+1}^{j-1} y_n + \zeta_{5ij} \sum_{n=i+1}^{j-1} y_n^2 + \kappa_{ij},
\end{aligned}$$

where

$$\begin{aligned}
\zeta_{1ij} &= \frac{(y_j - y_i)^4 + (x_j - x_i)^2(y_j - y_i)^2}{\left((x_j - x_i)^2 + (y_j - y_i)^2\right)^2},\\
\zeta_{2ij} &= 2\frac{y_i(x_j - x_i)^3(y_j - y_i) + \left(x_i^2 - x_i x_j - y_i^2 + y_i y_j\right)(x_j - x_i)(y_j - y_i)^2}{\left((x_j - x_i)^2 + (y_j - y_i)^2\right)^2}\\
&\quad - \frac{x_i(y_j - y_i)^4}{\left((x_j - x_i)^2 + (y_j - y_i)^2\right)^2},\\
\zeta_{3ij} &= -2\frac{(x_j - x_i)^3(y_j - y_i) + (x_j - x_i)(y_j - y_i)^3}{\left((x_j - x_i)^2 + (y_j - y_i)^2\right)^2},\\
\zeta_{4ij} &= 2\frac{x_i(x_j - x_i)(y_j - y_i)^3 + \left(y_i^2 - y_i y_j - x_i^2 + x_i x_j\right)(x_j - x_i)^2(y_j - y_i)}{\left((x_j - x_i)^2 + (y_j - y_i)^2\right)^2}\\
&\quad - \frac{y_i(x_j - x_i)^4}{\left((x_j - x_i)^2 + (y_j - y_i)^2\right)^2},\\
\zeta_{5ij} &= \frac{(x_j - x_i)^4 + (x_j - x_i)^2(y_j - y_i)^2}{\left((x_j - x_i)^2 + (y_j - y_i)^2\right)^2},
\end{aligned}$$

$$\kappa_{ij} = (j-i-1)\frac{\left(x_i(y_j-y_i)^2+(x_j-x_i)y_i^2-y_iy_j(x_j-x_i)\right)^2}{\left((x_j-x_i)^2+(y_j-y_i)^2\right)^2}$$
$$+(j-i-1)\frac{\left(y_i(x_j-x_i)^2+(y_j-y_i)x_i^2-x_ix_j(y_j-y_i)\right)^2}{\left((x_j-x_i)^2+(y_j-y_i)^2\right)^2}.$$

Each of the coefficients $\zeta_{kij}$, $k = 1,\dots,5$, and $\kappa_{ij}$ depend only on $x_i$, $x_j$, $y_i$ and $y_j$, and computing all of them can be accomplished by $\mathcal{O}(N^2)$ operations. Computing the sums $\sum_{n=i+1}^{j-1} x_n^2$, $\sum_{n=i+1}^{j-1} x_n$, $\sum_{n=i+1}^{j-1} x_n y_n$, $\sum_{n=i+1}^{j-1} y_n$ and $\sum_{n=i+1}^{j-1} y_n^2$ for all legal combinations of $i$ and $j$ is performed in a recursive manner, applying the same procedure as described in Section 4.1.1, and are thus accomplished in $\mathcal{O}(N^2)$ time. Hence the arc lengths $d_{ij}$ are available in $\mathcal{O}(N^2)$ time.

When all arc lengths are available, the actual extraction of contour points takes place. This is accomplished by the dynamic programming algorithm thoroughly described in Section 3.4. The computational complexity of the total CCSP compression algorithm for image contours is $\mathcal{O}(MN^2)$.

## 6.4 Coding scheme

Recall that the curve points extracted by the CCSP algorithm are $\mathbf{p}_k$, $k = 1,\dots,M$. Since the extracted curve points are along a natural boundary, consecutive extracted points will be closely spaced. This indicates that there is a profit in encoding the points by a predictive scheme. In this context we use a simple predictive scheme in which the differences between the consecutively extracted points are encoded. We denote these differences by $\delta x_k = x_{n_{k+1}} - x_{n_k}$ and $\delta y_k = y_{n_{k+1}} - y_{n_k}$. In the case of a closed contour, the last extracted point is discarded as this is equal to the first extracted point. When encoding the difference between the extracted curve points, we need to encode the first absolute point position as well.

We have several choices when encoding $\delta x_k$ and $\delta y_k$, $k = 1,\dots,M-1$. We may either use a VLC approach, or we could apply a fixed-length codeword scheme. In Section 6.5 we present results from both the VLC and fixed-length codeword encoding scheme. The results from these two schemes will constitute performance limits for the total compression system. The structure of the total encoding system is illustrated in Figure 6.2.

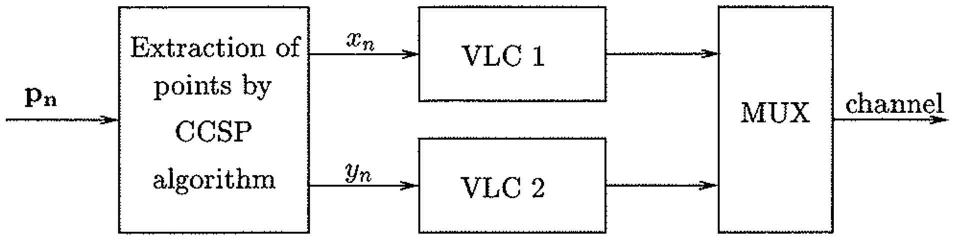


Figure 6.2: Structure of encoding system.

### 6.4.1 VLC encoding scheme

In encoding the $\delta x_k$ and $\delta y_k$, $k = 1, \ldots, M-1$, by a VLC, we have to choose a strategy by which to generate the VLC table. We may apply a fixed VLC table, in which case the table is generated once and for all in advance. We also have the choice of applying an adaptive VLC scheme, where the VLC tables are designed to match the contour at hand perfectly. In this case we have to consider side information due to the VLC tables which will also have to be transmitted.

In our experiments in Section 6.5 we have chosen to consider the VLC tables to be known by both encoder and decoder. Consequently we quote calculated entropies. This will constitute a best-case performance for the coder.

### 6.4.2 Fixed-length codeword scheme

In the fixed-length codeword case, we are not concerned with the probabilities of occurrence of the different source symbols. Instead we use a fixed number of bits in encoding each of the $\delta x_k$ and $\delta y_k$, $k = 1, \ldots, M-1$.

## 6.5 Numerical experiments and discussion

For quantitative evaluation of the performance of the coder described in Section 6.4, the *Mean Square Error* is applied:

$$MSE = \frac{1}{N} \sum_{n=1}^{N} \|\mathbf{p}_n - \hat{\mathbf{p}}_n\|^2, \tag{6.18}$$

where $\mathbf{p}_n$ and $\hat{\mathbf{p}}_n$ denotes the original and reconstructed curve points, respectively, and $N$ is the total number of original curve points. As the MSE hardly qualifies as an authorative yardstick, it is supplemented by visual inspection of the reconstructed boundaries.

The performance of the coding scheme is measured in *bits per boundary point* (bpp). This is calculated by dividing the total number of bits needed to encode the boundary approximation by the number of original boundary points.

Several contours were used in the coding experiments. Here we present two test contours. The first is the contour of Australia and is shown in Figure 6.4. This contour consists of 1708 points originally. The second is an extracted area of a frame of the Miss America sequence and is shown in Figure 6.6. This is the same contour used in [86]. It consists of 90 points originally.

### 6.5.1 Evaluation based on the MSE

The rate-distortion curve for the two test contours are shown in Figure 6.3. The solid line indicates the results by VLC encoding the extracted curve points. Side information has not been taken into consideration here, and this curve will thus serve as a best-case limit for the performance of the total encoding scheme. The dotted line is the result from applying a fixed-length codeword scheme to the extracted points. This constitutes a worst-case performance limit for the coder.

### 6.5.2 Evaluation based on visual inspection

Visual inspection of the reconstructed curves is important in evaluating the performance of the coders. Figure 6.4 shows the original test contour 1. In Figure 6.5 the reconstructed contour at a bit rate of 0.5 bpp is shown. The encoding technique applied here is fixed-length codeword scheme. Each of the parameters $\delta x_k$ and $\delta y_k$ is encoded using 7 bits, which means 14 bits per point.

Figure 6.6 shows test contour 2. The original contour is denoted as stars, and the solid line indicates the reconstructed contour at a bit rate of 1.5 bpp. Each of the parameters $\delta x_k$ and $\delta y_k$ is encoded using 5 bits in this case, which means 10 bits per point.

In the case of a closed contour we have to choose a starting point for our algorithm. Different starting points may lead to different results. However, experiments show that for the test contours we have applied, different starting points will only lead to marginal differences in terms of bit rates. For this reason we have chosen not to emphasize the problem associated with selecting a starting point.

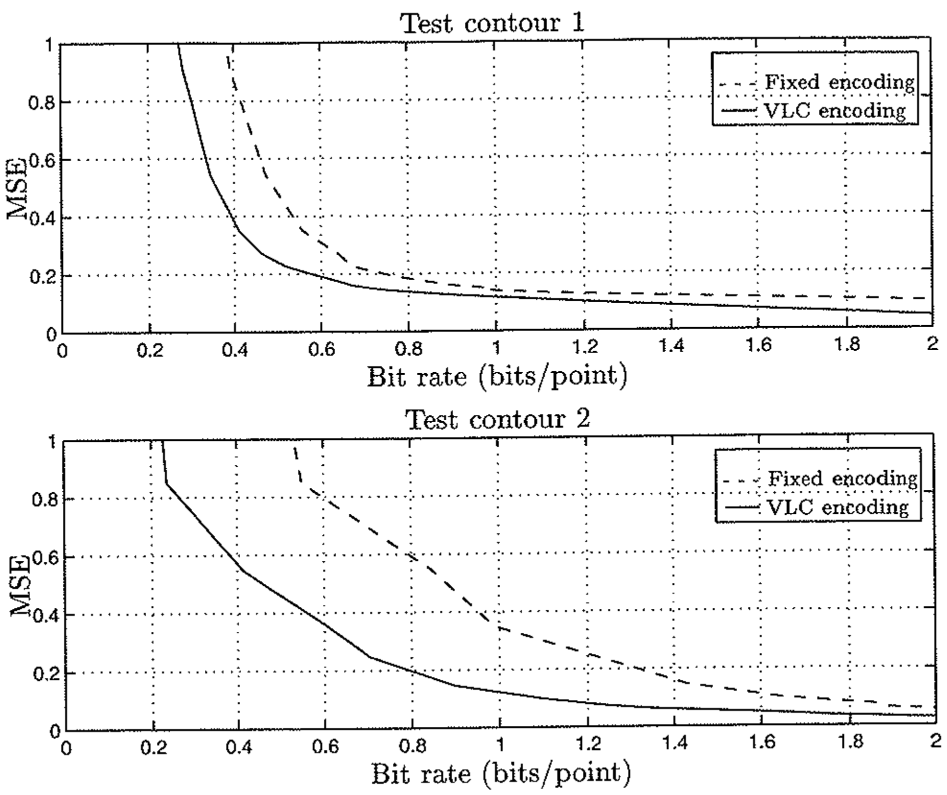


Figure 6.3: MSE as a function of bit rate for two test contours.

The complexity of the algorithm presented in this chapter is $\mathcal{O}(MN^2)$. When it comes to execution time, it will be approximately the same as for the CCSP algorithm based on the linear interpolating approach presented in Section 4.1. Whether or not this satisfies real time constraints depends on the application. If it is to be used within digital libraries the real time constraints will be less tight than for instance in a real time video compression system.

In the version of the contour compression scheme presented in this chapter we have applied linear interpolation among the extracted signal samples. It is possible to replace this with higher order polynomials without increasing the computational complexity of the algorithm by using the method described in Section 4.3. Applying higher order polynomials will smooth out the obvious straight lines visible in the reconstructed contours in Figures 6.5 and 6.6. It is also possible to incorporate multiple error measures by using the method described in Section 4.2.

In order to evaluated the results obtained with the compression scheme pre-

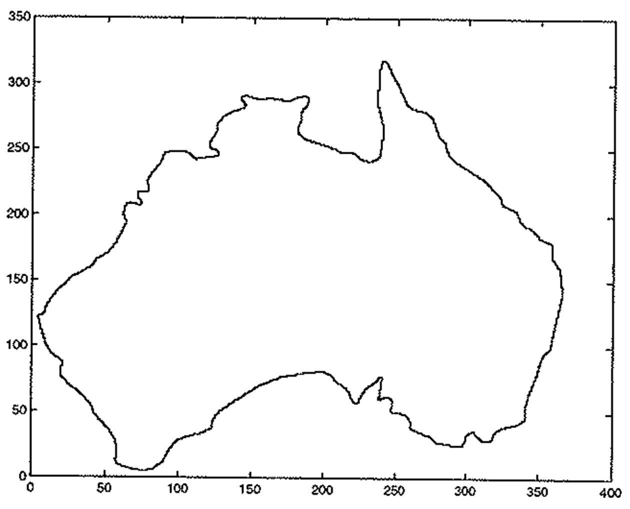


Figure 6.4: The original test contour 1.

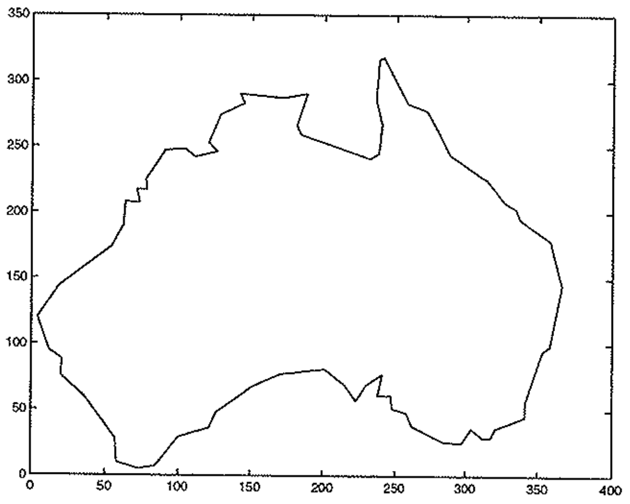


Figure 6.5: Reconstructed test contour 1 at a bit rate of 0.5 bpp.

sented in this chapter, it would be interesting to do an inter-method comparison with other compression schemes for image contours. In [86] several approaches are presented which approximate a boundary by a polygon and consider the problem of finding the polygon which leads to the smallest distortion for a given number of bits. In the experimental section of [86], it is reported that test contour 2 is compressed to a rate of 1.88 bpp at an MSE of 0.1 and to a bit rate of 2.22 bpp at an MSE of 0.05. As compared to our

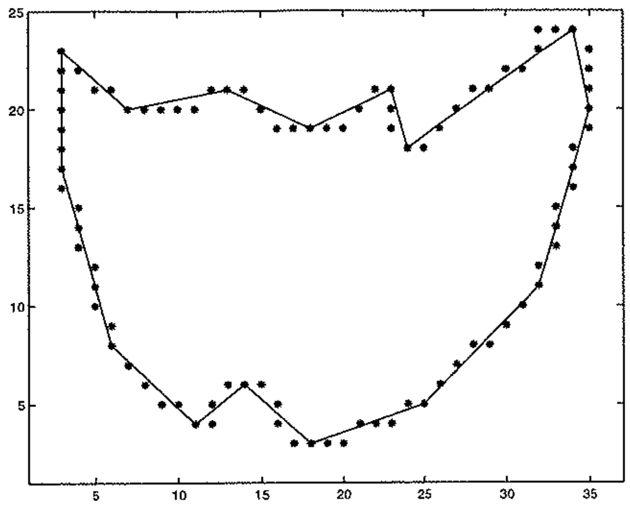


Figure 6.6: Test contour 2, both original and reconstructed.

method, the same test contour at an MSE of 0.1 reaches a bit rate of 1.09 bpp and at an MSE of 0.05, a rate of 1.63 bpp is obtained. In other words our method is significantly better than these approaches for the reported bit rates.

## 6.6 Summary

In this chapter it is demonstrated how the optimal graph-theoretic approach presented in Chapter 3 can be further developed in order to compress planar curves in an optimal way with respect to a given distortion measure. This is done without increase in the computational complexity of the algorithm. An approach is developed in which we approximate the original contour by applying straight lines between a number of extracted boundary points. These extracted boundary points are fitted in an optimal way with respect to reconstruction error. We present a solution scheme resulting in an approach with a complexity which is no higher than cubic in the number of curve points.

Our compression algorithm is based on combinatorial optimization theory. By the very nature of our approach the distortion is guaranteed to be the smallest possible of all techniques using linear interpolation, given the number of retained curve points.

Coding experiments show that the optimal time domain compression method presented in this chapter gives very good results. It is not straightforward to compare different encoding techniques to each other. However, when comparing the results obtained by our method to the ones in [86], it is evident that our algorithm performs significantly better in terms of achieving lower bit rates at the same distortion.

# Chapter 7

# Rate distortion optimal time domain signal compression

The ECG encoding techniques presented in Chapters 3, 4 and 5 compress ECG signals by representing them by signal samples that, after reconstruction, will best represent the original signal given an upper bound on their number. After the approximated samples are found they are encoded by a Variable Length Code (VLC) approach. This leads to the best possible representation in terms of the number of signal samples used in the approximation, but not necessarily in terms of bits used to encode such samples. In this chapter we modify the problem definition, taking the bit rate into consideration in the optimization process [70, 71]. We thus solve the problem of minimizing the distortion of the reconstructed signal given an upper bound on the number of bits. The resulting solution is optimal in the operational rate distortion (ORD) sense. Given the structure of the coder, no other technique based on linear interpolation will give a lower distortion for the same bit rate. In addition, we apply an iterative procedure to find the underlying parameter probability distribution resulting in the locally most efficient ORD curve. Similar techniques, with and without VLC optimization, have been used for compression of image contours [86, 88, 58, 59, 87].

As opposed to the ECG compression algorithms introduced in the previous chapters, which are all based on minimization of the same error measure, namely the sum of squared errors [1], we introduce two different solution algorithms in this chapter: The minimum maximum (*min max*) algorithm, where we minimize the maximum error, and the minimum average (*min ave*) algorithm, in which we minimize the average error. We could use both of the

[1]This does not apply to the FAN algorithm which limits the maximum error.

error measures given in Equations (2.1) and (2.2) with both of the solution algorithms as we have done in previous chapters, and thus have four different solutions. However, we have chosen to combine the maximum error measure with the *min max* algorithm and the sum of squared errors with the *min ave* algorithm, and we thus present results from two different solution schemes in this chapter. We compare the results from the two schemes both in terms of Percentage Root-mean-square Difference (PRD), maximum error, visual performance and execution time.

This chapter is organized as follows: In the next section the problem is defined mathematically. We introduce the two philosophies that we apply to the segment distortions: The maximum error and the sum of squared errors, and discuss bit efficient representation of the retained samples. For each of the two error measures, we introduce efficient solution methods in Section 7.2. We introduce a shortest path solution method and show how the Lagrangian multiplier method can be applied to solve our constrained problem as a series of unconstrained problems in the *min ave* case, and as a single unconstrained problem in the *min max* case. Finally, in Section 7.3, experimental results are reported and discussed, before we summarize in Section 7.4.

## 7.1 Problem formulation

In the following we introduce the notation used throughout this chapter. We define the distortion measures and introduce an efficient way to count bits before we go on defining the problem in mathematical terms.

Denote the *set of sample points* taken from a signal at constant time intervals by $S = \{(1, y(1)), \dots, (N, y(N))\}$. Define the set of *admissible points* by $Y = \{(n, y(n,l)), n = 1, \dots, N; l = -p, \dots, p\}$, where $y(n,-p), \dots, y(n,p)$ are evenly distributed signal values with $y(n,0) = y(n)$ being the median. Denote the cardinality of $Y$ by $N_Y = N(2p+1)$. Let $y(n,1) - y(n,0) = \Delta_Q$ where $\Delta_Q$ equals the quantization step size of the quantizer applied to the original signal. We seek a compression set $C = \{n_1, \dots, n_M\} \subseteq \{1, \dots, N\}$, the cardinality $M$ of $C$, as well as integers $l_1, \dots, l_M \in \{-p, \dots, p\}$.

The above definitions suggest that the original sample set $S$ be replaced by the points $\hat{S} = \{(n_k, y(n_k, l_k)), k = 1, \dots, M\}$ implying a sample reduction ratio equal to $N/M$. Since $S \subseteq Y$, the set $Y$ represents an extension over $S$ as the set of points used to approximate the signal (cf. the method given in Chapter 3).

To guide the selection of $C$ and $l_1, \dots, l_M$, we assume a signal reconstruction based on straight lines interpolating $\hat{S}$. This could easily be extended to curves

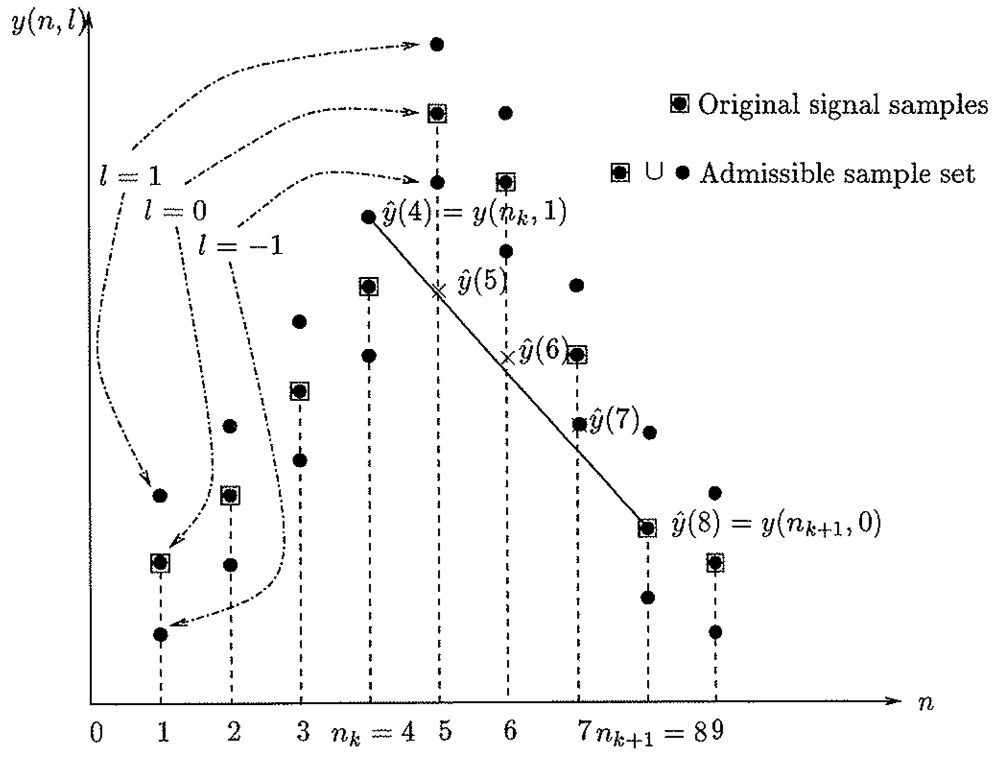


Figure 7.1: The original signal and the admissible sample set.

of higher order [87]. The approximation to the $n$th sample value is thus given by

$$\hat{y}(n) = y(n_k, l_k) + \frac{y(n_{k+1}, l_{k+1}) - y(n_k, l_k)}{n_{k+1} - n_k}(n - n_k) \quad \text{for } n_k \leq n \leq n_{k+1}. \tag{7.1}$$

Hence all samples made in the time interval $[n_k, n_{k+1}]$ are restored by linear interpolation between $(n_k, y(n_k))$ and $(n_{k+1}, y(n_{k+1}))$. This is illustrated in Figure 7.1 where we approximate the samples at time indices $n = \{4, \dots, 8\}$ by the straight line starting in $n_k = 4$ and ending in $n_{k+1} = 8$. In this case, $y(n_k, 1)$ belongs to $Y$, but *not* to $S$, while $y(n_{k+1}, 0)$ belongs to both $Y$ and $S$, which means it is one of the original signal samples.

### 7.1.1 Distortion measure

We already defined the maximum error distortion measure and sum of squared distortion measure in Equations (2.1) and (2.2), respectively. However, introducing the admissible point set, $Y$, corresponds to increasing the problem from one to two dimensions. We thus need to fit the distortion measures to this problem formulation.

Consider two arbitrary admissible sample points $(n, y(n,l)), (n', y(n',l')) \in Y$ where $n < n'$. The maximum absolute distance between original and restored signal implied by accepting these as consecutive members of $\hat{S}$, e.g. by letting $n_k = n, n_{k+1} = n', l_k = l, l_{k+1} = l'$, is given by

$$d^{\infty}_{nln'l'} = \max_{t \in [n_k, n_{k+1}]} |\hat{y}(t) - y(t)| . \tag{7.2}$$

Given the same sample points as above, the sum of squared distances between the original and the reconstructed signal is given by

$$d_{nln'l'} = \begin{cases} \sum_{t=n_k}^{n_{k+1}-1} (\hat{y}(t) - y(t))^2 & \text{if } n' < N, \\ \sum_{t=n_k}^{n_{k+1}} (\hat{y}(t) - y(t))^2 & \text{if } n' = N. \end{cases} \tag{7.3}$$

In other words, the segment distortion is given as the contribution to reconstruction error resulting from the straight line interpolation between points $(n, y(n,l))$ and $(n', y(n',l'))$.

By calculating the distortion between two points $(n, y(n,l))$ and $(n, y(n',l'))$, $n' > n$, up to but not including $(n', y(n',l'))$, as long as we have not reached the last sample point, we ensure that the distortion is additive. This means that the distortion for the total reconstructed signal can be made up of a sum of the segment distortions, which is an important property of the distortion measure in the *min ave* algorithm.

The segment distortion measure established so far will serve as a quality measure for parts of the signal. Based on the segment distortions defined in Equations (7.2) and (7.3) we can now define distortion for the entire reconstructed signal. As pointed out earlier we apply two different solution algorithms: The *min max* algorithm and the *min ave* algorithm. We use the *min max* algorithm with the maximum error and the *min ave* algorithm with the sum of squared errors. This means that in the *min max* case, the segment distortion between any two points $(n, y(n,l))$ and $(n', y(n',l'))$, where $n' > n$ is given by Equation (7.2). The maximum distortion of the total reconstructed signal, $D^{\infty}$, is then the maximum of all segment distortions among the segments included in the final solution, that is,

$$D^{\infty} = \max_{k=1,\dots,M-1} d^{\infty}_{n_k l_k, n_{k+1} l_{k+1}} . \tag{7.4}$$

If the segment distortion between any pair of samples is defined according to Equation (7.3), which is the metric we use in combination with the *min ave* algorithm, it is clear that the total distortion of the reconstructed signal, $D$, is made up of a sum of the segment distortions of the segments included in the final solution, that is,

$$D = \sum_{k=1}^{M-1} d_{n_k l_k, n_{k+1}, l_{k+1}}, \quad n_k, n_{k+1} \in C. \tag{7.5}$$

### 7.1.2 Bit rate

Using straight lines in reconstruction of the signal, two parameters must be encoded for each retained sample of the signal; the amplitude and the position. We apply a simple predictive encoding scheme and encode the first order difference of both parameters (first order DPCM), that is, each segment of the signal is represented by the two parameters $\delta_{y(k)} = y(n_k, l_k) - y(n_{k-1}, l_{k-1})$ and $\delta_{n(k)} = n_k - n_{k-1}, k = 2, 3, \ldots, M$. In addition, we need to encode the absolute amplitude of the first point, $y(n_1, l_1)$.

We thus have a pair of $(\delta_{y(k)}, \delta_{n(k)})$ to be encoded for each segment of the signal. From the discussion in Section 4.1.2, we know that we have at least two alternatives when choosing between coding strategies. By encoding $(\delta_{y(k)}, \delta_{n(k)})$ by one single coder, we are able to utilize the dependency that exists between them and thus we get a more efficient compression of the signal. In this context we thus choose to encode the concatenated symbol $(\delta_{y(k)}, \delta_{n(k)})$ by one single coder.

The bit rate is based on a particular VLC table. This means that the output of the DPCM system is mapped into variable length codewords, where the length of the codewords is inversely related to the frequencies of the source symbols represented by the codeword. The approach on how we generate the VLC will be discussed in more detail in Section 7.2.5. As for now, let us just assume that the VLC is given.

Let us denote the number of bits needed to encode the segment between points $(n, l)$ and $(n', l')$ by $r_{nln'l'}$. The total bit rate, $R$, can then be expressed as

$$R = \sum_{k=1}^{M-1} r_{n_k l_k, n_{k+1} l_{k+1}}. \tag{7.6}$$

We are then faced with the following problem : Choose $M$, $n_1 < n_2 < \cdots < n_M$ and $l_1, \ldots, l_M \in \{-p, \ldots, p\}$ which minimize the distortion of the reconstructed signal, $D$ or $D^\infty$, under the constraint that the total bit rate, $R$, is less than the maximum allowable bit rate, $R_{max}$.That is,

$$\min_{\hat{S} \in Y} D^\infty, \quad \text{subject to} \quad R \leq R_{max}, \tag{7.7}$$

or

$$\min_{\hat{S} \in Y} D, \quad \text{subject to} \quad R \leq R_{max}. \tag{7.8}$$

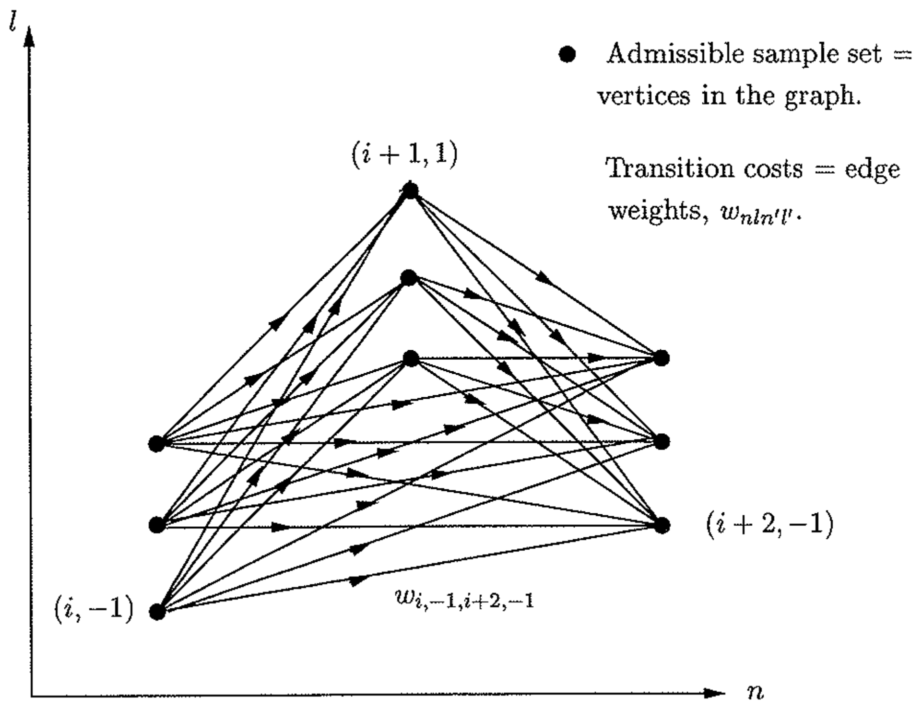
We thus solve the problem for both distortion measures under consideration, the one given by Equations (7.2) and (7.4) by Equation (7.7) and the one given by Equations (7.3) and (7.5) by Equation (7.8).

## 7.2 Solution method

Having defined the problem in mathematical terms, we now look for a suitable way to solve it efficiently and optimally. The solution method will be slightly different for the two classes of distortion measures that we consider. However, both solutions are based on a *shortest path* Dynamic Programming (DP) technique. By carefully defining the problem in terms of a graph, we will show that both problems correspond to searching for a shortest path through a predefined directed graph. The shortest path algorithm presented in this chapter is *not* identical to the cardinality constrained shortest path algorithm presented in Section 3.4 as we have now removed the cardinality restriction and replaced it by bit rate restriction. In this section we first show the mapping of our problem into a graph with the necessary notation. We then proceed to introduce the shortest path solution method before we discuss the differences in the solution methods for the *min max* case (Equations (7.2) and (7.4)) and the *min ave* case (Equations (7.3) and (7.5)).

### 7.2.1 Graph formulation

In order to be able to apply a shortest path solution scheme to our problem, we define the problem in terms of graph theory. We build a graph, $G$, from the admissible sample set as shown in Figure 7.2. The graph is directed and is defined as $G = (V, A)$ where the *vertex set* $V = \{(n, l), n = 1, \ldots, N, l = -p, \ldots, p\}$ and the *arc set* $A$ contains vertex pairs $((n, l), (n', l'))$, where $(n, l), (n', l') \in V$,



$l$
● Admissible sample set = vertices in the graph.
Transition costs = edge weights, $w_{nln'l'}$.
$(i+1,1)$
$(i+2,-1)$
$(i,-1)$
$w_{i,-1,i+2,-1}$
$n$

Figure 7.2: The mapping from the admissible sample set into a graph.

$n < n'$ and $l, l' \in \{-p, \ldots, p\}$ as described in Section 7.1. Note that there are no arcs in the graph between vertices with the same index $n$. This is due to the fact that we can not extract two samples for the same time index. If $(n_1, l_1), (n_M, l_M) \in V$, the set $(n_1, l_1), \ldots, (n_M, l_M)$ is said to be a *path* from $(n_1, l_1)$ to $(n_M, l_M)$ in $G$ if $(n_1, l_1), \ldots, (n_M, l_M) \in V$ are distinct vertices and $n_1 < n_2 < \cdots < n_M$. Let $P$ denote a path from vertex $(n_1, l_1)$ up to vertex $(n_M, l_M)$. The cost of each arc $((n,l),(n',l')) \in A$ is denoted $w_{nln'l'}$ and has different definitions in the *min max* and the *min ave* cases. It is made up of a combination of distortion and bit rate and will be explained in more details in Sections 7.2.3 and 7.2.4 for the *min max* and *min ave* methods, respectively. The length of $P$ is thus the sum of the costs of segments included in the path up to vertex $(n_M, l_M)$. Defining the problem this way, we are looking for the shortest path from vertex $(n_1, l_1)$ to vertex $(N, l_M)$.

### 7.2.2 Shortest path algorithm

We apply a shortest path algorithm to the graph defined in Section 7.2.1 to solve the problems stated in (7.7) and (7.8). The algorithm is a modified version of Dijkstra's shortest path algorithm [20], where the modification consists of taking into account the fact that we are working with a *directed* acyclic graph. This occurs as a natural consequence of the data we are working with.

We now introduce the shortest path algorithm that we use to search for the shortest path through our graph. To simplify notation, we assume for now that the admissible sample set, $Y$, equals the original sample set, $S$. This means that $l = 0$ for every point, so this index is dropped.

Denote by $C^*(i)$ the cost of the shortest path up to vertex $i$. Let $q(i)$ be the back pointer used to remember the optimal path. At each vertex we keep track of the length of the shortest path up to this vertex and the back pointer to the previous vertex on this path. We then consider vertex $i+1$. Clearly, $C^*(i+1)$ could equal $C^*(i) + w_{i,i+1}$, where $w_{i,i+1}$ equals the cost of the path from $i$ to $i+1$. But $C^*(i+1)$ could also equal $C^*(j)+w_{j,i+1}, j = i, i-1, \ldots, 0$. In other words we are looking for the combination of the cost of one of the optimal paths which we have already computed and a new cost introduced by including vertex $i+1$ which leads to the smallest overall cost. Let us set $C^*(0) = w_{-1,0}$, the cost of encoding the absolute value of the amplitude of the very first sample point. We then arrive at the following recursive equations:

$$C^*(0) = w_{-1,0}, \tag{7.9}$$
$$C^*(j) = \min\{C^*(i) + w_{ij},\ i = j-1, j-2, \ldots, 0\}. \tag{7.10}$$

Equations(7.9)-(7.10) constitute a *dynamic programming* solution to the shortest path problem. From the above formulation we present the resulting algorithm in Figure 7.3. The optimal path can be found by backtracking the pointers $q(i), i = N_Y - 1, \ldots, 0$. The formal proof of the correctness of this shortest path algorithm can be found in [20].

From the algorithm in Figure 7.3 we see that there are two nested loops, resulting in a time complexity of $\mathcal{O}(N_Y^2)$. However , it may not be necessary to go through all of these computations. By using a window which restricts how far apart two consecutive indices of $C$ are allowed to be, we can reduce the time complexity to $\mathcal{O}(N_Y \cdot w_{win})$ where $w_{win}$ is the window size. We will demonstrate the impact of the window size on the operational optimality of the solution in Section 7.3

The difference between the shortest path algorithm of Figure 7.3 and the CCSP algorithm given in Section 3.4 is that the CCSP algorithm incorporates the cardinality constraint into the CCSP algorithm and the problem is thus solved by running the CCSP algorithm once, finding solutions for increasing numbers of $M$, recording the optimal solution as we go along, until the upper

```
$C^*(0) = w_{-10}$;                                   // The cost of including the very first point
for $i = 1, \ldots, N_Y - 1$
    $C^*(i) = \infty$;                                // Start by labeling all vertices unprocessed
end
for $i = 0, \ldots, N_Y - 2$                          // Possible starting points for an edge
    for $j = i + 1, \ldots, N_Y - 1$                  // Possible ending points for an edge
        calculate segment distortion $d_{ij}$;
        calculate segment rate $r_{ij}$;
        calculate segment cost $w_{ij}$ as a function of $d_{ij}$ and $r_{ij}$;
        if($C^*(i) + w_{ij} < C^*(j)$);               // Is path shorter than the shortest so far?
            $C^*(j) = C^*(i) + w_{ij}$;               // Update shortest length
            $q(j) = i$;                               // Back pointer to previous vertex
                                                      // on shortest path
        end
    end
end
```

Figure 7.3: Shortest path algorithm.

bound on $M$ is reached. In the shortest path algorithm of Figure 7.3 the rate constraint is incorporated into the arc weight, $w_{ij}$, and the optimal solution is found by iteratively invoking the shortest path algorithm, in each run varying $w_{ij}$ in an intelligent way, until the optimal solution is found. The difference between the algorithm of Figure 7.3 and the CCSP algorithm of Figure 3.4 is reflected in the complexity of the algorithm. While the CCSP algorithm has a complexity of $\mathcal{O}(MN^2)$, the complexity of the shortest path algorithm of Figure 7.3 is of $\mathcal{O}(N_Y^2)$.

### 7.2.3 Distortion measure based on the maximum operator

In this section we introduce a solution method for the distortion measure based on the maximum absolute distance as stated in Equations (7.2) and (7.4). In order to solve this problem efficiently, we start out by solving the dual problem, that is [85, 88],

$$\min_{\hat{S} \in Y} R, \quad \text{subject to} \quad D^\infty \leq D^\infty_{max}, \tag{7.11}$$

where $D^\infty_{max}$ is the maximum distortion permitted. We then solve the problem stated in Equation (7.7) by iteratively solving the dual problem presented here.

To solve problem (7.11) the following definition of the cost function between any pair of vertices $(n,l)$ and $(n',l')$, where $n' > n$, is used

$$w_{nln'l'} = \begin{cases} \infty, & \text{if } d^{\infty}_{nln'l'} > D^{\infty}_{max}, \\ r_{nln'l'}, & \text{if } d^{\infty}_{nln'l'} \leq D^{\infty}_{max}. \end{cases} \tag{7.12}$$

We apply this definition of the cost function of the arcs to the shortest path algorithm of Figure 7.3. By assigning a length of infinity to any segment having a distortion larger than $D^{\infty}_{max}$ the shortest path algorithm will never select these segments to be included in a path. The length of every path through the graph equals the rate of the approximation. Therefore the shortest of all paths from the first to the last vertex in the graph corresponds to the minimum rate solution and hence is a solution to problem (7.11).

Having solved the problem in Equation (7.11), we now present a method to solve the problem of Equation (7.7). The optimal solution denoted by an asterisk, i.e., $R^*(D^{\infty}_{max})$, equals the minimum achievable bit rate for the maximum allowable distortion, $D^{\infty}_{max}$. A key issue here is the fact that $R^*(D^{\infty}_{max})$ is a non-increasing function of $D^{\infty}_{max}$. This means that $D^{1\infty}_{max} < D^{2\infty}_{max}$ implies $R^*(D^{1\infty}_{max}) \geq R^*(D^{2\infty}_{max})$. This can be proven as follows [85]:

**Proof (by contradiction):** Let compression set $\hat{S}_1$ and rate $R^*(D^{1\infty}_{max})$ be solutions to minimum rate optimization problem 1. Let compression set $\hat{S}_2$ and rate $R^*(D^{2\infty}_{max})$ be solutions to minimum rate optimization problem 2. Assume $D^{1\infty}_{max} < D^{2\infty}_{max}$ and $R^*(D^{1\infty}_{max}) < R^*(D^{2\infty}_{max})$. Then $\hat{S}_1$ is an admissible compression set for optimization problem 2, since $D^{1\infty}_{max} < D^{2\infty}_{max}$. Since by assumption $R^*(D^{1\infty}_{max}) < R^*(D^{2\infty}_{max})$, $\hat{S}_1$ is a better solution than $\hat{S}_2$, which is a contradiction since the selection algorithm employed to find $\hat{S}_2$ is optimal (see the algorithm of Figure 7.3). Hence $D^{1\infty}_{max} < D^{2\infty}_{max}$ implies $R^*(D^{1\infty}_{max}) \geq R^*(D^{2\infty}_{max})$.

The $R^*(D^{\infty}_{max})$ curve thus has a monotonously non-increasing piecewise linear form as illustrated in Figure 7.4. Utilizing this fact, we can use bisection [31] to find the optimal $D^{*\infty}_{max}$ such that $R^*(D^{*\infty}_{max}) = R_{max}$. An iterative approach is used, invoking the shortest path algorithm several times using bisection to find a $D^{*\infty}_{max}$ which results in $R^*(D^{*\infty}_{max}) = R_{max}$.

The problem at hand is a discrete optimization problem and the function $R^*(D^{\infty}_{max})$ is not a continuous function. For this reason there might not exist a $D^{*\infty}_{max}$ such that $R^*(D^{*\infty}_{max}) = R_{max}$, as shown in Figure 7.4. The proposed algorithm will still find an optimal solution of the form $R^*(D^{*\infty}_{max}) \leq R_{max}$ but only after an infinite number of iterations. Therefore, if an optimal solution is not found after a given number of iterations, the algorithm is terminated.

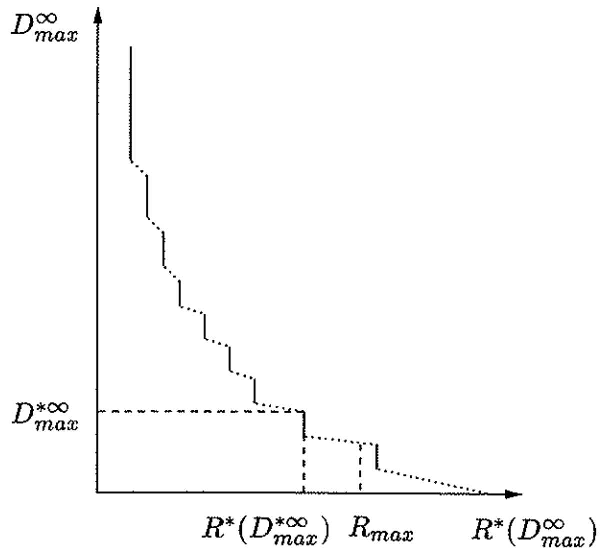


Figure 7.4: The characteristics of the $R^*(D^{\infty}_{max})$ function.

### 7.2.4 Distortion measure based on the summation operator

In this section we introduce a solution method for the distortion measure based on the sum of squared distances as stated in Equations (7.3) and (7.5). Again, we will use the shortest path algorithm in Figure 7.3 to solve the problem. A key issue in this algorithm is the definition of the cost function $w_{nln'l'}$.

In order to be able to solve the problem given in Equation (7.8) efficiently and optimally we use the Lagrangian multiplier approach [27]. This approach is extremely useful for solving constrained resource allocation problems. We will use it to relax the constraint, so that the relaxed problem can be solved using the shortest path algorithm. The basic idea behind the approach is to include the constraint into the objective function with a Lagrangian multiplier $\lambda$. This results in a Lagrangian cost function of the following form

$$J_{\lambda}^{C} = D^{C} + \lambda \cdot R^{C}, \tag{7.13}$$

where $\lambda$ is the Lagrange multiplier and the superscript $C$ denotes that the expression is a function of the compression set $C$ under consideration. The minimization of the expression given in Equation (7.13) is well suited to be performed with the shortest path algorithm.

It has been shown in [27, 89] that if there is a $\lambda^*$ such that

$$C^* = \arg \min_{C} J_{\lambda^*}^{C}, \tag{7.14}$$

and which leads to $R^{C^*} = R_{max}$, then $C^*$ is also an optimal solution to (7.8). As $\lambda$ sweeps from zero to infinity, the solution to (7.14) traces the convex hull of the operational rate distortion function, which is a non-increasing function. Therefore, by solving the unconstrained problem (7.14) repeatedly for different $\lambda$'s we can find the optimal solution to the constrained problem (7.8).

Having introduced the Lagrangian multiplier approach, the edge weight between any two graph vertices $(n, l)$ and $(n', l')$, $n' > n$, $l, l' \in \{-p, \ldots, p\}$, is given by

$$w_{nln'l'} = d_{nln'l'} + \lambda \cdot r_{nln'l'}. \tag{7.15}$$

Applying the shortest path algorithm with this definition of an edge weight, leads to the minimization of

$$\textstyle\sum_{k=1}^{M-1} w_{n_k l_k, n_{k+1} l_{k+1}}, \quad n_k \in C,\ l_k \in [-p, p], \tag{7.16}$$

and, hence, to an optimal solution to the unconstrained problem (7.14). By solving this unconstrained problem repeatedly for different $\lambda$'s we can find the optimal solution to the constrained problem (7.8).

The time complexity of the Lagrangian approach proposed here will be directly proportional to the number of iterations needed to achieve the target bit rate. The shortest path algorithm is invoked several times with different $\lambda$'s until $\lambda^*$ is found. By applying intelligent search criteria for $\lambda^*$, such as the Bezier curve fitting [87], execution time can be kept low.

### 7.2.5 VLC optimization

Our claim of optimality is clearly dependent on the chosen code structure, the width of the admissible sample point band, the size of our window restricting how far apart two consecutive points of $C$ may be, and, to a great extent, on the VLC tables. If we base the algorithm on a fixed VLC table generated off line, this will clearly make the performance of the coder signal dependent as it is hard to find *one* VLC table to match the characteristics of different ECG waveforms. In this case, operational optimality of the solution could only be claimed in the following sense:

$$C^* = \arg\min_{C}\{D^C + \lambda \cdot R^C | VCL\}. \tag{7.17}$$

We could choose to base our scheme on having a set of indexed VLC tables in both the encoder and the decoder and switching between these as the signal changes. This way we will avoid side information due to transmitting the VLC table between encoder and decoder.

In our case we iterate on the VLC table as a part of the compression scheme as shown in Figure 7.5. For each $\lambda$ we find the VLC table which matches the frequencies of the output symbols. We thus find the solution to the following problem

$$C^* = \arg \min_{C;f \in F} \{D^C + \lambda \cdot R^C\}, \tag{7.18}$$

where $f$ is a member of the family of context-conditioned parameter probability mass functions $F$. As a result the signal approximation and the parameter probability model are found jointly and the VLC locally optimized.

To start out the iterative process, depicted in Figure 7.5, the proposed ORD optimal coder processes the ECG signal with an initial fixed rate-distortion tradeoff, $\lambda$, and an initial probability mass function for $(\delta_{y(k)}, \delta_{n(k)}|VLC_{init})$. Having coded the input sequence at iteration $t$, based on the probability mass function $f^t()$, we use the frequency of the output symbols to compute $f^{t+1}()$ and then use $f^{t+1}()$ as basis for the VLC table in iteration $t+1$. The iterative process of Figure 7.5 stops when the cost improvement is less than $\varepsilon$. At this point an outer loop checks if the total bit rate $R$, is close enough to the target bit rate, $R_{max}$. If it is, the symbols are encoded by a variable length coder. If not, another guess for $\lambda$ is made, and the process is repeated. Since the sequence of selected points at iteration $t$ is available to the coder at iteration $t+1$, with the VLC's derived from that sequence, the coder can, at the very least, select the same path and thus, incur a lower or equal Lagrangian cost. The lower or equal cost is a consequence of the VLC tables optimized for that particular path. Therefore, the cost $C$ in Figure 7.5 is a non-increasing function of iteration $t$ and, hence, the iterative process converges to a local minimum.

## 7.3 Numerical experiments and discussion

For evaluation of the performance of the coders, we use the PRD distortion measure, given in Equation (2.3) and the maximum error given in Equation (2.1). We evaluate these as a function of bit rate. We supplement these error measures by visual inspection of the reconstructed signal in addition to reporting some execution times.

The recordings used in the coding experiments in this section are described in Appendix A.

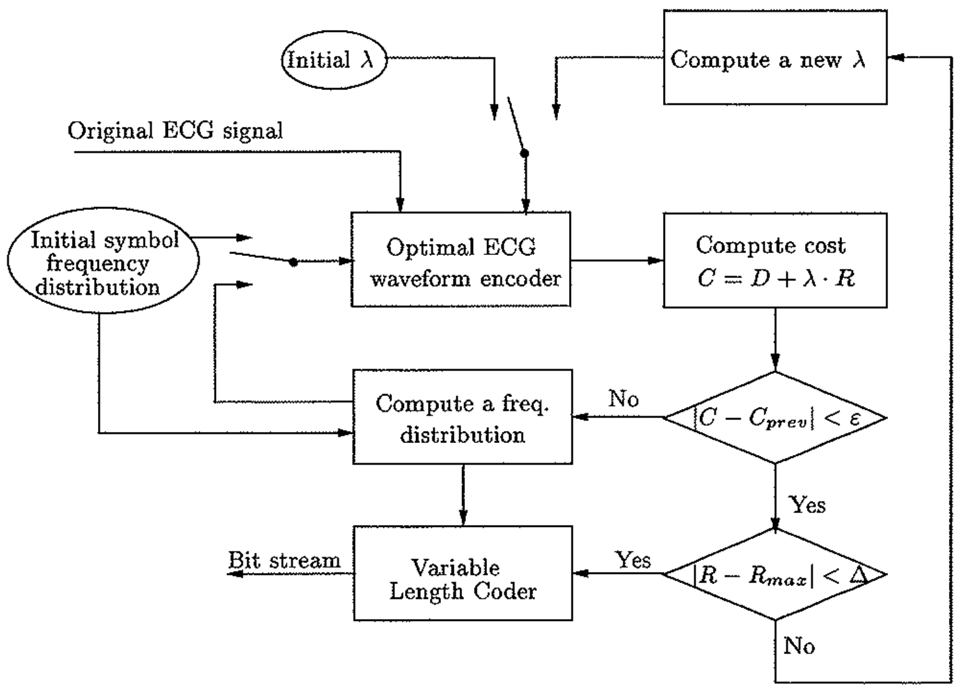

Initial $\lambda$
Compute a new $\lambda$
Original ECG signal
Initial symbol frequency distribution
Optimal ECG waveform encoder
Compute cost $C = D + \lambda \cdot R$
Compute a freq. distribution
No
$|C - C_{prev}| < \varepsilon$
Yes
Bit stream
Variable Length Coder
Yes
$|R - R_{max}| < \Delta$
No

Figure 7.5: The structure of the encoder.

We process the input signal in blocks of 500 time indices at a time and use an admissible control point set with $p = 3$, corresponding to introducing three points on each side of the original samples. To keep execution time down, we introduce an analysis window restricting how far apart two successive samples of $C$ are allowed to be. This will lead to a minimal sacrifice in optimality as is shown in Figure 7.6. From the figure we see that the window size has most severe impact on low bit rates. This is natural as low bit rates correspond to extraction of few points. The average distance between two consecutive points increases as the bit rate decreases. In our experiments we have set the window size to 50.

We compare the results in terms of PRD versus bit rate, from the compression schemes presented in this chapter to the results from the CCSP algorithm based on polynomial interpolating approximation presented in Section 4.3, as well as the 32-tap ECG optimized filter bank described in Section 2.3.1. The reason for choosing the CCSP method based on polynomial interpolating approach for comparison in terms of PRD is that this method shows good results, especially for low bit rates, cf. the discussion in Section 5.2.3. When it comes to the maximum error we compare the results from the approaches

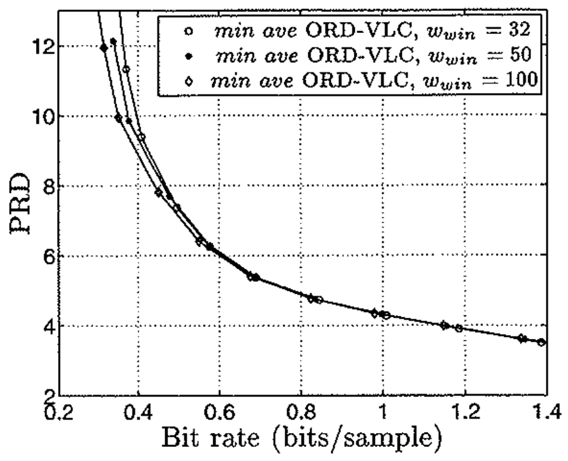


Figure 7.6: Evaluation of different window sizes.

developed in this chapter to the FAN algorithm presented in Section 2.2.1 as well as the ECG optimized filter bank. As we have seen in the experimental sections in Chapters 4 and 5, the FAN method gives good results in terms maximum error, and it is therefore interesting to compare our results to this compression scheme in this case.

We refer to the method introduced in this chapter as joint operational rate distortion and VLC optimal approach (ORD-VLC optimal approach) and distinguish between the two different distortion measures implemented by using the notation *min max* ORD-VLC optimal approach for the method introduced in Section 7.2.3 and *min ave* ORD-VLC optimal approach for the approach introduced in Section 7.2.4.

### 7.3.1 Evaluation based on the PRD measure

Figure 7.7 presents operational rate-distortion plots for the various coders for bit rates between 0.2 and 1.4 bits per sample (bps).

For the test signals mit100_1000 and mit207_1800, the *min ave* ORD-VLC method outperforms all other methods by a significant margin, especially for low bit rates (below 0.8 bps). For the other test signals, the *min ave* ORD-VLC has the best performance among the coders except for a small region for the mit202_0800 test signal (above 1 bps) and some areas of PRD's above 10 for some of the other signals, where the filter bank method has a slightly better performance.

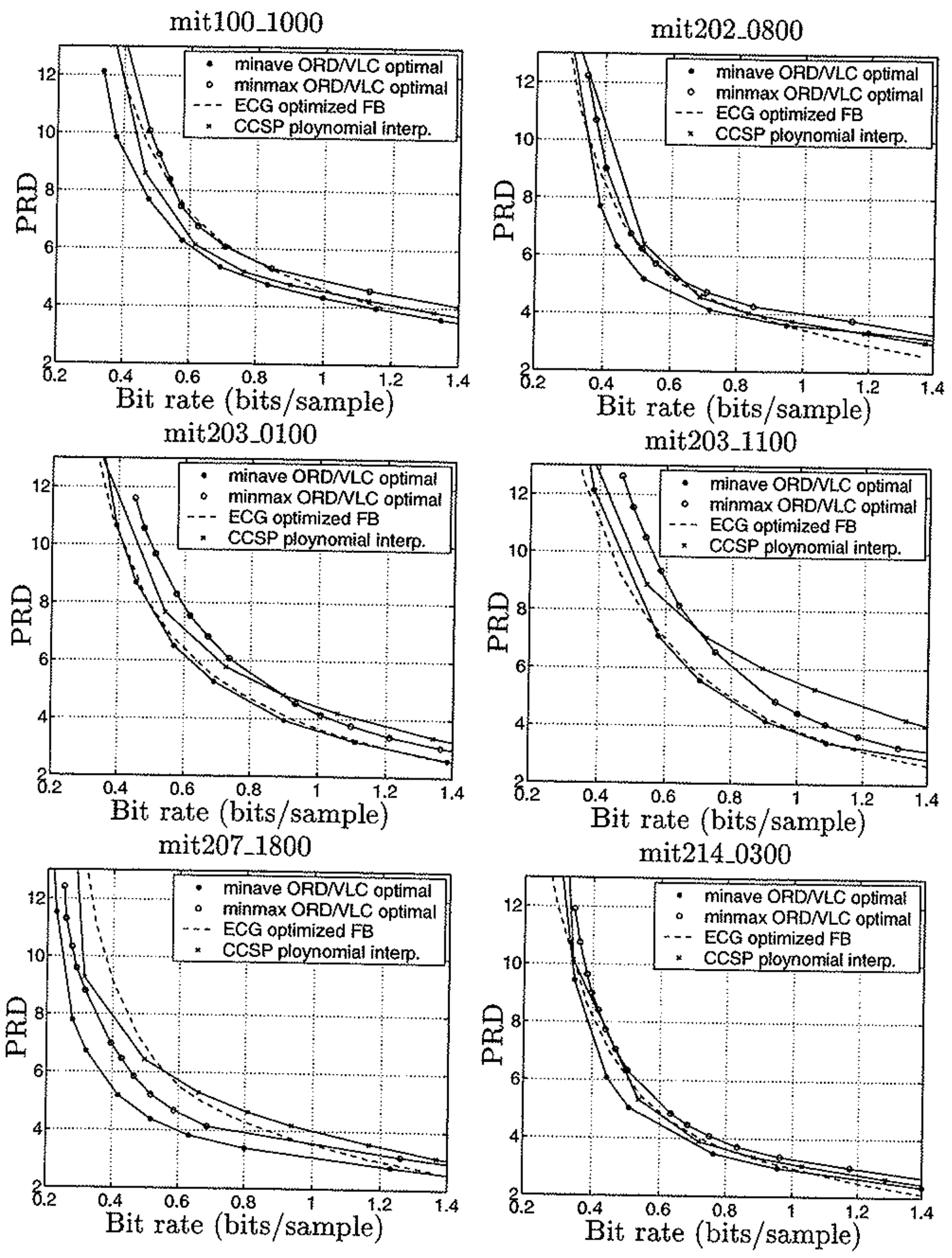


Figure 7.7: Coding performance for the different coders. Solid line with stars: *min ave* ORD-VLC optimal approach with $p = 3$. Solid line with circles: *min max* ORD-VLC optimal approach with $p = 3$. Dotted line: ECG optimized filter bank. Solid line with crosses: CCSP, polynomial interpolation.

The filter bank method performs similarly to the *min max* ORD-VLC method for test signal mit100_1000, mit202_0800 and mit214_0300. For the two test signals mit203_0100 and mit203_1100 the filter bank outperforms the *min max* ORD-VLC method, while for test signal mit207_1800 it is the other way around.

The difference in performance of the coders for the various test signals is related to the nature of the signals. Test signal mit207_1800 has approximately half the dynamic range of mit203_0100. Higher dynamic range of the original signal leads to a possibly higher number of different symbols representing the signal and this will affect the ORD-VLC methods and the CCSP method since they are based on extraction of points in the plane. The filter bank method will not be affected in the same way as its performance is related to extraction of frequencies rather than points in the plane.

### 7.3.2 Evaluation based on the maximum error

Figure 7.8 shows obtained maximum errors versus bit rates for the different coders and test signals. We see that the *min max* ORD-VLC optimal approach has superior performance in terms of maximum error for all test signals and all bit rates. This indicates that even though the FAN algorithm shows better results than the CCSP algorithms developed in Chapters 4 and 5 and the ECG-optimized filter bank presented in Section 2.3.1 in terms of maximum error, there is still much more to be gained.

The *min ave* ORD-VLC approach is based upon minimization of the same distortion measure as the ECG-optimized filter bank, namely the sum of squared errors. From Figure 7.8 we see that the *min ave* ORD-VLC approach performs better than the filter bank for test signals mit202_0800, mit207_1800, and for most bit rates for test signals mit100_1000. For the other test signals the filter bank has similar or better performance to the *min ave* ORD-VLC approach.

The reason for the apparently casual variation in performance between the *min ave* ORD-VLC approach and the filter bank, is probably that they both suffer from the same artifact: They do not attempt to minimize the maximum error, and hence, if there is a large deviation between the original and reconstructed signal somewhere in a long signal sequence, this will have crucial impact on the maximum error.

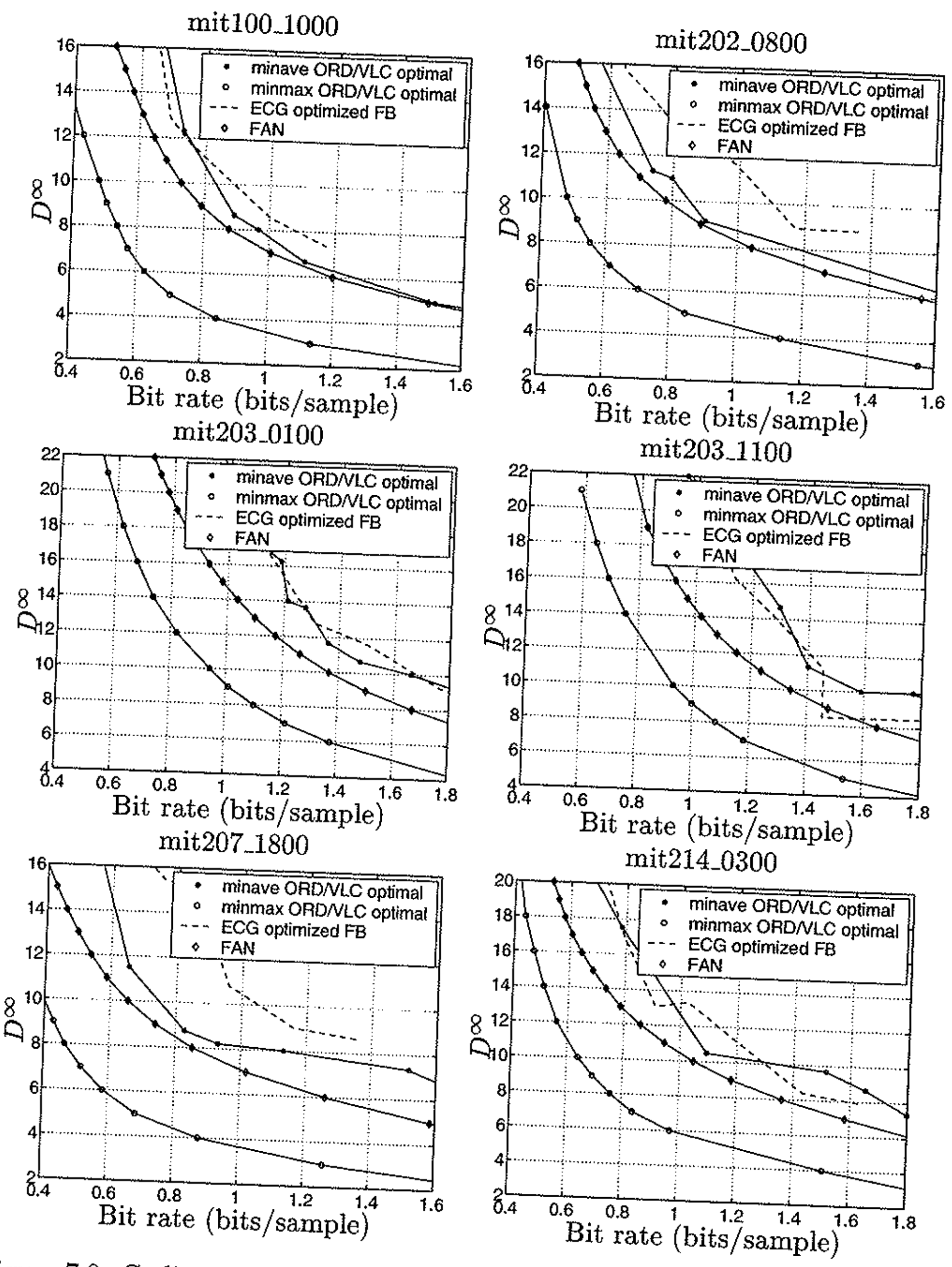


Figure 7.8: Coding performance for the different coders. Solid line with stars: *min ave* ORD-VLC optimal approach with $p = 3$. Solid line with circles: *min max* ORD-VLC optimal approach with $p = 3$. Dotted line: ECG optimized filter bank. Solid line with diamonds: FAN.

### 7.3.3 Evaluation based on visual inspection

The reconstructed signal is shown at a bit rate of 1.0 bits per sample in Figure 7.9 and at 0.5 bits per sample in Figure 7.10. The original signal is also included. From the plots we can conclude that the FAN algorithm does not include as much details as the other methods in the reconstructed signal. This is especially prominent at low bit rates.

The ECG-optimized filter bank includes more of the ripple noise seen in the original signal in the reconstructed signal than the other methods. This is undesirable noise, and will often be removed by a filter before transmission or storage of the ECG signal. The ORD-VLC optimal approaches, the CCSP method based on polynomial interpolation and the FAN algorithm smoothes out this ripple noise.

Working with biomedical signals, we could argue that some parts of the signal are more important than others. That is, some parts of the signal may contain more critical diagnostic information than others. Ideally we would like to compress the signal losslessly, but in many applications this is not an option. Given a fixed bit budget, it would therefore make sense to spend more bits on some parts of the signal and thus reconstruct some parts of the signal with higher fidelity while allowing other parts to have a larger distortion. The approaches proposed in this chapter are very flexible in this respect. We could easily adjust the bit budget spent on different parts of the signal simply by adjusting the width of the admissible sample point band [51].

### 7.3.4 Evaluation based on execution time

To get an impression of the execution time of the ORD-VLC optimal approaches presented in this chapter, we report some benchmarks in the following, i.e., some execution times for the algorithm for different compression ratios run on a specific machine. We only report execution times for the *min ave* ORD-VLC method, the *min max* ORD-VLC approach will have similar performance in terms of execution time.

The complexity of the *min ave* ORD-VLC algorithm is of $\mathcal{O}(N_Y w_{win})$ where $N_Y$ is the number of points in the admissible point set $Y$, and $w_{win}$ is the window size.

Table 7.1 on page 134 shows execution times for the *min ave* ORD-VLC approach for different block sizes and different number of points in the admissible sample point band. The experiments are run on an HP9000 C360 work station

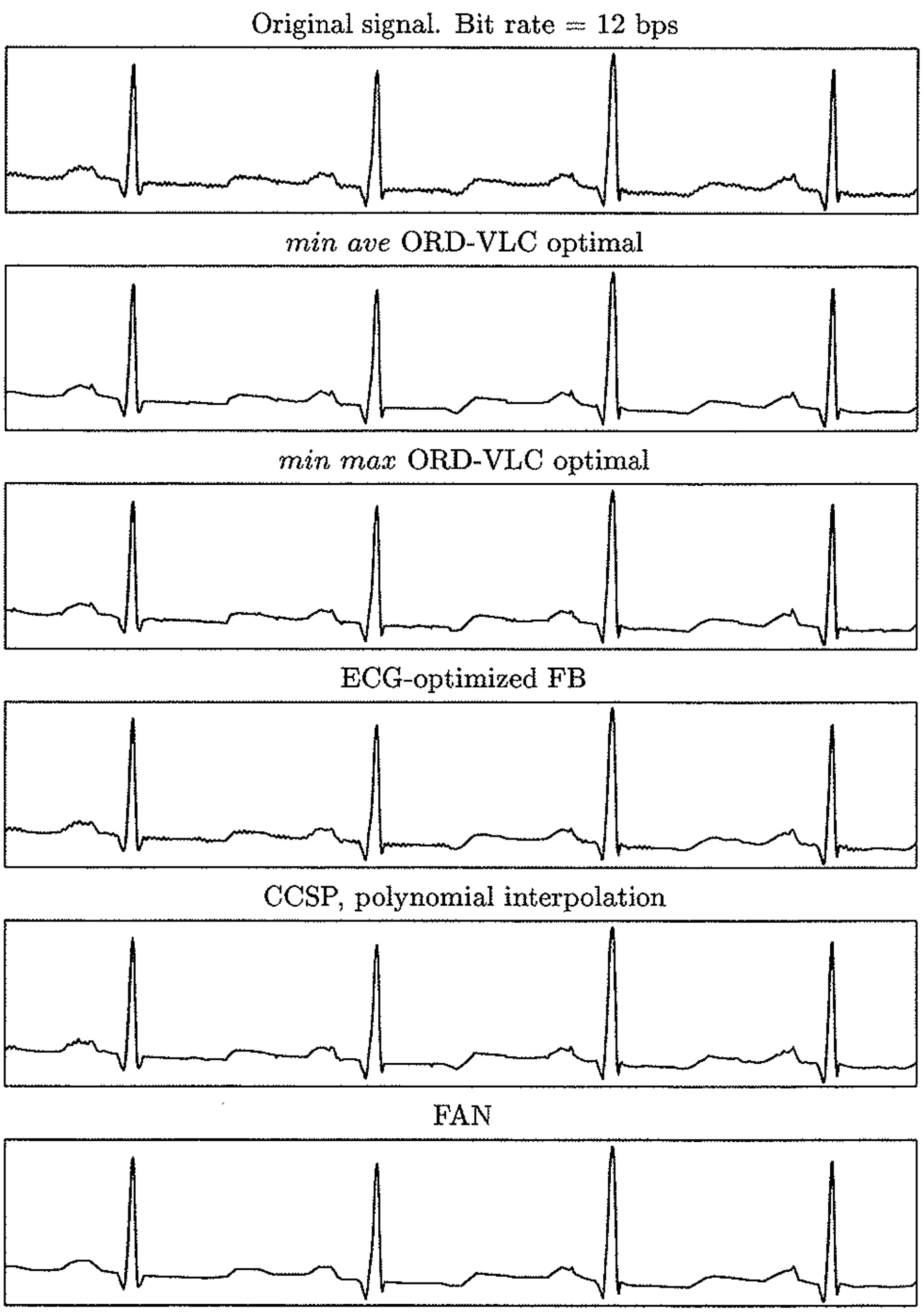


Figure 7.9: Short segment of reconstructed signal (taken from mit100_1000) at 1.0 bits per sample.

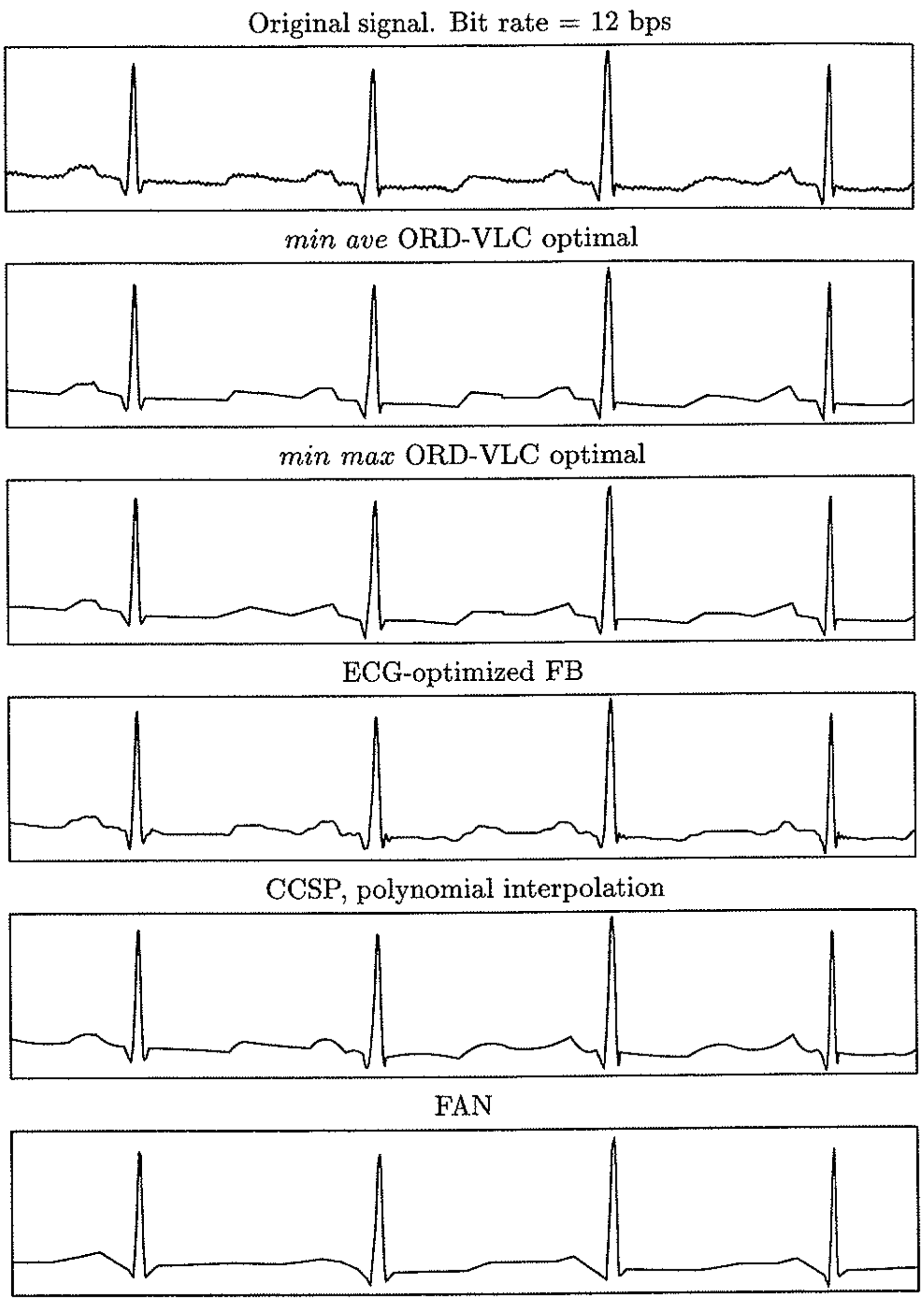


Figure 7.10: Short segment of reconstructed signal (taken from mit100_1000) at 0.5 bits per sample.

with a 367 MHz processor, and the test signal is extracted from the beginning of mit100_1000. The execution times are based on the average of 10 runs.

From Table 7.1 it can be seen that for the *min ave* ORD-VLC approach with a sampling frequency of 360 Hz, real time requirements are fulfilled for *one* run of the algorithm, for all block sizes as long as the size of the admissible sample point set is kept low, i.e., $p = 0, 1$. For larger values of $p$, real time requirements are violated. Keep in mind that in the version of the *min ave* algorithm implemented in this chapter, we iterate on the VLC tables in order to find the underlying parameter probability distribution best matching the source symbols. If we apply a fixed VLC table generated off line, the execution time of the algorithm will be shorter than the ones reported in Table 7.1.

In order to obtain the optimal solution we have to invoke the *min ave* ORD-VLC algorithm repeatedly for different values of $\lambda$ to find the optimal solution. The algorithm has to be invoked at least three times, maybe more. Wee see that even though the algorithm is invoked three times, the real time requirements are still fulfilled for $p = 0$, i.e., when the admissible sample point set equals the original signal. The execution time may be lowered by restricting the size of the window $w_{win}$ further, by limiting the width of the admissible sample point band, $p$, placed around the original signal and by applying an intelligent search criterion for the $\lambda$ resulting in the optimal solution [85].

### 7.3.5 Output symbol distributions

Treating tuplets $(\delta_{y(k)}, \delta_{n(k)})$ as one concatenated symbol was motivated by the hypothesis that the optimal VLC's for the two components are dependent, i.e., that knowledge of a particular $\delta_{y(k)}$ affects the distribution of $\delta_{n(k)}$ at convergence, and vice-versa. To test this hypothesis we observe, in Figures 7.11 and 7.12, probability distributions at convergence of signal mit100_1000 encoded at 0.38 and 1.15 bps, respectively. Clearly, the two locally optimal distributions are far from being uniform and exhibit significant dependency between the horizontal and the vertical displacement components. In particular, it is evident from the figures that symbols having both $\delta_{y(k)}$ and $\delta_{n(k)}$ components large are assigned probabilities close to zero or very long code words, while symbols with only one of the components large are assigned relatively high probabilities, or shorter code words.

## 7.4 Summary

In this chapter we have developed an operationally optimal rate distortion algorithm for ECG signal compression. By the very nature of our approach,

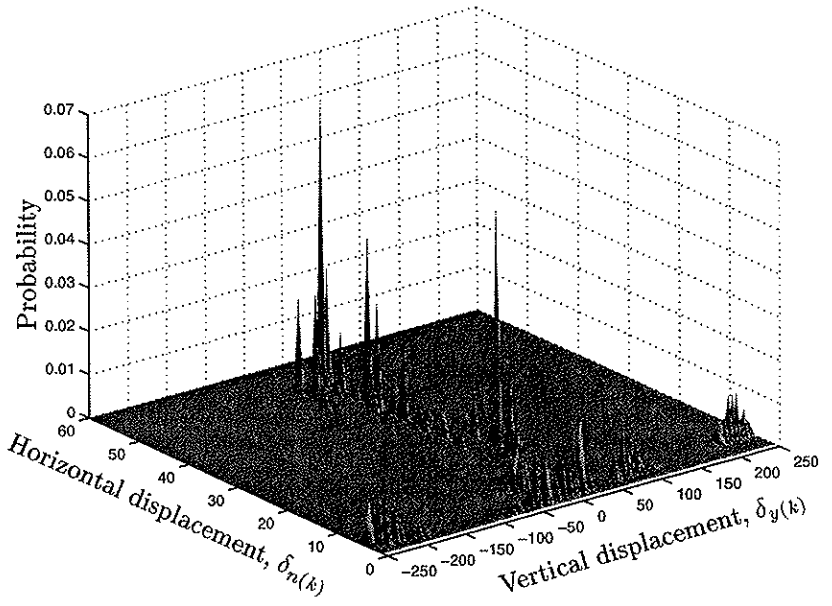


Figure 7.11: Symbol probability distributions at convergence of signal mit100_1000 encoded at 0.38 bps.

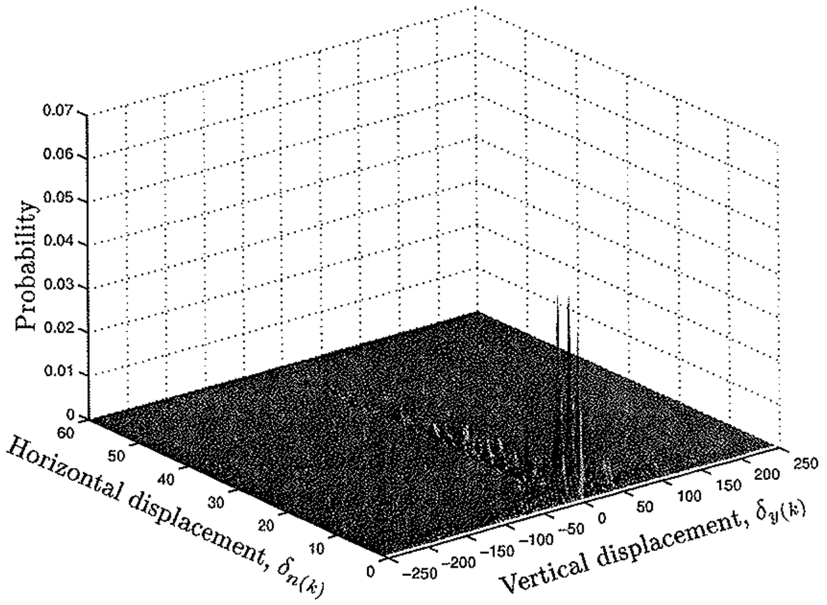


Figure 7.12: Symbol probability distributions at convergence of signal mit100_1000 encoded at 1.15 bps.

| **min ave ORD-VLC optimal approach** | | |
|---|---|---|
| Block size | Adm. band, $p$ | Execution time in sec. |
| 100 | 0 | 0.08 |
| | 1 | 0.22 |
| | 2 | 0.32 |
| | 3 | 0.42 |
| 200 | 0 | 0.15 |
| | 1 | 0.45 |
| | 2 | 0.70 |
| | 3 | 0.97 |
| 300 | 0 | 0.22 |
| | 1 | 0.68 |
| | 2 | 1.09 |
| | 3 | 1.52 |
| 400 | 0 | 0.29 |
| | 1 | 0.91 |
| | 2 | 1.46 |
| | 3 | 2.07 |
| 500 | 0 | 0.36 |
| | 1 | 0.15 |
| | 2 | 1.84 |
| | 3 | 2.62 |

Table 7.1: Execution times for the *min ave* ORD-VLC optimal approach CCSP run on an HP9000 C360.

no other technique based on linear interpolation will result in a smaller reconstruction error for the same bit rate, given the structure of the coder. The approach is not dependent on a particular VLC, it computes the probability distribution of the parameters resulting in the locally most efficient ORD curve as part of the compression scheme.

We apply two philosophies for the distortion measure in our approach, the sum of squared errors and the maximum absolute distance. Based on these two distortion measures we present two solution schemes, both based on a shortest path algorithm.

We compare the performance of the coders to both the CCSP ECG compression method based on polynomial interpolating approach presented in Section 4.3, the FAN method presented in Section 2.2.1, as well as a state of the art filter bank coder presented in Section 2.3.1. Coding experiments show that the operational rate distortion (ORD) coding techniques based on minimization of the sum of squared errors has superior performance compared to both the CCSP algorithm based on polynomial interpolation and the filter bank approach in terms of PRD. The *min max* ORD-VLC method has superior performance compared to the traditional FAN algorithm and the filter bank method in terms of maximum error. These results are verified by visual inspection of the reconstructed signal.

# Chapter 8

# Summary and conclusions

In this dissertation we have investigated the use of optimization theory in signal representation and compression. The focus has been on designing new approaches to signal representation and compression based on shortest path methods.

As opposed to traditional transform-based compression methods, we propose a new way of looking at the compression problem. Formulating the problem in terms of graph theory, shortest path methods are applied to solve the problem.

By the very nature of our approaches, the problem is solved in an optimal manner with respect to the given constraints. However, as the methods may be somewhat computationally expensive we focus on solving the problem in a computationally effective way. This involves preprocessing the graph and computing all arc weights in an intelligent way.

## 8.1 Contributions of this dissertation

1. A general theoretical framework for signal compression from an optimization point of view - the *Cardinality Constrained Shortest Path method* (CCSP) - is developed in Chapter 3. The methodology is based upon modelling the problem in terms of graph theory, thereby making it suitable for solving with a shortest path algorithm. We also show that, assuming the arc weights in the graph can be computed in a computationally effective way, the compression problem can be solved with a complexity no higher than cubic in the number of samples.

2. In Section 4.1 a new compression scheme based on the encoding of linear line pieces which are used to approximate the signal is developed. The segments are fit in an optimal way under the given constraints. The general theoretical framework from Chapter 3 is applied in solving the problem. We focus on effective computation of the arc weights in order to keep the computational complexity as low as possible.

3. In ECG compression both the sum of squared errors and the maximum error are frequently used as distortion measures. In Section 4.2 an approach based on incorporation of both these error measures into one compression scheme is developed. Again, it is crucial that this is done in a computationally effective way. We thus propose an effective way of limiting the maximum error by the use of convex hull theory. The problem is formulated in a way that is suitable for usage in combination with the theoretical framework from Chapter 3.

4. In Section 4.3 and Appendix B the compression algorithm is further developed, extending it to an approach where polynomials of second order can be applied in approximation and compression of the signal. We still focus our attention on effective computation of all the arc weights, and it is shown in Appendix B how this is done in the polynomial interpolating case.

5. Releasing the interpolation restriction, we propose new versions of the compression scheme in Sections 5.1 and 5.2. We do no longer insist on exact equality between the original and the reconstructed signal at any specific points, i.e., a non-interpolating approach is developed. We develop two new coders, one based on linear approximation and one based on second order approximations in Sections 5.1 and 5.2, respectively. The algorithms developed are extensions of the CCSP algorithm developed in Chapter 3. This is obtained without increasing the computational complexity of the algorithm. Unfortunately the results are not as good as we hoped for, but the approach gives valuable insight into the problem.

6. In Chapter 6 an image contour compression scheme is developed. The approach is based on the fitting of linear line segments to the original contour in an optimal way under the given constraints. This is a modified version of the algorithm presented on Section 4.1. Its performance is superior when compared to similar compression schemes like the one in [86].

7. Modifying the constraint in our optimization problem applied in the previous chapters, we develop a theoretical framework for signal compression in a rate-distortion optimal sense in Chapter 7. Instead of limiting the number of points in our solution, we rather develop an approach limiting the number of bits needed to encode such points. We develop two approaches, one minimizing the maximum error and one minimizing the sum of squared errors, in Section 7.2.3 and 7.2.4, respectively.

With each of the methodologies presented in Sections 4.1, 4.2, 4.3, 5.1, 5.2, 7.2.3, 7.2.4 and Chapter 6 belongs implementation of coders based on the respective approaches.

Most of these contributions, as well as related material, have also been published in [64, 66, 65, 67, 69, 71, 70, 68, 4].

## 8.2 Conclusions from this dissertation

Several conclusions may be drawn from the experiments reported in this dissertation. Below is a list of the main conclusions:

- Looking at the compression problem from a graph-theoretic point of view is untraditional, but very fruitful. In Chapter 7 we see in the coding experiments that the rate distortion optimal approaches developed in this chapter generally perform much better than both traditional and more recently developed time domain ECG compression methods as well as a state of the art filter bank coder.

- We have investigated the use of polynomials in compression of ECG signals. From this we draw the conclusion that applying a polynomial of order one, i.e., linear line segments, in approximating the signal gives promising results. However, the straight lines are prominent in the reconstructed signal. Increasing the order of the polynomial from one to two, increases the performance of the compression scheme both in terms of PRD and visual appearance, especially at low bit rates. Increasing the order of the polynomials from two to three, results in marginal gain in performance in terms of PRD. However, the increased execution time introduced as a consequence of increasing the order of the polynomial from two to three, indicates that this may not be worthwhile. Increasing the order of the polynomial beyond three results in a decreasing performance of the total compression scheme.

| Algorithm | Error measure | Arc description | Interpol./ Non-interpol. | Complexity |
|---|---|---|---|---|
| CCSP | $D$ | Linear | Interpol. | $\mathcal{O}(MN^2)$ |
| CCSP | $D, D^\infty$ | Linear | Interpol. | $\mathcal{O}(N^2 logN + MK)$ |
| CCSP | $D$ | Sec. order pol. | Interpol. | $\mathcal{O}(MN^2)$ |
| CCSP | $D$ | Linear | Non-interpol. | $\mathcal{O}(MN^2)$ |
| CCSP | $D$ | Sec. order pol. | Non-interpol. | $\mathcal{O}(MN^2)$ |
| *min ave* ORD-VLC | $D$ | Linear | Interpol. | $num_{iter}\mathcal{O}(w_{win}N_Y)$ |
| *min max* ORD-VLC | $D^\infty$ | Linear | Interpol. | $num_{iter}\mathcal{O}(w_{win}N_Y)$ |

Table 8.1: Overview of the complexity for the different algorithms developed.

- We have demonstrated that it is possible to combine a constrained shortest path algorithm of complexity $\mathcal{O}(MN^2)$ with arc models being polynomials of order one or two, both interpolating and non-interpolating approaches, without increasing the computational complexity of the algorithm. An overview of the complexity of the different approaches developed in this dissertation is shown in Table 8.1.

## 8.3 Directions for future research

### Approximating signals by the use of polynomials of varying orders

In this dissertation we have compressed signals by keeping the order of the polynomial constant for one given run. It is doubtful that this is optimal. Some pieces of the signal may be represented most efficiently by a second order polynomial, while others, where there is less activity, may be better approximated using linear line segments. Coding with a varying order of the polynomial involves one extra parameter per arc of the signal, namely the order of the polynomial. Most likely, this parameter will have a small dynamic range and will thus hopefully not give a big contribution to the total bit rate. It would be fruitful to examine an approach based on this idea further.

### Minimization of maximum error

A main challenge for the different versions of the CCSP algorithm developed in Chapters 4 and 5 is the of limitation of maximum error. It would be fruitful to develop an approach based on the CCSP algorithm minimizing the maximum error.

### Extraction of a variable number of samples from each signal block

As we have seen in the experimental sections in Chapters 4 and 5, the CCSP algorithm faces a challenge in coping with the maximum error. This may be partly due to the fact that it is only allowed to extract the same predefined number of samples from each block of the signal. If the signal contains high frequencies, the peaks resulting in the overall lowest cost may be left out. One such peak in a long signal sequence, results in a high maximum error for the total reconstructed signal.

One way to account for this effect could be by allowing the CCSP algorithm to extract a variable number of samples from each block of the signal. This could be implemented in the version of the CCSP algorithm with the maximum error bound incorporated. We could picture a situation where we impose a maximum error bound on the solution as was done in Section 4.2. If this results in no valid solution for a given number of extracted signal samples, due to the fact that there does not exist a path through the reduced graph, the number of allowable extracted signal samples may be increased for this particular block. For other blocks (containing lower frequencies), we allow fewer samples to be extracted. This may cause the execution time of the algorithm to increase, but as we saw in Section 4.2, reduced execution time is a nice side effect from incorporating the maximum error into the CCSP algorithm.

### Improvement of encoding techniques

The compression schemes developed in this dissertation are mainly based upon approximating the signal by extraction of some significant parameters. The encoding of these parameters is thus a vital part of the compression scheme. We have put a considerable amount of effort into encoding the parameters as effectively as possible. However, preliminary investigations indicate that there is more to be gained in this part. In [43] some experiments were done with second order DPCM in encoding of the parameters. The reported results were good, and more effort should be put into investigating this and related approaches for the algorithms developed in this dissertation.

### CCSP algorithm with maximum error incorporated and second order polynomial interpolation

From the experimental sections in Chapter 4, we see that the CCSP method with the maximum error incorporated yields good results, especially in terms of execution time as real time performance is achieved. On the other hand the CCSP method based on polynomial interpolation gives good results in terms of PRD and visual performance of the reconstructed signal. It would therefore be interesting to combine these two approaches and develop a CCSP compression scheme based on polynomial interpolation, incorporating an upper bound on the maximum error.

### Implementation of a uniform distortion measure in compression of image contours

In Chapter 6, containing representation and compression of image contours, it is hard to do inter-method comparisons as there is no uniform distortion measure developed for this purpose. The development of such a distortion measure would be very useful in evaluating the different coders. The metric used to evaluate distortion within MPEG-4 is

$$\mathrm{D_{MPPEG4}} = \frac{\text{Number of pixles in error}}{\text{Number of interior pixels}},$$

where a pixel is said to be in error if it belongs to the interior of the original object and the exterior of the approximating object or vice-versa. By incorporating this error measure into the contour compression scheme of Chapter 6 it would be easier to evaluate the performance of the algorithm toward other algorithms applying the MPEG-4 distortion metric.

### Second order polynomial approximation in compression of image contours

As was seen for the CCSP approach, we achieved a better approximation in terms of PRD and visual performance by increasing the order of the polynomial from one to two. There is reason to believe that this also holds for image contours. Extension of the approach developed in Chapter 6 into an approach where second order polynomials are applied, could therefore be an interesting means of increasing the performance of the algorithm.

### Effective implementation of the coders

The compression methods we focus on in this dissertation are optimal approaches, under the given constraints. Instead of employing a heuristic, the proposed methods search through possible solutions in an intelligent way in order to find the optimal solution. Although real time requirements are satisfied in many cases, this makes the approaches more time-consuming than conventional heuristics. In order to cope with this and to make the algorithms even faster, the technique presented in [38] should be incorporated into the algorithms.

### Application to image compression

The graph-theoretic approach presented in this dissertation may also be useful in other areas, such as in compression of images. Traditionally, image compression is performed by subdividing the image into blocks, each of which is processed by the means of a transform in order to obtain more efficient representation of data.

An alternative viewpoint is to consider the data as samples of a one-dimensional waveform. This can be implemented by scanning the image in a smart way, to transform it from a 2-D representation to a 1-D representation suitable for input to our graph-theoretic methods. The approaches developed in chapters 4, 5 and 7 can then all be applied to process and compress the image.

# Appendix A

# ECG compression test signals

This appendix contains short segments of the ECG signals used in the coding experiments throughout the dissertation in addition to the power density spectra plots for these signals. This is to illustrate how the nature of the ECG signal varies with different heart arrythmias.

All the test signals are taken from the MIT-BIH Arrythmia Database [60]. "mitxxx_yyyy" denotes record number xxx starting at time yy:yy. Each total record time is ten minutes, corresponding to 216 000 samples. The sampling frequency is 360 Hz with a resolution of 12 bits per sample. The signal mit100_1000 is normal sinus rhythm while the others are various abnormal rhythms.

Figure A.1 shows the first 10 seconds of the test signals applied in this dissertation.

Table A.1 shows the annotations used in the test signals, i.e., comments made on the signal to describe the heart rhythms.

Figure A.2 shows the power spectral density of the test signals computed using Welch's averaged, modified periodogram method.

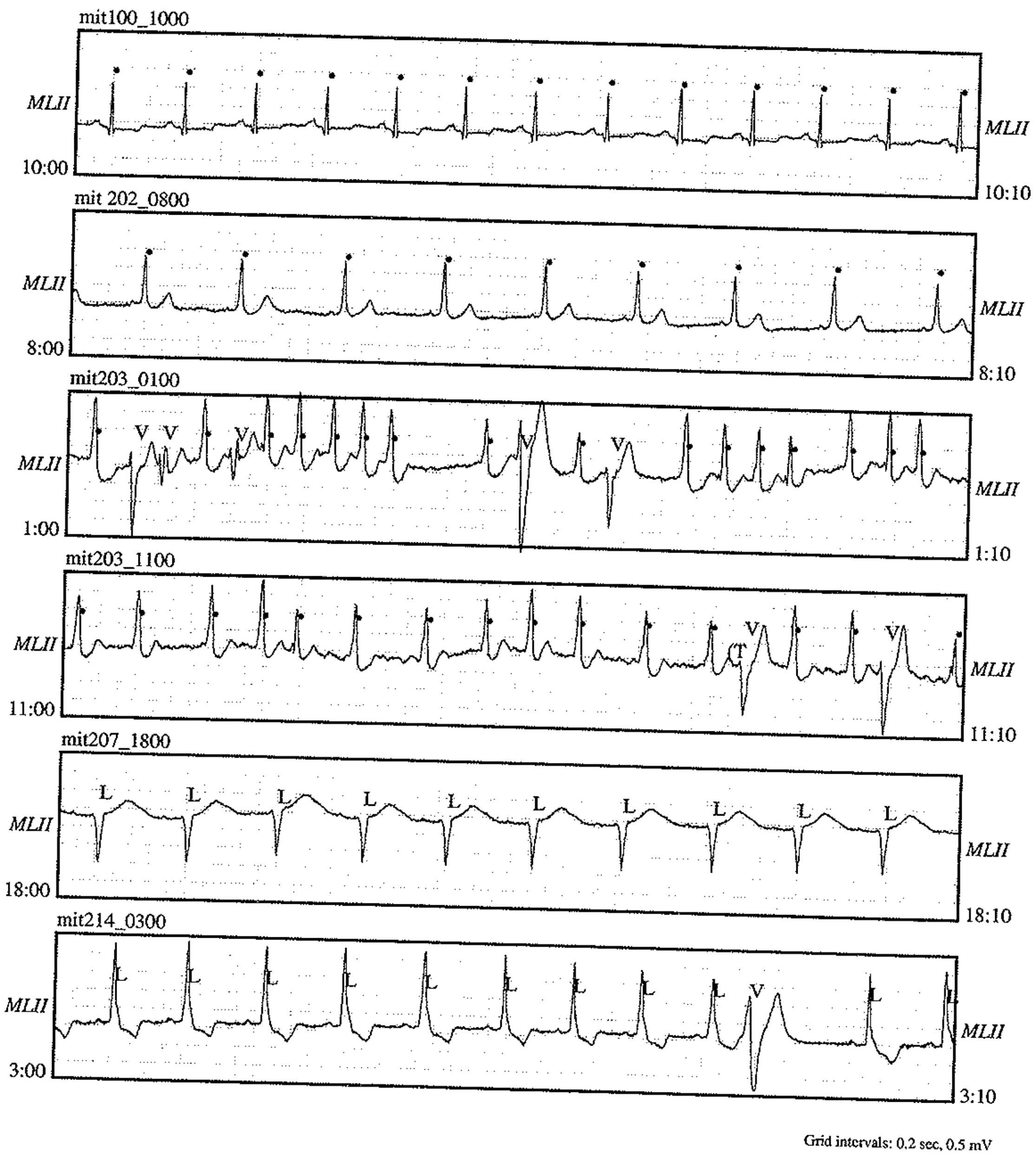


Figure A.1: Original ECG's (first 10 seconds) used in the experiments. "mitxxx_yyyy" denotes record number xxx starting at time yy:yy. Each record is ten minutes long, corresponding to 216 000 samples.

| *Symbol* | *Meaning* |
|---|---|
| Beat annotations: | |
| · | Normal beat |
| L | Left bundle branch block beat |
| V | Premature ventricular contraction |
| Rhythm annotations (appear below the level used for beat annotations): | |
| (T | Ventricular trigeminy[1] |

Table A.1: Annotation used on the ECG test signals.

[1]The occurrence of three pulse beats in rapid succession [83].

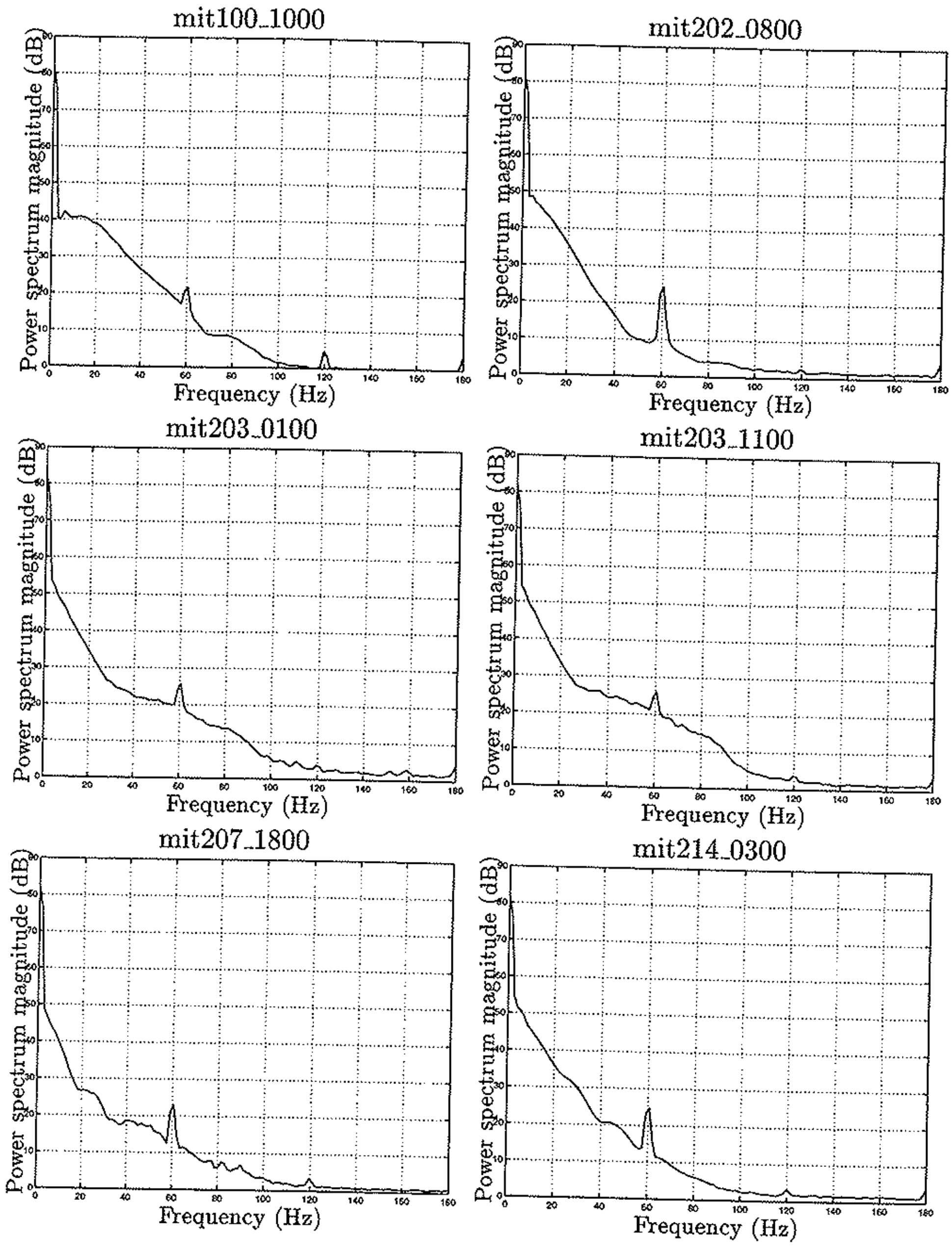


Figure A.2: Power density spectra for the different test signals.

# Appendix B

# Computation of arc lengths in the CCSP-algorithm in the polynomial interpolating reconstruction case

This appendix contains documentation of the computation of the arc lengths in the CCSP-algorithm when second order polynomials are used in reconstruction of the signal and we apply an interpolating approach. Between any two samples $i$ and $j$, a second order polynomial is fitted in a way that minimizes reconstruction error. This appendix contains a detailed mathematical deduction of the computation of these polynomials and the total arc lengths.

## Problem definition

Denote the samples taken from an ECG signal at constant intervals by $y(1)$, $y(2)$, $\ldots$, $y(N)$. We know that a second order polynomial between any two points $i$ and $j$, can be described as

$$f_{ij}(n) = a_{0ij} + a_{1ij}n + a_{2ij}n^2. \tag{B.1}$$

In our case, the function $f_{ij}$ is fitted to the data set $\{(n, y(n)) : n = i, \ldots, j)\}$ in such a way that reconstruction error is minimized.
The length of each arc $(i, j)$ is given by

$$d_{ij} = \sum_{n=i+1}^{j-1} (f_{ij}(n) - y(n))^2 = \sum_{n=i+1}^{j-1} \left(a_{0ij} + a_{1ij}n + a_{2ij}n^2 - y(n)\right)^2.$$

If we interpolate between two points $i$ and $j$, we have for each arc $(i,j)$ :

$$d_{ij} = \sum_{n=i+1}^{j-1} \left(a_{0ij} + a_{1ij}n + a_{2ij}n^2 - y(n)\right)^2, \tag{B.2}$$

$$a_{0ij} + a_{1ij}i + a_{2ij}i^2 = y(i), \tag{B.3}$$

$$a_{0ij} + a_{1ij}j + a_{2ij}j^2 = y(j). \tag{B.4}$$

The optimal parameters $a_{0ij}$, $a_{1ij}$ and $a_{2ij}$ are found by minimizing Equation (B.2) under the constraint given in Equations (B.3) and (B.4).

## Computational details

We start by expressing Equation (B.2) in terms of $a_{2ij}$. From Equation (B.3) we have

$$a_{0ij} = y(i) - a_{1ij}i - a_{2ij}i^2. \tag{B.5}$$

By subtracting Equation (B.4) from Equation (B.3) we get

$$a_{1ij}(i-j) + a_{2ij}(i^2 - j^2) = y(i) - y(j).$$

This leads to the following expression

$$a_{1ij} = \frac{y(i) - y(j)}{i - j} - a_{2ij}(i + j), \tag{B.6}$$

under the assumption that $i \neq j$, which is true for our case.
By inserting Equation (B.5) into Equation (B.2) we arrive at

$$d_{ij} = \sum_{n=i+1}^{j-1} \left(y(i) + a_{1ij}(n-i) + a_{2ij}(n^2 - i^2) - y(n)\right)^2. \tag{B.7}$$

We then insert the expression given for $a_{1ij}$ in Equation (B.6) into Equation (B.7)

$$\begin{aligned}
d_{ij} &= \sum_{n=i+1}^{j-1} \left( y(i) + \left( \frac{y(i)-y(j)}{i-j} - a_{2ij}\,(i+j) \right)(n-i) + a_{2ij}\left(n^2 - i^2\right) - y(n) \right)^2 \\
&= \sum_{n=i+1}^{j-1} \left( y(i) + \frac{n-i}{i-j}\,(y(i)-y(j)) - a_{2ij}(i+j)\,(n-i) + a_{2ij}\left(n^2-i^2\right) - y(n) \right)^2 \\
&= \sum_{n=i+1}^{j-1} \left( y(i) + \frac{n-i}{i-j}\,(y(i)-y(j)) + a_{2ij}\left(\left(n^2-i^2\right) - (i+j)(n-i)\right) - y(n) \right)^2 \\
&= \sum_{n=i+1}^{j-1} \left( y(i) + \frac{n-i}{i-j}\,(y(i)-y(j)) + a_{2ij}\,(n-i)\left((n+i)-(i+j)\right) - y(n) \right)^2 \\
&= \sum_{n=i+1}^{j-1} \left( y(i) + \frac{n-i}{i-j}\,(y(i)-y(j)) + a_{2ij}\,(n-i)\,(n-j) - y(n) \right)^2 .
\end{aligned}$$

We would then like to find $\frac{\partial d_{ij}}{\partial(a_{2ij})}$ :

$$\begin{aligned}
&\frac{\partial d_{ij}}{\partial(a_{2ij})} = \\
&\sum_{n=i+1}^{j-1} 2\left( y(i) + \frac{n-i}{i-j}(y(i)-y(j)) + a_{2ij}\,(n-i)\,(n-j) - y(n) \right)(n-i)\,(n-j) \\
&= \sum_{n=i+1}^{j-1} 2(n-i)(n-j)\left( y(i) - y(n) + \frac{n-i}{i-j}(y(i)-y(j)) + a_{2ij}(n-i)(n-j) \right).
\end{aligned}$$

By letting $\frac{\partial d_{ij}}{\partial(a_{2ij})} = 0$, we find the optimal $a_{2ij}$'s:

$$\sum_{n=i+1}^{j-1} 2(n-i)(n-j)\left( y(i) - y(n) + \frac{n-i}{i-j}(y(i)-y(j)) + a_{2ij}(n-i)(n-j) \right) = 0$$

Rearranging a bit:

$$\begin{aligned}
&2\sum_{n=i+1}^{j-1}\Big( (n-i)(n-j)y(i) - (n-i)(n-j)y(n) \\
&+ \tfrac{(n-i)^2(n-j)}{i-j}\,(y(i)-y(j)) + a_{2ij}(n-i)^2(n-j)^2 \Big) &= 0
\end{aligned}$$

$$\begin{aligned}
&y(i)\sum_{n=i+1}^{j-1}(n-i)(n-j) - \sum_{n=i+1}^{j-1}(n-i)(n-j)y(n) \\
&+\tfrac{y(i)-y(j)}{i-j}\sum_{n=i+1}^{j-1}(n-i)^2(n-j) + a_{2ij}\sum_{n=i+1}^{j-1}(n-i)^2(n-j)^2 = 0
\end{aligned}$$

This leads to

$$a_{2ij} \sum_{n=i+1}^{j-1} (n-i)^2(n-j)^2 = \sum_{n=i+1}^{j-1} (n-i)(n-j)y(n) - y(i) \sum_{n=i+1}^{j-1} (n-i)(n-j)$$
$$- \frac{y(i)-y(j)}{i-j} \sum_{n=i+1}^{j-1} (n-i)^2(n-j),$$

and thus

$$a_{2ij} = \frac{\sum_{n=i+1}^{j-1} \left( (n-i)(n-j)\left(y(n)-y(i)\right) - \frac{y(i)-y(j)}{i-j}(n-i)^2(n-j) \right)}{\sum_{n=i+1}^{j-1} (n-i)^2(n-j)^2}. \quad \text{(B.8)}$$

We then have to write out the sums in Equation (B.8) in order to see which sums of powers of $n$ that is to be computed

$$\begin{aligned} \sum_{n=i+1}^{j-1} (n-i)^2(n-j)^2 &= \sum_{n=i+1}^{j-1} (n^2 - 2ni + i^2)(n^2 - 2nj + j^2) \\ &= \sum_{n=i+1}^{j-1} \left(n^4 - 2(i+j)n^3 + (i^2+j^2+4ij)n^2 \right. \\ &\quad \left. - 2(i^2j + j^2i)n + i^2j^2\right). \end{aligned}$$

The denominator of $a_{2ij}$ can thus be written as

$$\sum_{n=i+1}^{j-1} \left(\gamma_{0ij} + \gamma_{1ij}n + \gamma_{2ij}n^2 + \gamma_{3ij}n^3 + \gamma_{4ij}n^4\right), \quad \text{(B.9)}$$

where

$$\begin{aligned} \gamma_{0ij} &= i^2j^2, & \gamma_{3ij} &= -2(i+j), \\ \gamma_{1ij} &= -2ij(i+j), & \gamma_{4ij} &= 1. \\ \gamma_{2ij} &= i^2+j^2+4ij, \end{aligned}$$

We then have to examine the numerator of $a_{2ij}$ closer.

$$\sum_{n=i+1}^{j-1} \left( (n^2 - jn - in + ij)\left(y(n) - y(i)\right) - A_{ij}(n^2 - 2in + i^2)(n-j) \right),$$

where

$$A_{ij} = \frac{y(i) - y(j)}{i - j}.$$

After a bit manipulation we arrive at

$$\sum_{n=i+1}^{j-1} \left(n^2 - (i+j)n + ij\right) y(n) + \sum_{n=i+1}^{j-1} \Big(-A_{ij}n^3 + \left(A_{ij}(2i+j) - y(i)\right) n^2$$
$$+ \left((i+j)y(i) - A_{ij}i(2j+i)\right) n + ij\left(A_{ij}i - y(i)\right) \Big).$$

The numerator of $a_{2ij}$ can thus be written as

$$\sum_{n=i+1}^{j-1} \left(\alpha_{0ij} + \alpha_{1ij}n + \alpha_{2ij}n^2\right) y(n) + \sum_{n=i+1}^{j-1} \left(\beta_{0ij} + \beta_{1ij}n + \beta_{2ij}n^2 + \beta_{3ij}n^3\right),$$

where

$$\begin{array}{llll} \alpha_{0ij} &= ij, & \beta_{0ij} &= ij(A_{ij}i - y(i)), \\ \alpha_{1ij} &= -(i+j), & \beta_{1ij} &= (i+j)y(i) - A_{ij}i(2j+i), \\ \alpha_{2ij} &= 1, & \beta_{2ij} &= A_{ij}(2i+j) - y(i), \\ A_{ij} &= \frac{y(i)-y(j)}{i-j}, & \beta_{3ij} &= -A_{ij}. \end{array}$$

To summarize, we have the following expression for the coefficients in the second order polynomial:

$$\begin{aligned} a_{2ij} &= \frac{\sum_{n=i+1}^{j-1} \left(\alpha_{0ij} + \alpha_1 n + \alpha_2 n^2\right) y(n)}{\sum_{n=i+1}^{j-1} \left(\gamma_{0ij} + \gamma_{1ij}n + \gamma_{2ij}n^2 + \gamma_{3ij}n^3 + \gamma_{4ij}n^4\right)} \\ &\quad + \frac{\sum_{n=i+1}^{j-1} \left(\beta_{0ij} + \beta_{1ij}n + \beta_{2ij}n^2 + \beta_{3ij}n^3\right)}{\sum_{n=i+1}^{j-1} \left(\gamma_{0ij} + \gamma_{1ij}n + \gamma_{2ij}n^2 + \gamma_{3ij}n^3 + \gamma_{4ij}n^4\right)}, \\ a_{1ij} &= A_{ij} - a_{2ij}(i+j), \\ a_{0ij} &= y(i) - a_{1ij}i - a_{2ij}i^2. \end{aligned}$$

We would like to substitute the terms including sums of powers of $n$ with closed form expressions. In oder to do so we need the following expressions in closed form :

$$\sum_{n=i+1}^{j-1} n^p, p = 0, 1, 2, 3, 4. \qquad \text{(B.10)}$$

This is computed in the following way:

$$\sum_{n=i+1}^{j-1} n^p = \sum_{n=1}^{j-1} n^p - \sum_{n=1}^{i} n^p, \quad p = 0, 1, 2, 3, 4.$$

From Rottmann's table of formulas [81] we have :

$$\begin{aligned}
\sum_{n=1}^{n} n &= \frac{n(n+1)}{2}, \\
\sum_{n=1}^{n} n^2 &= \frac{n(n+1)(2n+1)}{6}, \\
\sum_{n=1}^{n} n^3 &= \left(\frac{n(n+1)}{2}\right)^2, \\
\sum_{n=1}^{n} n^4 &= \frac{n^5}{5} + \frac{n^4}{2} + \frac{n^3}{3} - \frac{n}{30}.
\end{aligned}$$

In our case we get :

$$\begin{aligned}
\sum_{n=i+1}^{j-1} n^0 &= j - i - 1, \\
\sum_{n=i+1}^{j-1} n &= \sum_{n=1}^{j-1} n - \sum_{n=1}^{i} n = \frac{(i+j)(j-i-1)}{2}, \\
\sum_{n=i+1}^{j-1} n^2 &= \sum_{n=1}^{j-1} n^2 - \sum_{n=1}^{i} n^2 = \frac{j(j-1)(2j-1) - i(i+1)(2i+1)}{6} \\
&= \frac{(j-i-1)(2j^2 - j + 2ij + i + 2i^2)}{6}, \\
\sum_{n=i+1}^{j-1} n^3 &= \sum_{n=1}^{j-1} n^3 - \sum_{n=1}^{i} n^3 = \left(\frac{j(j-1)}{2}\right)^2 - \left(\frac{i(i+1)}{2}\right)^2 \\
&= \frac{(i+j)(j-i-1)(j^2 + i^2 - j + i)}{4},
\end{aligned}$$

$$
\begin{aligned}
\sum_{n=i+1}^{j-1} n^4 = \sum_{n=1}^{j-1} n^4 - \sum_{n=1}^{i} n^4 &= \frac{(j-1)^5}{5} + \frac{(j-1)^4}{2} + \frac{(j-1)^3}{3} - \frac{j-1}{30} \\
&\quad - \left(\frac{i^5}{5} + \frac{i^4}{2} + \frac{i^3}{3} - \frac{i}{30}\right) \\
&= \frac{6(j^5-i^5) - 15(j^4+i^4) + 10(j^3-i^3) + i - j}{30}.
\end{aligned}
$$

The total expression for $a_{2ij}$ becomes :

$$
a_{2ij} = \frac{\sum_{n=i+1}^{j-1}(\alpha_{0ij} + \alpha_{1ij}n + \alpha_{2ij}n^2)y(n) + \eta_{1ij}}{\eta_{2ij}}, \tag{B.11}
$$

where

$$
\begin{aligned}
\eta_{1ij} &= \sum_{n=i+1}^{j-1} \left(\beta_{0ij} + \beta_{1ij}n + \beta_{2ij}n^2 + \beta_{3ij}n^3\right) \\
&= \beta_{0ij}(j-i-1) + \beta_{1ij}\frac{(i+j)(j-i-1)}{2} \\
&\quad + \beta_{2ij}\frac{(j-i-1)(2j^2-j+2ij+i+2i^2)}{6} \\
&\quad + \beta_{3ij}\frac{(i+j)(j-i-1)(j^2+i^2-j+i)}{4}, \\
\eta_{2ij} &= \sum_{n=i+1}^{j-1} \left(\gamma_{0ij} + \gamma_{1ij}n + \gamma_{2ij}n^2 + \gamma_{3ij}n^3 + \gamma_{4ij}n^4\right) \\
&= \gamma_{0ij}(j-i-1) + \gamma_{1ij}\frac{(i+j)(j-i-1)}{2} \\
&\quad + \gamma_{2ij}\frac{(j-i-1)(2j^2-j+2ij+i+2i^2)}{6} \\
&\quad + \gamma_{3ij}\frac{(i+j)(j-i-1)(j^2+i^2-j+i)}{4} \\
&\quad + \gamma_{4ij}\frac{6(j^5-i^5) - 15(j^4+i^4) + 10(j^3-i^3) + i - j}{30}.
\end{aligned}
$$

Besides we have

$$
\begin{aligned}
a_{1ij} &= A_{ij} - a_{2ij}(i+j), \\
a_{0ij} &= y(i) - a_{1ij}i - a_{2ij}i^2.
\end{aligned}
$$

$a_{2ij}$, $a_{1ij}$ and $a_{0ij}$ have to be computed for all legal combinations of $i$ and $j$. $\eta_{1ij}$ and $\eta_{ij}$ are expressions in $i$, $j$, $y(i)$ and $y(j)$ and hence all these are computed in $\mathcal{O}(N^2)$ time. By defining $\Delta_{pj} = \sum_{n=1}^{j} n^p y(n)$, $p = 0, 1, 2$, we see that $\Delta_{p1}, \dots, \Delta_{pN}$ are computed in $\mathcal{O}(N)$ time. Next, we compute

$$a_{2ij} = \frac{\sum_{p=0}^{2} \alpha_{pij}(\Delta_{p,j-1} - \Delta_{pi}) + \eta_{1ij}}{\eta_{2ij}}, \tag{B.12}$$

$1 < j < N$, involving $\mathcal{O}(N^2)$ operations.
For every legal combination of $i$ and $j$ we have to compute $d_{ij}$ as well since this is the quantity we are seeking:

$$\begin{aligned} d_{ij} &= \sum_{n=i+1}^{j-1} \left(a_{0ij} + a_{1ij}n + a_{2ij}n^2 - y(n)\right)^2 \\ &= \sum_{n=i+1}^{j-1} \left(a_{0ij}^2 + 2a_{0ij}a_{1ij}n + (2a_{0ij}a_{2ij} + a_{1ij}^2)n^2 + 2a_{1ij}a_{2ij}n^3 + a_{2ij}^2 n^4\right) \\ &\quad + \sum_{n=i+1}^{j-1} \left(-2a_{0ij} - 2a_{1ij}n - 2a_{2ij}n^2 + y(n)\right) y(n). \end{aligned}$$

By using the same explicit expressions as before for the sums we get:

$$\begin{aligned} d_{ij} &= a_{0ij}^2(j-i-1) + a_{0ij}a_{1ij}(i+j)(j-i-1) \\ &\quad + (2a_{0ij}a_{2ij} + a_{1ij}^2)\frac{(j-i-1)(2j^2 - j + 2ij + i + 2i^2)}{6} \\ &\quad + a_{1ij}a_{2ij}\frac{(i+j)(j-i-1)(j^2+i^2-j+i)}{2} \\ &\quad + a_{2ij}^2\frac{6(j^5-i^5) - 15(j^4+i^4) + 10(j^3-i^3) + i - j}{30} \\ &\quad - \sum_{n=i+1}^{j-1} \left(2a_{0ij} + 2a_{1ij}n + 2a_{2ij}n^2 - y(n)\right) y(n). \end{aligned} \tag{B.13}$$

The five first terms in Equation (B.13) are expressions in $a_{0ij}$, $a_{1ij}$, $a_{2ij}$, $i$, $j$, $y(i)$ and $y(j)$. When $a_{pij}, p = 0, 1, 2$ are available, all these are computed in $\mathcal{O}(N^2)$ time. By applying a similar procedure as in the computation of $a_{2ij}$ and define $\Delta_{pj}^1 = \sum_{n=1}^{j} n^p y(n)$, $p = 0, 1, 2$, and $\Delta_j^2 = \sum_{n=1}^{j} y^2(n)$ we see that $\Delta_{p1}^1, \dots, \Delta_{pN}^1$ and $\Delta_1^2, \dots, \Delta_N^2$ are computed in $\mathcal{O}(N^2)$ time. The last term of $d_{ij}$ is then computed as

$$d_{ij} = -2\sum_{p=0}^{2} a_{pij}n^p(\Delta_{p,j-1}^1 - \Delta_{pi}^1) + \Delta_{j-1}^2 - \Delta_i^2, \tag{B.14}$$

$1 < j < N$, involving $\mathcal{O}(N^2)$ operations.

## Summary

For each combination of $i$ and $j$, $i = 1, \dots, N-1$, $j = i+1, \dots, N$ the following has to be computed:

$$a_{2ij} = \frac{\sum_{n=i+1}^{j-1} (\alpha_{0ij} + \alpha_{1ij} n + \alpha_{2ij} n^2) y(n) + \eta_{1ij}}{\eta_{2ij}}, \tag{B.15}$$

where

$$\begin{aligned}
\eta_{1ij} &= \sum_{n=i+1}^{j-1} \left(\beta_{0ij} + \beta_{1ij} n + \beta_{2ij} n^2 + \beta_{3ij} n^3\right) \\
&= \beta_{0ij}(j-i-1) + \beta_{1ij} \frac{(i+j)(j-i-1)}{2} \\
&\quad + \beta_{2ij} \frac{(j-i-1)(2j^2 - j + 2ij + i + 2i^2)}{6} \\
&\quad + \beta_{3ij} \frac{(i+j)(j-i-1)(j^2 + i^2 - j + i)}{4}, \\
\eta_{2ij} &= \sum_{n=i+1}^{j-1} \left(\gamma_{0ij} + \gamma_{1ij} n + \gamma_{2ij} n^2 + \gamma_{3ij} n^3 + \gamma_{4ij} n^4\right) \\
&= \gamma_{0ij}(j-i-1) + \gamma_{1ij} \frac{(i+j)(j-i-1)}{2} \\
&\quad + \gamma_{2ij} \frac{(j-i-1)(2j^2 - j + 2ij + i + 2i^2)}{6} \\
&\quad + \gamma_{3ij} \frac{(i+j)(j-i-1)(j^2 + i^2 - j + i)}{4} \\
&\quad + \gamma_{4ij} \frac{6(j^5 - i^5) - 15(j^4 + i^4) + 10(j^3 - i^3) + i - j}{30}.
\end{aligned}$$

The other two optimal coefficients are given as :

$$a_{1ij} = A_{ij} - a_{2ij}(i+j), \tag{B.16}$$

$$a_{0ij} = y(i) - a_{1ij} i - a_{2ij} i^2. \tag{B.17}$$

The $\alpha'_{kij}s$, $\beta'_{kij}s$ and $\gamma'_{kij}s$ are given by :

$$\begin{aligned}
\alpha_{0ij} &= ij, & \gamma_{0ij} &= i^2 j^2, \\
\alpha_{1ij} &= -(i+j), & \gamma_{1ij} &= -2ij(i+j), \\
\alpha_{2ij} &= 1, & \gamma_{2ij} &= i^2 + j^2 + 4ij, \\
\beta_{0ij} &= ij(A_{ij} i - y(i)), & \gamma_{3ij} &= -2(i+j), \\
\beta_{1ij} &= (i+j) y(i) - A_{ij} i (2j + i), & \gamma_{4ij} &= 1, \\
\beta_{2ij} &= A_{ij}(2i + j) - y(i), & A_{ij} &= \tfrac{y(i) - y(j)}{i - j}, \\
\beta_{3ij} &= -A_{ij}.
\end{aligned}$$

Finally, $d_{ij}$ is given by :

$$\begin{aligned}
d_{ij} = {} & a_{0ij}^2(j-i-1) + a_{0ij}a_{1ij}(i+j)(j-i-1) \\
& + (2a_{0ij}a_{2ij} + a_{1ij}^2)\frac{(j-i-1)(2j^2-j+2ij+i+2i^2)}{6} \\
& + a_{1ij}a_{2ij}\frac{(i+j)(j-i-1)(j^2+i^2-j+i)}{2} \\
& + a_{2ij}^2\frac{6(j^5-i^5)-15(j^4+i^4)+10(j^3-i^3)+i-j}{30} \\
& - \sum_{n=i+1}^{j-1}\left(2a_{0ij}+2a_{1ij}n+2a_{2ij}n^2-y(n)\right)y(n).
\end{aligned}$$

# Bibliography

[1] S. O. Aase, "Filter bank design for subband compression of ECG signals," in *Proc. of Norwegian Signal Processing Symp.*, (Stavanger, Norway), pp. 113–118, Aug 1995.

[2] S. O. Aase, "Filter bank design for subband ECG compression," in *IEEE Engineering in Medicine and Biology*, (Amsterdam, The Netherlands), pp. 1382–1383, Oct. 1996.

[3] S. O. Aase, R. Nygaard, and J. H. Husøy, "A comparative study of some novel ECG data compression techniques," in *Proc. of Norwegian Signal Processing Symp.*, (Vigsø, Denmark), pp. 273–276, June 1998.

[4] S. O. Aase, R. Nygaard, J. H. Husøy, and D. Haugland, "Optimised time and frequency domain methods for ECG signal compression," *Applied Signal Processing*, vol. 5, pp. 210–225, October 1998.

[5] S. O. Aase, *Image Subband Coding Artifacts: Analysis and Remedies.* PhD thesis, The Norwegian Institute of Technology, Mar. 1993.

[6] J. Abenstein and W. Tompkins, "New data-reduction algorithm for real-time ECG analysis," *IEEE Trans. on Biomedical Engineering*, vol. BME-29, pp. 43–48, 1982.

[7] N. Ahmed, P. J. Milne, and S. G. Harris, "Electrocardiographic data compression via orthogonal transforms.," *IEEE Trans. on Biomedical Engineering*, vol. 22, pp. 484–487, 1975.

[8] A. N. Akansu and R. A. Haddad, *Multiresolution Signal Decomposition.* San Diego: Academic Press, 1992.

[9] Y. Aneja, V. Aggarwal, and K. Nair, "Shortest chain subject to side conditions," *Networks*, vol. 13, pp. 295–302, 1983.

[10] Y. Aneja and K. Nair, "The constrained shortest path problem," *Nav. Res. Log. Q*, vol. 25, pp. 549–553, 1978.

[11] M. C. Aydin, A. E. Çetin, and H. Kőymen, "ECG data compression by sub–band coding," *Electronics Letter*, vol. 27, pp. 359–360, February 1991.

[12] M. R. Banham and J. C. Brailean, "An overview of the MPEG-4 standard: Enabling digital multimedia compression," in *Proc. of International conference on advanced science and technology*, (Schaumburg, IL, USA), pp. 2–19, 1997.

[13] R. C. Barr, "Adaptive sampling of cardiac waveforms," *Journal of Electrocardiography*, vol. 21, pp. 57–60, 1988.

[14] J. Beasley and N. Christofides, "An algorithm for the resource constrained shortest path problem," *Networks*, vol. 19, pp. 379–394, 1989.

[15] R. Bellman, "On the approximation of curves by line segments using dynamic programming," *Communications of the ACM*, vol. 4, p. 284, April 1961.

[16] V. Bhaskaran, B. K. Natarjan, and K. Konstantinides, "Optimal piecewise-linear compression of images," in *Data Compression Conference*, (Cliff Lodge, Utah, USA), pp. 168–177, 1993.

[17] S. M. Blanchard and R. C. Barr, "Comparison of methods for adaptive sampling of cardiac electrograms and electrocardiograms," *Medical & Biological Engineering & Computing*, vol. 23, pp. 377–386, July 1985.

[18] B.Sigurd and E. Sandøe, *Klinisk elektrokardiologi.* Ockenheimer Chaussee 5, D-6530 Bingen: Publishing Partners Verlags GmbH, 1991.

[19] J. Chen, S. Itoh, and T. Hashimoto, "ECG data compression by using wavelet transform," *IEICE Transactions on Information and Systems*, vol. E76-D, pp. 1454–1461, December 1993.

[20] T. Cormen, C. Leiserson, and R. Rivest, *Introduction to algorithms.* New York: McGraw-Hill, 1991.

[21] T. Cover and J. Joy, *Elements of Information Theory.* New York: Wiley, 1991.

[22] J. Cox, F. Noelle, H. Fozzard, and G. Oliver, "AZTEC: A preprocessing program for real-time ECG rhythm analysis," *IEEE Trans. on Biomedical Engineering*, vol. BME-15, pp. 128–129, 1968.

[23] L. D. Davisson, "The Fan method of data compression," in *1966 Goddard summer workshop*, (NASA TM X-55472, X-700-67-94, Final rep.), pp. 23–30, 1967.

[24] E. Dijkstra, "A note on two problems in connection with graphs," *Numerische Math.*, vol. 1, pp. 269–271, 1959.

[25] D. A. Dipersio and R. C. Barr, "Evaluation of the fan method of adaptive sampling on human electrocardiograms," *Medical & Biological Engineering & Computing*, pp. 401–410, September 1985.

[26] M. Eden and M. Kocher, "On the performance of contour coding algorithm in the context of image coding. Part I: Contour segment coding," *Signal Processing*, vol. 8, pp. 381–386, July 1985.

[27] H. Everett, "Generalized Lagrange multiplier method for solving problems of optimum allocation of resources," *Operations Research*, vol. 11, pp. 399–417, 1963.

[28] H. Freeman, "On the encoding of arbitrary geometric configurations," *IRE Trans. Electron. Comput.*, vol. EC-10, pp. 260–268, June 1961.

[29] L. W. Gardenhire, *Data compression for biomedical telemetry.* C. A. Caceres, New York: Academic, 1965,ch. 11.

[30] L. W. Gardenhire, "Redundancy reduction : the key to adaptive telemetry," *Proc. Nat. Tel. Conf. Los Angeles CA.*, pp. 1–16, 1964.

[31] C. F. Gerald and P. O. Wheatley, *Applied numerical analysis.* Addison Wesley, fourth. ed., 1990.

[32] A. Gersho, *Vector Quantization and Signal Compression.* Boston: Kluwer Academic Publishers, 1992.

[33] P. S. Hamilton and W. J. Tompkins, "Compression of the ambulatory ECG by average beat subtraction and residual differencing," *IEEE Trans. on Biomedical Engineering*, vol. 38, pp. 253–259, March 1991.

[34] G. Handler and I. Zang, "A dual algorithm for the constrained shortest path problem," *Networks*, vol. 10, pp. 293–310, 1980.

[35] T. Haugan, "Compression of ECG signals using optimized FIR filter banks and entropy allocation," Master's thesis, Rogaland University Center, 1995 (in Norwegian).

[36] D. Haugland, J. G. Heber, and J. H. Husøy, "Compressing data by shortest path methods," in *Operations Research Proceedings 1996*, (Springer, Berlin, Germany), pp. 145–150, 1997.

[37] D. Haugland, J. Heber, and J. Husøy, "Optimisation algorithms for ECG data compression," *Medical & Biological Engineering & Computing*, vol. 35, pp. 420–424, July 1997.

[38] J. G. Heber, D. Haugland, and J. H. Husøy, "An efficient implementation of an optimal time domain ECG coder," in *Proc. of IEEE Engineering in Medicine and Biology*, (Amsterdam, The Netherlands), pp. 1384–1385, Oct. 1996.

[39] J. G. Heber, *Compression of ECG signals – Time and frequency domain approaches.* PhD thesis, Aalborg Universitet/Høgskolen i Stavanger, Aug. 1996.

[40] J. A. Horst and I. Beichl, "A simple algorithm for efficient piecewise linear approximation of space curves," in *Proc. of Int. Conference on Image Processing*, (Santa Barbara, California, USA), pp. Vol. 2, 744–747, October 1997.

[41] M. Hötter, "Object-oriented analysis-synthesis coding based on moving two-dimensional objects," *Signal Processing: Image Communication*, vol. 2, pp. 409–428, December 1990.

[42] J. H. Husøy and T. Gjerde, "Computationally efficient subband coding of ECG signals," *Medical Engineering and Physics*, vol. 18, pp. 132–142, Mar. 1996.

[43] S. H. Ingebrigtsen, "Komprimering av EKG signaler ved hjelp av stykkevis polynomisk approksimasjon," Master's thesis, Høgskolen i Stavanger, 1998 (in Norwegian).

[44] J. Ishijima, S. B. Shin, G. H. Hostetter, and J. Skalansky, "Scan along polygon approximation for data compression of electrocardiograms," *IEEE Trans. on Biomedical Engineering*, vol. 30, pp. 723–729, 1983.

[45] A. K. Jain, *Fundamentals of Digital Image Processing.* Englewood Cliffs: Prentice-Hall, 1989.

[46] S. M. S. Jalaleddine, C. G. Hutchens, R. D. Strattan, and W. A. Coberly, "ECG data compression techniques – a unified approach," *IEEE Trans. on Biomedical Engineering*, vol. 37, pp. 329–343, April 1990.

[47] N. S. Jayant and P. Noll, *Digital Coding of Waveforms.* Englewood Cliffs: Prentice-Hall, 1984.

[48] Joint Photographic Experts Group ISO/IEC, JTC/SC/WG8, CCITT SGVIII, "JPEG technical specifications, revision 5," *Report JPEG-8-R5*, Jan. 1990.

[49] H. Joksch, "The shortest route problem with constraints," *J. Math. Anal. Appl.*, vol. 14, pp. 191–197, 1966.

[50] A. K. Katsaggelos, L. P. Kondi, F. W. Meier, J. Osterman, and G. M. S. and, "MPEG-4 and rate-distortion-based shape-coding techniques," *Proc. IEEE*, vol. 86, pp. 1126–1154, June 1998.

[51] L. P. Kondi, F. W. Meier, G. M. Schuster, and A. K. Katsaggelos, "Joint optimal object shape estimation and encoding," in *Proc. SPIE's Visual Communications and Image Processing*, (San Jose, California, USA), pp. 14–25, January 1998.

[52] W. S. Kuklinski, "Fast Walsh transform data-compression algorithm; ECG applications," *Medical & Biological Engineering & Computing*, vol. 21, pp. 465–472, July 1983.

[53] M. Kunt, A. Ikonomopoulos, and M. Kocher, "Second generation image-coding techniques," *Proc. IEEE*, vol. 73, pp. 549–574, April 1985.

[54] H. Lee and K. M. Buckley, "ECG data compression using cut and align beats approach and 2-D transform," *IEEE Transactions on Biomedical Engineering*, vol. 46, pp. 556–564, May 1999.

[55] H. Malvar, *Signal Processing with Lapped Transforms.* Artech House, 1992.

[56] C. P. Mammen and B. Ramamurthi, "Vector quantization for compression of multichannel ECG," *IEEE Trans. on Biomedical Engineering*, vol. 37, pp. 821–825, September 1990.

[57] F. Meier, G. Schuster, and A. K. Katsaggelos, "An efficient boundary encoding scheme which is optimal in the rate distortion sense," in *Proc. of Int. Conference on Image Processing*, (Santa Barbara, California, USA), pp. Vol.2, 9–12, October 1997.

[58] G. Melnikov, G. M. Schuster, and A. K. Katsaggelos, "Simultaneous optimal boundary encoding and variable-length code selection," in *Proc. of*

*Int. Conference on Image Processing*, (Chicago, Illinois, USA), pp. 256–260, October 1998.

[59] G. Melnikov, G. M. Schuster, and A. K. Katsaggelos, "Shape coding using temporal correlation and joint VLC optimization," *IEEE Trans. on Circuits and System for Video Technology*, vol. 10, August 2000.

[60] G. Moody, *MIT-BIH Arrhythmia Database CD-ROM (second edition), overview.* Massachusetts Institute of Technology, August 1992.

[61] W. Mueller, "Arrhythmia detection program for an ambulatory ECG monitor," *Biomed. Sci. Instrument*, vol. 14, pp. 81–85, 1978.

[62] I. S. N. Murthy and B. Madhukar, "Analysis of ECG from pole-zero models," *IEEE Trans.Biomed.Eng.*, vol. 39, pp. 741–751, 1992.

[63] H. Musmann, M. Hötter, and J. Ostermann, "Object-oriented analysis-synthesis coding of moving images," *Signal Processing: Image Communication*, vol. 1, pp. 117–138, October 1989.

[64] R. Nygaard and D. Haugland, "Multiple error measures in ECG data compression," in *Proc. of Norwegian Signal Processing Symp.*, (Tromsø, Norway), pp. 134–139, May 1997.

[65] R. Nygaard and D. Haugland, "Complete coding scheme using optimal time domain ECG compression methods," in *Proc. of European Signal Processing Conf. (EUSIPCO)*, (Island of Rhodes, Greece), pp. 2473–2476, September 1998.

[66] R. Nygaard and D. Haugland, "Compressing ECG signals by piecewise polynomial approximation," in *Proc. of International Conference on Acoustics, Speech and Signal Processing*, (Seattle, Washington, USA), pp. 1809–1812, May 1998.

[67] R. Nygaard, J. H. Husøy, and D. Haugland, "Compression of image contours using combinatorial optimization," in *Proc. of Int. Conference on Image Processing*, (Chicago, Illinois, USA), pp. 266–270, October 1998.

[68] R. Nygaard, J. H. Husøy, and D. Haugland, "Signal compression by second order polynomials and piecewise non-interpolating approximation," in *Proc. of Norwegian Signal Processing Symp.*, (Oslo, Norway), September 1999.

[69] R. Nygaard, J. H. Husøy, D. Haugland, and S. O. Aase, "Signal compression by piecewise linear non-interpolating approximation," in *Proc. of International Conference on Acoustics, Speech and Signal Processing*, (Phoenix, Arizona, USA), pp. 1273–1276, March 1999.

[70] R. Nygaard, G. Melnikov, and A. K. Katsaggelos, "Rate distortion optimal ECG signal compression," in *Proc. of Int. Conference on Image Processing*, (Kobe, Japan), pp. 348–351, October 1999.

[71] R. Nygaard, G. Melnikov, and A. K. Katsaggelos, "A rate distortion optimal ECG coding algorithm," *IEEE Transactions on Biomedical Engineering*, Accepted for publication, June 2000.

[72] S. Olmos, M. Millán, J. García, and P. Laguna, "ECG data compression with the Karhunen-Loève transform," in *Computers in Cardiology Proceedings*, (Indianapolis, USA), pp. 253–256, September 1996.

[73] M.-W. OnLine, *WWWebster Dictionary.* http://www.m-w.com/: Merriam-Webster Incorporated, 2000.

[74] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and complexity.* Englewood Cliffs, New Jersey: Prentice-Hall, inc., 1982.

[75] A. E. Pollard and R. C. Barr, "Adaptive sampling of intracellular and extracellular cardiac potentials with the Fan method," *Medical & Biological Engineering & Computing*, vol. 25, pp. 261–268, May 1987.

[76] T. A. Ramstad, S. O. Aase, and J. H. Husøy, *Subband Compression of Images – Principles and Examples.* North Holland: ELSEVIER Science Publishers BV, 1995.

[77] B. R. S. Reddy and I. S. N. Murthy, "ECG data compression using Fourier descriptions," *IEEE Trans. on Biomedical Engineering*, vol. 33, pp. 428–434, 1986.

[78] C. Ribeiro and M. Minoux, "A heuristic approach to hard constrained shortest path problems," *Discrete Appl. Math.*, vol. 10, pp. 125–137, 1985.

[79] C. Rosenberg, "A lossy image compression algorithm based on nonuniform sampling and interpolation of image intensity surfaces," Master's thesis, Dept. of Electrical and Computer Science, Mass. Inst. of Tech., 1990.

[80] M. Rosseel, "Comments on a paper by R. Saigal: A constrained shortest route problem," *Operations Research*, vol. 16, pp. 1232–1234, 1968.

[81] K. Rottman, *Matematisk formelsamling.* Bracan Forlag (Norsk utgave), 1995.

[82] R. Saigal, "A constrained shortest route problem," *Operations Research*, vol. 16, pp. 205–209, 1968.

[83] W. B. Saunders, *Dorland's Illustrated Medical Dictionary.* The Curtis Center, Independence Square West, Philadephia, PA 19106, USA: W. B. Saunders Company, 28 edition ed., 1994.

[84] D. Saupe, "Optimal piecewise linear image coding," in *Proc. SPIE's Visual Communications and Image Processing*, (San Jose, California, USA), pp. 747–760, January 1998.

[85] G. M. Schuster and A. K. Katsaggelos, *Rate-Distortion Based Video Compression, Optimal Video Frame Compression and Object Boundary Encoding.* The Netherlands: Kluwer Academic Publishers, 1997.

[86] G. M. Schuster and A. K. Katsaggelos, "An optimal polygonal boundary encoding scheme in the rate distortion sense," *IEEE Trans. Image Processing*, vol. 7, pp. 13–26, 1998.

[87] G. M. Schuster, G. Melnikov, and A. K. Katsaggelos, "Optimal shape coding technique," *IEEE Signal Processing Magazine*, pp. 91–108, November 1998.

[88] G. M. Schuster, G. Melnikov, and A. K. Katsaggelos, "A review of the minimum maximum criterion for optimal bit allocation among dependent quantizers," *IEEE Trans. Multimedia Signal Processing*, vol. 1, pp. 3–17, March 1999.

[89] Y. Shoham and A. Gersho, "Efficient bit allocation for an arbitrary set of quantizers," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 36, pp. 1445–1453, Sept. 1988.

[90] S. C. Tai, "Slope – a real-time ECG data compressor," *Medical & Biological Engineering & Computing*, vol. 29, pp. 175–179, March 1991.

[91] S. C. Tai, "Six-band sub-band coder on ECG waveforms," *Medical & Biological Engineering & Computing*, vol. 30, pp. 187–192, March 1992.

[92] S. C. Tai, "AZTDIS – a two phase real-time ECG data compressor," *Journal of Biomedical Engineering*, vol. 15, pp. 510–515, Nov. 1993.

[93] W. J. Tompkins, ed., *Biomedical Digital Signal Processing: C – Language Examples and Laboratory Experiments for the IBM PC.* Prentice – Hall Inc., 1993.

[94] P. P. Vaidyanathan, *Multirate Systems and Filter Banks.* Englewood Cliffs: Prentice Hall, 1993.

[95] M. Wan and Z. Tang, "A new algorithm for constructing dynamic convex hull in the plane," in *Proc. SPIE's Visual Communications and Image Processing*, pp. 273–282, 1996, vol. 2644.

[96] M. E. Womble and A. M. Zied, "A statistical approach to ECG/VCG data compression," in *Optimization of Computer ECG Processing*, (Halifax, Canada), pp. 91–101, 1980.

[97] Y. Ziegel, A. Cohen, and A. Katz, "A diagnostic meaningful distortion measure for ECG compression," in *Proc. of 19th Conf. of Electrical & Electronic Eng.*, (Jerusalem, Israel), pp. 117–120, Nov. 1996.