UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ELETRÔNICA

Compressão de Sinais de Eletrocardiograma Utilizando o Algoritmo MMP

Autor:
Viviane de Medeiros Calaça Gomes

Orientador:
Eduardo Antônio Barros da Silva, Ph.D.

Co-Orientador:
Murilo Bresciani de Carvalho, D.Sc.

Examinador:
Gelson Vieira Mendonça, Ph.D.

Examinador:
Eddie Filho, M.Sc.

DEL
Agosto de 2004

Agradecimentos

A Deus, pela conclusão deste projeto e de mais uma etapa importante em minha vida.

Aos meus pais, Sueli e Vinicius, pelo suporte e apoio ao longo destes anos de faculdade, pelo carinho, por estarem sempre ao meu lado e compreenderem o meu humor instável nas fases mais difíceis deste projeto.

A minha avó Dilma, que acompanha meus estudos desde pequena e principalmente pelo apoio no término deste projeto.

Ao meu irmão Marcus Vinicius, pela amizade e carinho e por ter cedido seu quarto inúmeras vezes para que eu usasse o computador, muitas vezes até altas horas da noite.

Ao professor Eduardo, meu orientador neste projeto, que sempre esteve disponível e não deixou que as dificuldades encontradas ao longo do projeto me deixassem desistir.

Ao Murilo Bresciani de Carvalho e ao Eddie Filho pela ajuda na implementação do MMP.

Aos colegas do LPS e amigos Fábio Pacheco Freeland e Solimar de Souza Silva, pela constante ajuda ao longo do desenvolvimento deste projeto, com idéias, sugestões e explicações.

Aos amigos do LPS, em especial a galera do cgtotal, agradeço pela amizade, pelo carinho, pela ajuda e por criarem um ambiente de trabalho agradável e estimulante.

Ao amigo Edson Hiroshi Watanabe, por ter me ajudado a revisar o texto deste projeto e pelo apoio constante nos momentos de desânimo.

A todos os amigos, que sempre estiveram ao meu lado, pela amizade e compreensão pela minha ausência nas etapas finais de conclusão deste projeto.

Resumo

Este trabalho consistiu em apresentar o algoritmo MMP, utilizado para compressão de dados, e utilizá-lo para compressão de sinais de eletrocardiograma.

O MMP é baseado no casamento de padrões multiescalas e enquanto codifica o sinal constrói um dicionário com versões contraídas, dilatadas e concatenadas de padrões previamente codificados, aprendendo desta forma os padrões típicos do sinal.

Apresentaremos o algoritmo simples desenvolvido originalmente para compressão de imagens e as alterações implementadas a fim de adaptar o algoritmo para compressão de sinais de ECG.

Com o objetivo de aumentar a eficiência do MMP para compressão de sinas de ECG, propomos algumas modificações ao MMP: tratar o sinal de ECG como imagem, retirar o nível DC do sinal e equalizar o sinal antes de aplicá-lo ao MMP.

Ao longo do trabalho apresentaremos os resultados obtidos com cada uma das novas propostas e discutimos a eficiência dos algoritmos.

Palavras-Chave

ECG

compressão

taxa-distorção

MMP

Sumário

Capítulo 1

Introdução

A evolução tecnológica tem revolucionado as mais diversas áreas. Desde aplicações pessoais a serviços profissionais, o processo de digitalização criou novos hábitos e serviços. Áudio e vídeo digital, telefonia móvel e internet já fazem parte do nosso cotidiano. Essa revolução digital chegou também na Medicina, modernizando a forma de comunicação entre os pacientes e seus médicos. O desenvolvimento dos sistemas de comunicação, das redes digitais de alto desempenho, a popularização da internet e a queda do custo dos sistemas computacionais impulsionaram a implementação de sistemas de telemedicina.

A telemedicina é a prática de armazenamento, transmissão e manipulação de informações para suporte de serviços de saúde através do uso das modernas técnicas de comunicação. Surgiu em torno de 1962, quando a Agência Espacial Americana criou um sistema para monitorar dados fisiológicos de seus astronautas em órbita. Uma das formas de telemedicina usadas hoje é a transmissão de sinais eletrocardiográficos para centros especializados, permitindo que o paciente seja diagnosticado sem nem mesmo precisar sair de casa.

O ECG digital não tem aplicação apenas na telemedicina. A possibilidade de armazenar sinais de ECG de forma digital possibilita que as clínicas criem um banco de dados com o histórico do paciente, diferenciando seus serviços. Existe um tipo de exame eletrocardiográfico, o ECG Holter, onde o paciente é monitorado durante 24 horas, através de um aparelho portátil que amplifica e grava o sinal de ECG. Esse aparelho já começa a ser digitalizado, o que torna o sinal mais adequado para alguns tipos de processamento e aplicações.

Para que o sinal de ECG digital seja armazenado/transmitido com uma qualidade aceitável, é gerada uma grande quantidade de dados. Por exemplo, para uma freqüência de amostragem de 200 Hz, 3 canais simultâneos e 12 bits por amostra, são gerados aproxima-

damente 80 Mbytes, em um período de 24 horas.

Como a quantidade de dados é muito grande e a memória física disponível em aparelhos portáteis é normalmente pequena torna-se necessário fazer a compressão do sinal de eletrocardiograma, a fim de representá-lo com o menor número de bits possível. Essa compressão deve ser tal que o sinal decodificado, ainda que distorcido, não perca as características relevantes para um diagnóstico correto.

A idéia de comprimir dados já existia no século XIX, quando Samuel Morse desenvolveu o código Morse para transmissão de texto via telégrafo. Ele percebeu que as letras ocorrem com freqüências diferentes e atribuiu códigos maiores para as letras que têm menos probabilidade de ocorrer e códigos menores para as letras mais prováveis, reduzindo o tempo necessário para enviar uma mensagem.

Os métodos de compressão de sinais de ECG descritos na literatura englobam os métodos diretos e os métodos por transformação. Os métodos diretos normalmente usam técnicas de predição e interpolação a fim de reduzir a redundância do sinal. Os métodos por transformação aplicam uma transformação ortogonal linear ao sinal, que diminui a correlação entre suas amostras e consequentemente, reduz a taxa necessária para codificá-lo. Segundo [1], os métodos de compressão diretos são mais eficientes em relação à velocidade de processamento e à taxa de compressão.

Este trabalho propõe que o sinal de ECG seja comprimido utilizando um algoritmo conhecido como MMP (Multidimensional Multiscale Parser), que se baseia na recorrência de padrões multiescalas e não se encaixa em nenhum dos métodos explicados acima. Enquanto codifica o sinal de entrada, o MMP constrói um dicionário que é atualizado com concatenações de versões contraídas/dilatadas de padrões previamente codificados.

Dessa forma, o dicionário armazena os padrões típicos do sinal, que são utilizados para representá-lo. Podemos concluir que o MMP será mais eficiente quando o sinal que estiver sendo codificado tiver padrões que se repetem, isto é, quando o sinal tiver redundâncias. Sinais periódicos são bastante redundantes já que a cada período existe a ocorrência do mesmo padrão. O sinal de eletrocardiograma (Figura 1.1) tem uma característica quase periódica e portanto, podemos utilizar o MMP com sucesso para comprimir este tipo de sinal.

Este projeto é baseado no trabalho de [2], que aplicou o MMP para sinais de ECG e introduziu modificações que permitissem explorar melhor a característica quase periódica

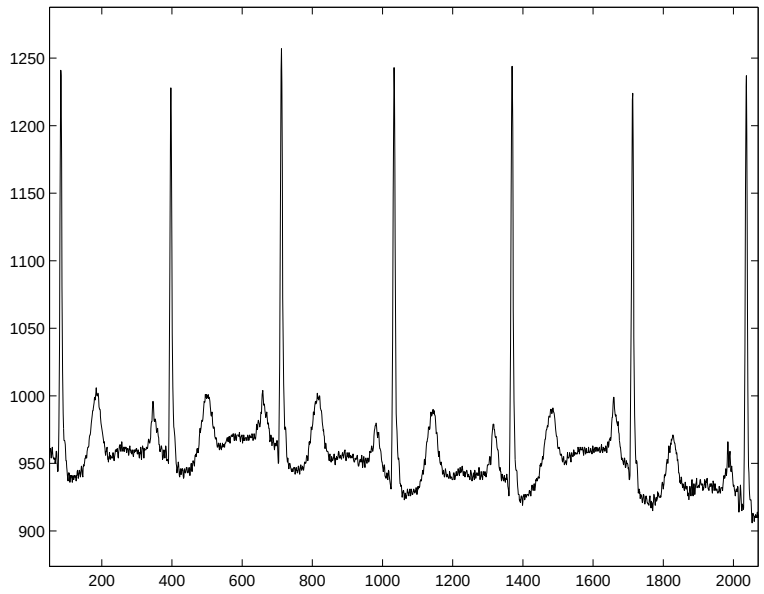


Figura 1.1: Sinal de ECG

do sinal. Apesar dos bons resultados, percebemos que o MMP ainda poderia se tornar mais eficiente para sinais de ECG e implementamos alterações que melhoraram os resultados obtidos por [2].

Apresentaremos no capitulo 2 uma breve introdução à eletrocardiografia para que o leitor familiarize-se com o tipo de sinal com que trabalharemos. No capítulo 3 será apresentada uma descrição detalhada do algoritmo MMP simples. No capítulo 4 apresentaremos o algoritmo MMP com otimização taxa-distorção, implementada primeiramente por [3]. Em seguida, será introduzido o Dicionário de Deslocamentos que foi aplicado primeiramente no trabalho de [2]. O capítulo seguinte detalha as alterações feitas neste projeto ao algoritmo MMP a fim de melhorar o seu desempenho na compressão de sinais eletrocardiográficos. Os resultados serão apresentados à medida que apresentarmos as alterações implementadas.

Capítulo 2

Eletrocardiografia

Neste capítulo faremos uma breve descrição da fisiologia do coração a fim de que possamos compreender como os sinais elétricos captados pelo eletrocardiograma são gerados. Em seguida explicaremos as principais características do ECG e os tipos de exames mais comuns. Serão apresentados também os critérios de qualidade para os sistemas de compressão de sinais de ECG, e por fim o leitor ficará familiarizado com o banco de dados de sinais de eletrocardiograma que foi utilizado.

2.1 Fisiologia do Coração

O coração é um órgão essencialmente muscular, localizado na caixa torácica e revestido por uma membrana denominada pericárdio. O coração apresenta quatro cavidades: duas superiores denominadas átrios e duas inferiores denominadas ventrículos. O átrio direito comunica-se com o ventrículo direito através da válvula tricúspide e o átrio esquerdo comunica-se com o ventrículo esquerdo através da válvula bicúspide ou mitral.

A função do coração é bombear o sangue oxigenado de forma que ele irrigue todos os tecidos e volte para o pulmão para ser oxigenado novamente. Para isso, as câmaras cardíacas contraem-se e dilatam-se alternadamente. O processo de contração de cada câmara do músculo cardíaco (miocárdio) denomina-se sístole, enquanto que o processo de dilatação que ocorre entre cada sístole chama-se diástole.

A frequência cardíaca é controlada por uma região do coração chamada nódulo sinusal, marcapasso ou nó sino-atrial, localizada entre o átrio direito e a veia cava superior e constituída por um conjunto de células musculares especializadas. Os impulsos originados no

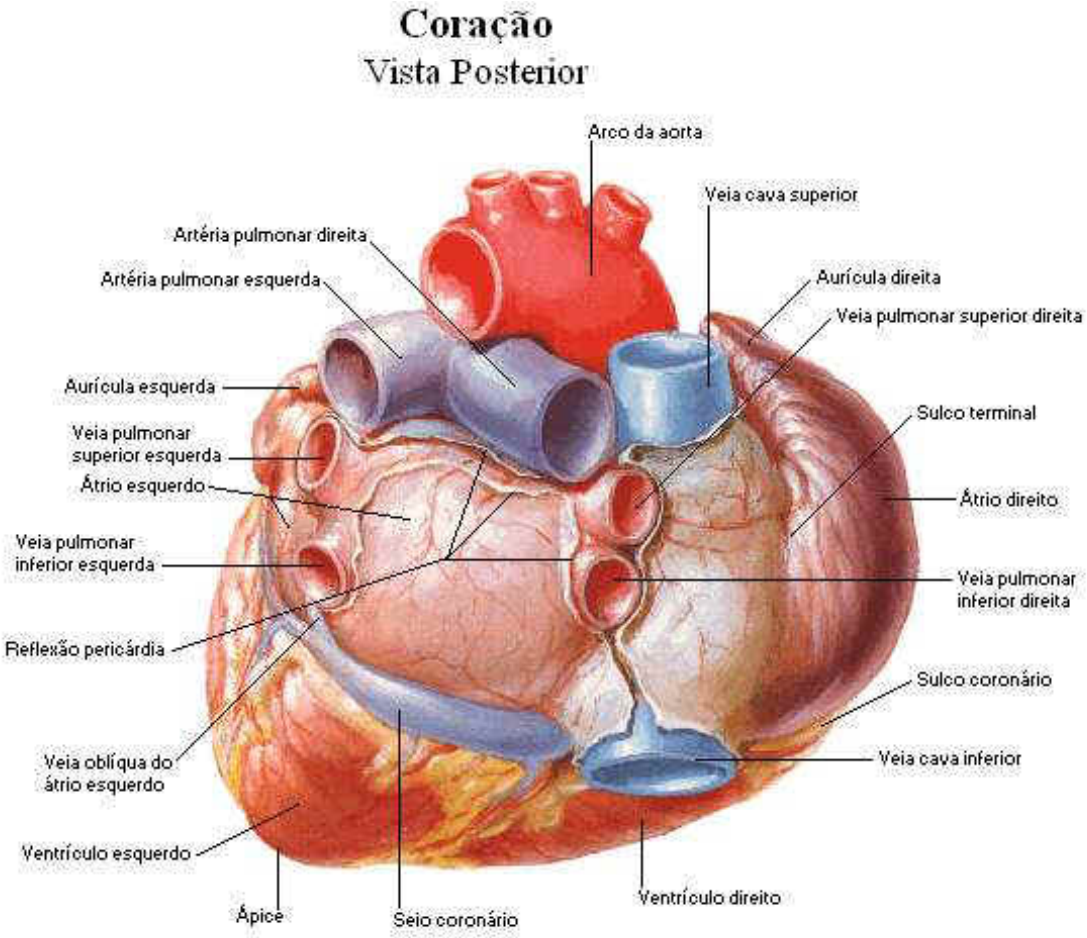


Figura 2.1: Fisiologia do Coração.

nódulo sinoatrial são transmitidos para os átrios e ventrículos. A passagem deste estímulo causa a despolarização das células e provoca a contração atrial e, em seguida, a ventricular.

As ondas de despolarização e repolarização das cavidades do coração geram tensões elétricas que podem ser medidas na superfície do corpo. Essas tensões elétricas são medidas através do exame eletrocardiográfico.

2.2 Eletrocardiograma

O eletrocardiograma é um exame não invasivo e de baixo custo que permite ao médico diagnosticar distúrbios cardíacos em seus pacientes através da análise de alterações do padrão elétrico do coração.

O registro da variação dos fenômenos elétricos do coração é obtido através de um galvanômetro, aparelho que mede a diferença de potencial entre dois pontos, a partir da corrente elétrica em dois eletrodos dispostos em determinados pontos do corpo humano. O primeiro galvanômetro que registrou as tensões elétricas produzidas na superfície do corpo pela atividade elétrica do coração foi construído no final do século XIX, por um físico holandês, William Einthoven, que recebeu o prêmio Nobel de Medicina, em 1924, por sua invenção.

A cada batimento cardíaco, o ECG registra um sinal que consiste de três complexos principais: a onda P, o complexo QRS e a onda T, como mostra a Figura 2.2, que são ondas de despolarização e repolarização. A onda P corresponde à despolarização atrial e ocorre imediatamente antes do início da contração. Já o complexo QRS corresponde à despolarização ventricular e ocorre imediatamente antes da contração ventricular. A onda T, que ocorre após o complexo QRS, corresponde à repolarização ventricular.

Cada medida de potencial elétrico entre dois pontos é chamada de derivação eletrocardiográfica. Essas derivações são definidas de acordo com a posição dos eletrodos. Nas derivações bipolares os dois eletrodos são dispostos equidistantes do coração, em pontos de potencial flutuante. Nas derivações unipolares, um dos eletrodos, chamado eletrodo indiferente, é colocado num ponto de potencial nulo, enquanto que o outro eletrodo, chamado eletrodo explorador, é posicionado num ponto de potencial flutuante.

O primeiro sistema de derivação, introduzido por Einthoven e empregado até hoje, é o triângulo de Einthoven, que mede três derivações, como mostra a Figura 2.3. Outras derivações bastante utilizadas nos exames de ECG são as derivações unipolares do plano

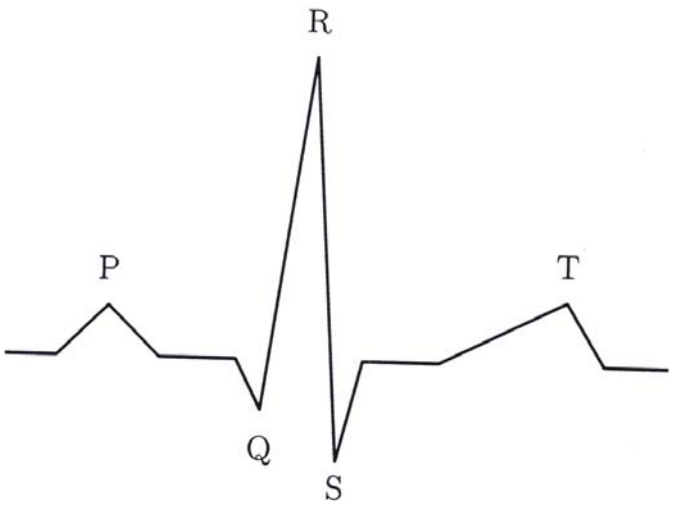


Figura 2.2: Complexos principais de um sinal de ECG

frontal (derivações aumentadas) e as derivações unipolares no plano transverso (derivações precordiais). As derivações aumentadas medem a diferença de potencial entre um eletrodo localizado em um dos membros e a média dos sinais dos outros dois eletrodos. As derivações precordiais registram a diferença de potencial entre os eletrodos posicionados sequencialmente no peito e uma referência virtual que é a média das três derivações principais. Utilizam-se seis derivações precordiais. Um exame de ECG normalmente envolve o registro de doze derivações, tomadas simultaneamente ou três a três.

Apresentaremos alguns tipos de exames de eletrocardiograma:

- Eletrocardiografia Convencional

  O eletrocardiograma convencional é realizado em consultórios e clínicas médicas. O exame tem a duração de 10 a 30 minutos, e registra as 12 derivações do ECG. Normalmente as derivações são tomadas uma a uma ou de três em três e o sinal de ECG é registrado em papel termosensível. Atualmente já existem eletrocardiógrafos digitais que permitem que as doze derivações sejam tomadas simultaneamente e armazenadas em formato digital.

- Eletrocardiografia Ambulatorial - Sistema Holter

  O sistema Holter permite a gravação do eletrocardiograma por períodos de 24 horas, com os pacientes desempenhando suas atividades habituais. O gravador, de pequenas dimensões, é conectado ao tórax do paciente por cabos e eletrodos especiais. O grava-

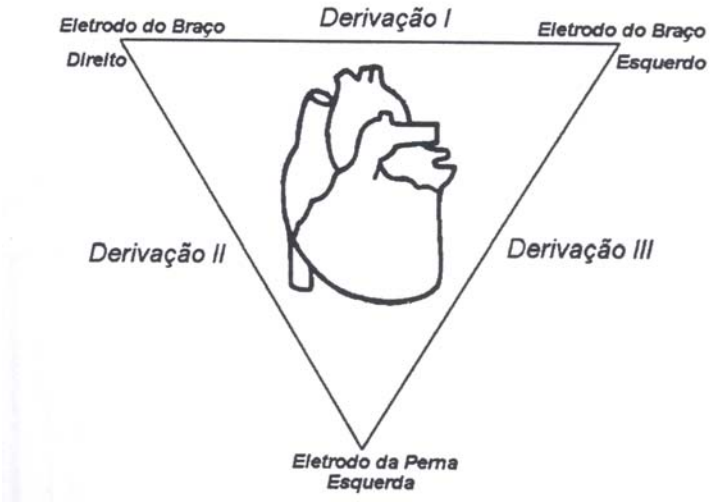


Figura 2.3: Triângulo Einthoven

dor pode ser analógico ou digital. Nos sistemas analógicos os dados são gravados em fita-cassete e posteriormente, são digitalizados e copiados para um computador. Nos sistemas digitais, o ECG já é armazenado em formato digital.

- ECG de Alta Resolução

  Segundo [4], o eletrocardiograma de alta resolução é um método computadorizado, não-invasivo, relativamente barato e de fácil aplicação clínica, que serve para identificar pacientes que apresentam alto risco de eventos cardíacos (taquicardia ventricular sustentada ou morte súbita), por meio da detecção dos potenciais tardios.

  Neste tipo de eletrocardiograma tenta-se evidenciar potenciais de baixa amplitude que podem ocorrer no final do complexo QRS, os potenciais tardios. Para isso, é necessária a eliminação dos ruídos gerados durante a gravação do sinal de ECG, que é feito através da aplicação de técnicas de processamento de sinais no ECG digitalizado. Dessa forma obtém-se um sinal de ECG sem ruídos e que pode ser ampliado em até 100 vezes.

- Telemedicina

  A Telemedicina é o emprego das tecnologias de informação e comunicações para prestar assistência médica a pacientes em locais distantes, através da transferência de informações médicas utilizando as redes de telecomunicações. Esses dados são recebidos e tratados em centrais especializadas.

  Uma das aplicações da Telemedicina é a telemetria de ECG, onde as derivações de ECG

do paciente são transmitidas para centros especializados, permitindo que o paciente seja diagnosticado sem precisar estar em contato direto com o médico.

2.3 Medidas de Qualidade da Compressão de Sinais de ECG

Os algoritmos de compressão de sinais reduzem o tamanho do sinal pois encontram uma maneira mais eficiente de representá-lo. Quando uma determinada aplicação exige que o sinal seja comprimido, é necessário que tenhamos como avaliar a qualidade do algoritmo que estamos utilizando para reduzir o tamanho do sinal.

Os algoritmos de compressão (com perdas e sem perdas) normalmente são avaliados quanto à taxa de compressão, isto é, a razão entre o número de bits do sinal original e o número de bits do sinal codificado e quanto à taxa de bits - o número de bits/amostra do sinal codificado. Nos sistemas de compressão com perdas, o sinal decodificado não é exatamente igual ao sinal original e portanto, uma medida da distorção, isto é, da diferença entre esses dois sinais, é importante para medir a qualidade da compressão.

Embora exista a preocupação com o diagnóstico do exame de eletrocardiograma quando utiliza-se o sinal reconstruído, não será proposta aqui nenhuma forma de avaliar até quando se pode comprimir sem que o diagnóstico seja comprometido. Esse tipo de avaliação é bastante subjetiva e iria requerer a colaboração de médicos, o que foge ao escopo deste trabalho. No entanto, uma medida como essa seria interessante e fica como proposta para futuros trabalhos em compressão de sinais de ECG.

Os sistemas de compressão de sinais de ECG utilizam como medidas de qualidade a taxa de compressão CR, a taxa de bits BR e o *percent root mean square difference* - PRD, uma métrica para a distorção.

A taxa de compressão é definida como:

$$CR = \frac{\text{Número de Bits do Sinal Original}}{\text{Número de Bits do Sinal Comprimido}} \tag{2.1}$$

A taxa de bits por amostra pode ser calculada segundo a equação abaixo:

$$BR = \frac{\text{Número de Bits do Sinal Comprimido}}{\text{Número de Amostras do Sinal Original}} \tag{2.2}$$

Para um canal c, o PRD é definido como:

$$PRD_c(\mu) = \sqrt{\frac{\sum_{i=1}^{n}(x_c[i] - \hat{x_c}[i])^2}{\sum_{i=1}^{n}(x_c[i] - \mu)^2}} \cdot 100\%, \tag{2.3}$$

onde x_c é o sinal original e $\hat{x_c}$ é o sinal reconstruído, ambos com tamanho de n amostras.

A constante μ pode assumir diversos valores. Alguns definem μ como sendo a média do sinal, enquanto outros definem μ como sendo zero. Nos cálculos que faremos de PRD, usaremos $\mu = 1024$, como fizeram [2] e [5], valor que corresponde à tensão nula dos sinais, após conversão AD de 11 bits.

Para sinais com vários canais, o PRD é definido como:

$$PRD^{(m)}(\mu) = \sqrt{\frac{\sum_{j=1}^{m}\sum_{i=1}^{n}(x_j[i] - \hat{x_j}[i])^2}{\sum_{j=1}^{m}\sum_{i=1}^{n}(x_j[i] - \mu)^2}} \cdot 100\%, \tag{2.4}$$

onde m é o número de canais, n é o número de amostras de cada canal, x_j é o sinal original e $\hat{x_j}$ é o sinal reconstruído.

Neste trabalho utilizaremos, para cáculo do PRD, a Equação 2.3

2.4 Banco de Dados - MIT-BIH

Os sinais de eletrocardiograma usados neste projeto para testar o algoritmo proposto fazem parte de um banco de dados criado pelo MIT e pelo Beth Israel Deaconess Medical Center - MIT-BIH Arrhythmia Database. Este banco de sinais de ECG é um padrão internacional para testar processadores de sinais de eletrocardiograma e também para estudo da dinâmica cardíaca.

O MIT-BIH Arrhythmia Database começou a ser distribuído em fitas analógicas em 1980, e hoje está disponível em CD-ROM e também pela Web (http://www.physionet.org/physiobank/database/mitdb/).

Os sinais do mitdb utilizados neste trabalho foram gravados no formato 16, onde cada amostra é representada por um valor na forma de complementado de 2 de 16 bits. O byte

mais significativo é armazenado depois do byte menos significativo e as amostras de cada canal são intercaladas, isto é, primeiro é gravada a amostra 1 do canal 1, em seguida, a amostra 1 do canal 2 e assim sucessivamente.

As informações sobre os sinais, relevantes para fazer qualquer tipo de processamento devem ser extraídas do arquivo de cabeçalho (extensão .hea). Os três primeiros campos da primeira linha do arquivo são: nome da gravação, número de canais (número de sinais) e frequência de amostragem. Nos quatro primeiros campos da segunda linha podemos obter o nome do arquivo de dados, o formato de gravação, o ganho e a resolução do conversor analógico-digital usado para digitalizar o sinal de ECG, respectivamente.

No apêndice B mostraremos os sinais que foram utilizados para os testes do algoritmo.

Capítulo 3

MMP

Neste capítulo será descrito o algoritmo MMP (Multidimensional Multiscale Parser), utilizado para compressão de dados multidimensionais. O MMP é baseado na recorrência de padrões multiescalas, ele constrói um dicionário que é atualizado com concatenações de versões contraídas/dilatadas de padrões previamente codificados. Dessa forma o MMP vai aprendendo os padrões típicos do sinal, e à medida que os padrões dos dados que estão sendo codificados se repetirem, existirá um elemento no dicionário para representar este padrão.

O MMP pode ser utilizado quando deseja-se obter altas taxas de compressão, isto é, quando é permitido distorcer o sinal original, o que caracteriza uma distorção com perdas. No entanto, se o sinal que será comprimido não puder sofrer nenhuma distorção, o MMP mesmo assim poderá ser utilizado para realizar uma compressão sem perdas, porém a uma taxa de compressão menor.

3.1 Descrição do MMP

Para codificar um segmento de um sinal $\mathbf{X}^j$ de tamanho $\mathbf{N}$, o MMP busca no dicionário $\boldsymbol{D}$ o elemento $\mathbf{S}_i$ que mais se aproxima de $\mathbf{X}^j$, isto é, aquele que minimiza a distorção $\| \mathbf{X^j - S_i} \|$. A seguir verifica-se se essa distorção é menor do que a distorção-alvo d* desejada (distorção/amostra).

Caso seja menor, o procedimento termina e o segmento $\mathbf{X}^j$ será representado pelo elemento $\mathbf{S}_i$ do dicionário $\boldsymbol{D}$ escolhido pelo algoritmo. Caso contrário, $\mathbf{X}^j$ será dividido em duas partes $\mathbf{X}^{2j+1}$ e $\mathbf{X}^{2j+2}$, de tamanho $\frac{\mathbf{N}}{\mathbf{2}}$, e cada uma delas será processada pelo MMP. Esse processo de segmentação continuará até que a representação do bloco processado satisfaça o

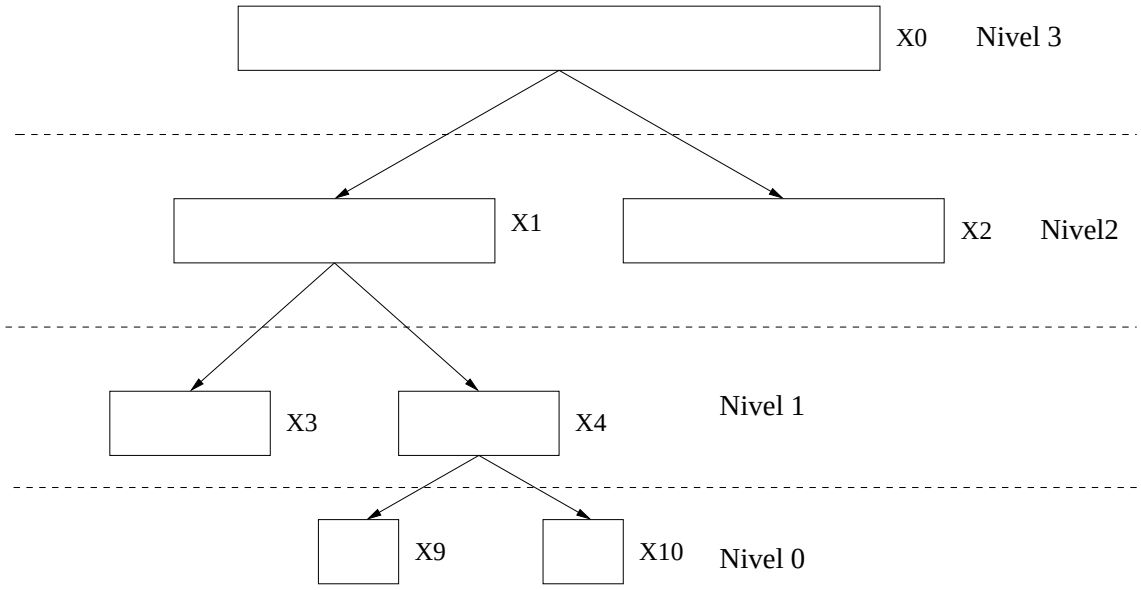


Figura 3.1: Árvore de segmentação.

critério da distorção-alvo.

Quando $\hat{\mathbf{X}}^{2j+1}$ e $\hat{\mathbf{X}}^{2j+2}$ estiverem disponíveis, $\mathbf{X}^j$ será representado pela concatenação destes dois blocos, isto é, $\mathbf{X}^j = [\hat{\mathbf{X}}^{2j+1}\ \hat{\mathbf{X}}^{2j+2}]$ e este novo bloco será incluído no dicionário $\boldsymbol{D}$.

Consideremos que um segmento de 8 amostras será codificado utilizando a algoritmo descrito acima e que para uma distorção-alvo d*, foi gerada a árvore de segmentação da Figura 3.1. Podemos ver que a cada nível da árvore de segmentação corresponde um tamanho do segmento que está sendo codificado, de forma que em cada nível o MMP estará processando um segmento com uma escala diferente.

A saída do MMP, isto é, os dados que serão armazenados/transmitidos, é uma seqüência de flags, que indica se houve segmentação do bloco sendo codificado, e índices, que indicam os elementos do dicionário que foram utilizados para representar os blocos $\mathbf{X}^j$. Quando existe o casamento de um bloco $\mathbf{X}^j$ por um padrão existente no dicionário, o flag 1 é codificado. Em seguida é codificado o índice do dicionário referente a este elemento. Quando o bloco é segmentado, codifica-se o flag 0. A seqüência de flags e índices gerada no exemplo anterior seria 0 0 1 i3 0 1 i9 1 i10 1 i2. Esses símbolos (flags e índices) são codificados utilizando o codificador aritmético.

Apresentaremos agora um pseudo-código do algoritmo:

Primeiro considere que temos um vetor de dicíonários $\boldsymbol{D}$, com L coordenadas, uma para cada escala, e que cada dicionário possui um conjunto de elementos $S_1^l, S_2^l, S_3^l, ..., S_K^l$. Seja um vetor de entrada X^j de tamanho $l(X^j)$, uma função $T(X^j, l(X^j), L)$ de trans-

formação de escala, que transforma o tamanho $l(X^j)$ de X^j para L, e uma distorção-alvo d*.

```
X̂^j = MMPcod(X^j, l(X^j), d*){
      S^l = D[l(X^j)]
      k = tamanhoDicionario(S^l)
      For i = 1:k{
            distorcao[i] =|| X^j − S_i^l ||
      }
      indice = min(distorção)
      If (distorcao[indice]^2) ⩽ (l(X^j).d*){
            Codifica(flag1)
            Codifica(índice)
            X̂^j = S_indice
      }
      Else{
             Codifica(Flag0)
            X̂^(2j+1) = MMPcod(X^(2j+1), l(X^j)/2, d*)
            X̂^(2j+2) = MMPcod(X^(2j+2), l(X^j)/2, d*)
            X̂^j = [X̂^(2j+1) X̂^(2j+2)]
            For i = 1:  L{
                  D(i) = T(X̂^j, l(X̂^j), L)
            }
      }
}
```

O processo de decodificação do MMP também é bastante simples e segue a idéia do codificador. O decodificador conhece a árvore de segmentação, e a partir dela, utilizando um procedimento recursivo, reconstrói o segmento que foi codificado. O dicionário é inicializado da mesma forma que no codificador e é atualizado com os segmentos que foram decodificados

e que não existiam anteriormente. Dessa forma, o dicionário criado no decodificador é uma réplica do dicionário criado pelo codificador, o que permite que o decodificador recupere o sinal através dos índices do dicionário enviados no processo de codificação. Vamos relembrar o exemplo que foi dado, onde foi gerada pelo codificador a árvore de segmentação da Figura 3.1. A sequência de saída gerada e que será recebida pelo decodificador é: 0 0 1 i3 0 1 i9 1 i10 1 i2. A partir desta sequência o decodificador irá reconstruir X0. O MMP então segue o seguinte procedimento:

```
X̂^j = MMPdec(l(X^j)){
      S^l = D[l(X^j)]
      flag = Decodifica(flag)
      If (flag == 1){
            Decodifica(indice)
            X̂^j = S^l_indice
      }
      Else{
            X̂^{2j+1} = MMPdec(l(X^j)/2)
            X̂^{2j+2} = MMPdec(l(X^j)/2)
            X̂^j = [X̂^{2j+1} X̂^{2j+2}]
            For (i = 0 :  L){
                  D(i) = T(X̂^j, l(X̂^j), L)
            }
      }
}
```

Apresentarenos agora o pseudo-código da função de transformação de escala.

```
X^j = transFormaçãoEscala(X0^j, l(X0^j), L ){
    aux = l(X0^j) - 1
    if (aux ≥ 0)&&(aux < L){
        for(n = 0 ; n < l(X0^j) ; n++){
            m = (aux*n)/L
            if (m+1 ≤ aux)
                aux1 = X0^j[m+1]
            else
                aux1 = X0^j[m]
            es1 = ((aux1−aux2)*((aux*n)%(L)))/L + aux2
            X^j[n] = es1
        }
    }
    if (aux ≥ L){
        for(n = 0 ; n < l(X0^j) ; n++){
            es1 = 0
            for (k = −aux/2 ; k ≤ aux/2 ; k++){
                m = (aux*n+k)/L
                if (m < 0)
                    m = -m
                if (m+1 ≤ aux)
                    aux1 = X0^j[m+1]
                aux2 = X0^j[m]
                es1 = (es1+((aux1−aux2)+((aux*n+k)%(L)))+aux2)/L
            }
            es1 = es1/aux
            X^j[n] = es1
        }
    }
}
```

Podemos ver na Figura 3.2 um exemplo do MMP aplicado ao sinal de ECG. Seguindo o

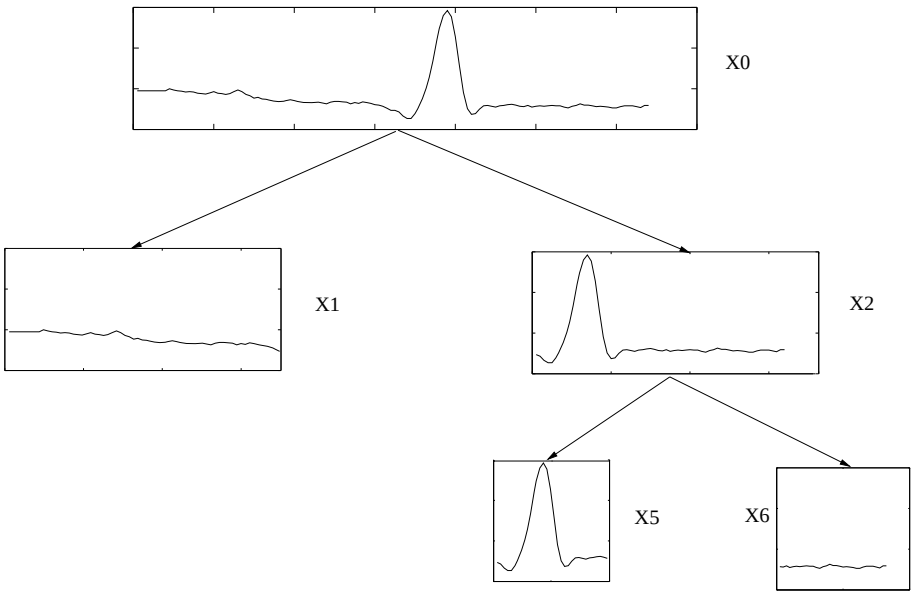


Figura 3.2: Exemplo - ECG.

procedimento explicado anteriormente, não foi encontrado nenhum elemento para representar o sinal X0 e ele foi dividido em 2 segmentos de mesmo tamanho. A primeira parte X1 não foi dividida porque algum elemento do dicionário satisfez o critério de distorção-alvo, enquanto que a segunda parte X2 não satisfez o critério e foi dividida também. Em seguida, X5 e X6 foram processados e ambos foram representados por um elemento do dicionário. A concatenação de $\hat{X}5$ e $\hat{X}6$ representada na Figura 3.3 é inserida no dicionário. Agora que temos $\hat{X}2$, $\hat{X}0$ será formado pela concatenação de $\hat{X}1$ com $\hat{X}2$, e também será adicionado ao dicionário.

3.2 Detalhes de Implementação

A fim de reduzir a complexidade computacional, o sinal que deseja-se codificar é particionado em blocos cujo tamanho $\mathbf{M}$ é uma potência de 2 e cada bloco é processado sequencialmente.

Como foi mostrado anteriormente, o bloco que está sendo codificado pode ser dividido até que se encontre uma boa representação para ele, de forma que o MMP irá processar blocos de tamanhos variados. O número de possíveis escalas (número de níveis da árvore de segmentação) é dependente do tamanho $\mathbf{M}$ de cada bloco e é igual a $\log_2(M) + 1$.

Assim, é necessário criar um dicionário para cada escala possível, de forma que quando um segmento $\mathbf{X}^j$ de tamanho N estiver sendo codificado, o dicionário de escala N será aces-

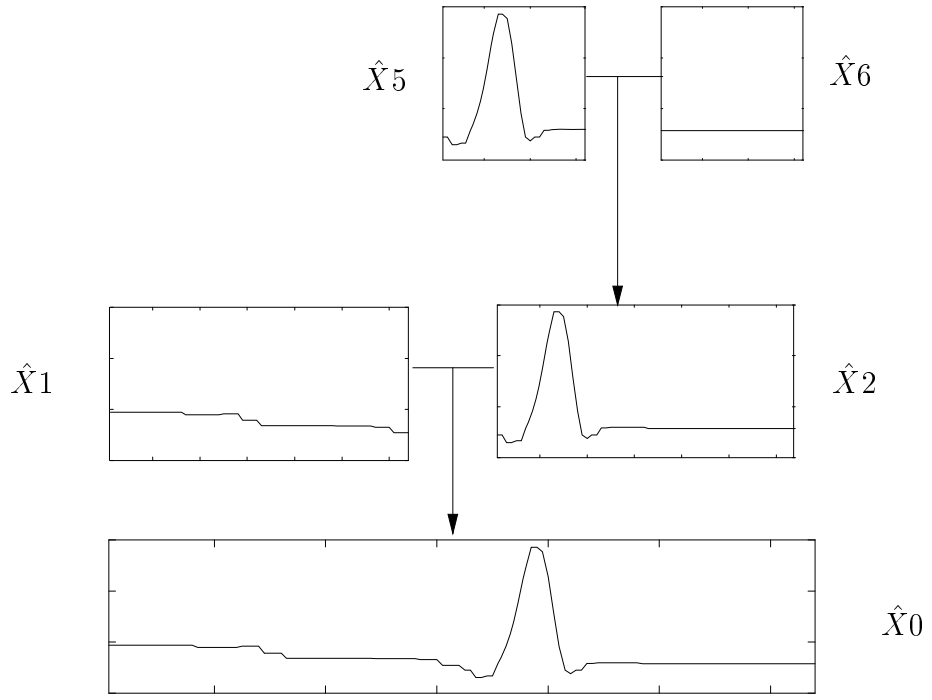


Figura 3.3:

sado para a busca da melhor representação de $\mathbf{X}^j$. Cada vez que um bloco de tamanho N é formado, ele é incluído nos dicionários de todas as escalas. Para que um vetor de tamanho N0 seja convertido para um vetor de tamanho N, é aplicada a função de transformação de escala. Quando a escala de um vetor é reduzida, é possível que o vetor resultante já exista no dicionário, sendo necessário verificar se este novo vetor já é um elemento do dicionário, para que não existam elementos iguais, o que iria reduzir a eficiência da codificação, pois seriam gastos bits com elementos repetidos sem necessidade. Devido à restrições de memória, os dicionários são limitados a um tamanho máximo. Como em [3], utilizamos o limite de 32.768 elementos.

Os dicionários são inicializados de forma a permitir que todo o sinal que será codificado seja representado utilizando apenas os elementos do dicionário inicial. Supondo um dicionário inicial de comprimento 1, para que o sinal possa ser comprimido sem perdas, o dicionário deve conter todos os possíveis valores das amostras do sinal. Vamos chamar de Xmin e Xmax o menor e o maior valores que as amostras do sinal podem assumir, respectivamente. O dicionário deve ser inicializado com valores no intervalo [Xmin Xmax], com passo igual a 1. Caso a compressão seja com perdas, o dicionário inicial não precisa ser tão rico, o que corresponde à etapa de quantização dos sistemas de compressão tradicionais. O passo δ, como utilizado por [3], depende da distorção-alvo d* e é igual a $\frac{d*}{2}$. Dessa forma, o

número de elementos do dicionário será $\frac{(Xmax-Xmin)}{\delta}$. Os dicionários de todas as escalas são inicializados da mesma forma.

Para que o codificador aritmético (ver seção 3.4) seja eficiente, é necessário que exista um modelo estatístico diferente para flags e para índices, para cada escala do dicionário. Por exemplo, em escalas menores, como por exemplo, blocos 2x1, a ocorrência do flag que indica aproximação será bem maior que em escalas superiores. A freqüência relativa de ocorrência dos possíveis índices também será diferente para cada escala.

3.3 Funções Principais

Serão descritas abaixo as funções principais do MMP que foram implementadas. O algoritmo foi implementado em C++, com excessão das funções do codificador aritmético, que foram implementadas em C.

- **indice = buscaDicionario(X^j, $l(X^j)$)**

  Descrição: Esta função faz uma busca no dicionário por um elemento para representar o segmento X^j, de tamanho $l(X^j)$.

  1 - Acessa o dicionário na escala $l(X^j)$

  2 - Para cada elemento deste dicionário, calcula-se $\| \mathbf{X^j - S_i} \|$

  3 - O elemento que fornece a menor distorção é escolhido para representar X^j.

  4 - Retorna-se o índice deste elemento.

- **atualizaDicionario(X^j, $l(X^j)$, L)**

  Descrição: Esta função é utilizada para atualizar o dicionário na escala L quando um novo segmento é formado.

  1 - X^j = transformaçãoEscala(X_0^j, $l(X_0^j)$, L)

  2 - Acessa o dicionário na escala $l(X^j)$.

  3 - Verifica-se se X^j já existe no dicionário. Caso não exista e o dicionário não esteja com tamanho máximo, vai para o passo 4. Caso contrário, a função finaliza.

  4 - O elemento X^j é inserido no dicionário.

  5 - O modelo dos índices é expandido.

- $\hat{X^j}$ = **codifica(**X^j**,** $l(X^j)$**, d*)**

  Descrição: Esta função gera e codifica a árvore de segmentação para um bloco X^j, de tamanho $l(X^j)$, para uma distorção-alvo d*.

  1 - indice = buscaDicionario(X^j, $l(X^j)$, d*)

  2 - Verifica-se se a distorção encontrada satisfaz o critério da distorção-alvo, isto é, se $\| \mathbf{X^j} - \mathbf{S_i} \|^2 \leqslant (l(X^j).d*)$. Caso seja menor vai para o passo 3. Caso contrário, vai para o passo 7.

  3 - Codifica-se o flag 1.

  4 - Codifica-se o índice.

  5 - $\hat{X^j} = S_{indice}$.

  6 - Retorna $\hat{X^j}$.

  7 - Codifica-se o flag 0.

  8 - Divide-se o bloco em X^{2j+1} e X^{2j+2} e chama-se a função para os 2 blocos:

  $$X^{2j+1} = \text{codifica}(X^{2j+1}, \frac{l(X^j)}{2}, \text{d*})$$

  $$X^{2j+2} = \text{codifica}(X^{2j+2}, \frac{l(X^j)}{2}, \text{d*})$$

  9 - $\hat{X^j} = [X^{2j+1}\ X^{2j+2}]$

  10 - Para todas as escalas possíveis:

  atualizaDicionario($\hat{X^j}$, l($\hat{X^j}$), escala)

- $\hat{X^j}$ = **decodifica(**$l(X^j)$**)**

  Descrição: Esta função decodifica os flags e índices e retorna $\hat{X^j}$.

  1 - Decodifica o flag. Caso o flag seja 1 vai para o passo 2. Caso o flag seja 0, vai para o passo 5.

  2 - Decodifica o índice.

  3 - Acessa o dicionário na escala $l(X^j)$.

  4 - retorna $\hat{X^j} = S_{indice}$

  5 - Divide-se o bloco em X^{2j+1} e X^{2j+2} e chama-se a função para os 2 blocos:

  $$\hat{X}^{2j+1} = \text{decodifica } (\frac{l(X^j)}{2})$$

$\hat{X}^{2j+2}$ = decodifica $(\frac{l(X^j)}{2})$

6 - $\hat{X}^j = [\hat{X}^{2j+1}\hat{X}^{2j+2}]$

7 - Para todas as escalas:

atualizaDicionario($\hat{X}^j$, l($\hat{X}^j$), L)

3.4 Codificação da Sequência de Saída

A saída do MMP é uma sequência de flags e índices, que queremos representar com a menor quantidade de bits possível. Para fazer essa codificação utilizamos um codificador aritmético.

O codificador aritmético transforma a sequência de símbolos que está sendo codificada em um intervalo entre 0 e 1. Como existem infinitos números neste intervalo, é possível associar um intervalo diferente para cada sequência de símbolos diferente. Para que seja feito o mapeamento dessa sequência de símbolos em um intervalo [0 1], usa-se a função de distribuição cumulativa (cdf). Cada possível símbolo s_i, com uma probabilidade diferente de zero, terá um valor distinto de $Fx(s_i)$. Devemos lembrar que o valor máximo da cdf é 1.

O intervalo que será transmitido vai diminuindo à medida que novos símbolos são recebidos. À medida que a mensagem se torna maior, o intervalo que a representa torna-se menor, o que aumenta o número de bits necessário para codificar o intervalo. O intervalo relativo à cada símbolo está associado à sua probabilidade de ocorrência. Símbolos com alta probabilidade de ocorrência estarão associados à subintervalos pequenos (menos bits), enquanto que símbolos com baixa probabilidade de ocorrência estarão associados à subintervalos maiores (mais bits).

O codificador aritmético é um codificador de entropia, isto é, o número de bits usado para representar um determinado símbolo aproxima-se da entropia da fonte. Shannon mostrou que é impossível codificar uma mensagem gerada aleatoriamente usando menos bits que a entropia da fonte.

3.4.1 Gerando o Intervalo

Consideremos que uma fonte gere os símbolos s1, s2, s3, s4 e s5, com uma probabilidade $P(s_i)$ associadas a cada um deles, como mostra a Tabela 3.1, e que deseja-se codificar a mensagem formada pelos símbolos s2 s1 s3 s3 s6, gerados nesta ordem.

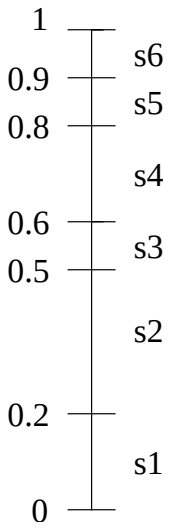


Figura 3.4: Intervalo Inicial

Tabela 3.1: Exemplo - Codificador Aritmético

Símbolo	P(símbolo)	Fx(símbolo)
s1	0.2	0.2
s2	0.3	0.5
s3	0.1	0.6
s4	0.2	0.8
s5	0.1	0.9
s6	0.1	1.0

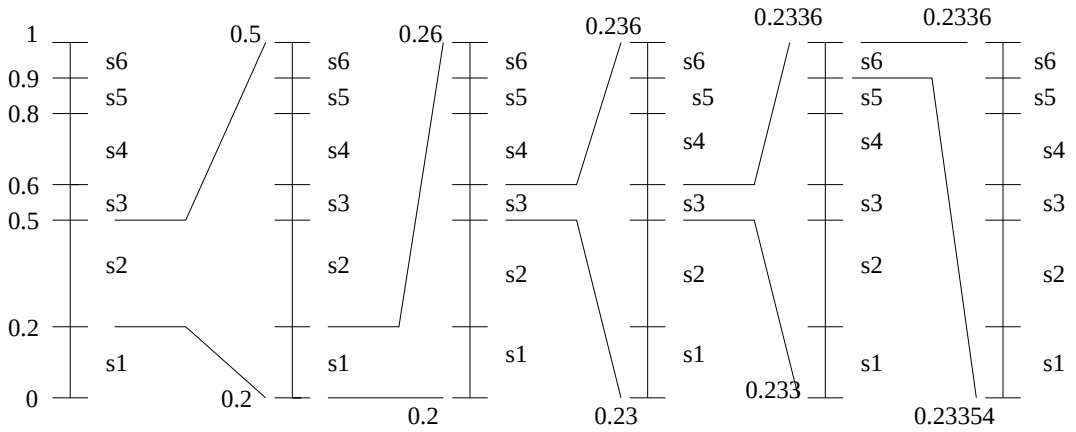


Figura 3.5: Exemplo - Codificador Aritmético

Podemos ver pela Figura 3.4 que antes que se inicie a transmissão, o intervalo [0 1] é particionado de acordo com a probabilidade de ocorrência (e de acordo com a Fx) do símbolo. À medida que cada símbolo é processado, os limites do intervalo são escalados de acordo com a probabilidade de ocorrência do símbolo e divididos na mesma proporção que o intervalo inicial. Podemos ver este processo ocorrendo na Figura 3.5

Para implementação do codificador aritmético utilizamos como base a implementação de [6]. Foi necessário fazer modificações para que pudéssemos trabalhar com diversos modelos e também para que os tipos das variáveis fossem condizentes com os dados com que estávamos trabalhando.

Utilizamos um modelo adaptativo, onde a probabilidade dos símbolos é atualizada a cada ocorrência do símbolo. No caso dos flags, o número de caracteres não varia ao longo do processamento, é sempre igual a 2. Para os índices, o número de caracteres possíveis de ocorrer varia já que o dicionário é atualizado durante o processamento. O modelo dos índices é inicializado com o número de elementos do dicionário inicial. Cada vez que um elemento é adicionado ao dicionário, o modelo dos índices é expandido, isto é, o número de caracteres é modificado, assim como a probabilidade de ocorrência de cada índice.

Para uma melhor compreensão da implementação do codificador aritmético sugerimos a leitura de [6].

Capítulo 4

MMP Otimizado

No capítulo anterior, a árvore de segmentação do MMP foi determinada seguindo o critério da distorção-alvo. Neste capítulo será apresentado um algoritmo onde a árvore de segmentação é determinada seguindo um critério taxa-distorção. Uma versão do MMP com otimização taxa-distorção foi descrito por [3]. O algoritmo que apresentaremos é um pouco diferente do que foi proposto inicialmente, sendo superior a ele já que reduz a complexidade computacional. Em seguida explicaremos o dicionário de deslocamentos, que foi proposto por [2] a fim de explorar a característica quase periódica do sinal de ECG.

4.1 Otimização Taxa - Distorção

Vimos no capítulo anterior como o MMP descobre a árvore de segmentação S utilizando como critério a distorção-alvo. Para chegarmos à melhor relação entre a taxa e a distorção, determinaremos a árvore de segmentação ótima segundo um critério taxa-distorção. Queremos que para uma dada taxa-alvo R*, a distorção seja mínima. Ou então, que para uma dada distorção-alvo D*, a taxa seja mínima. Temos um problema de minimização com restrições, onde buscamos a árvore de segmentação ótima S^*:

$$\begin{aligned} &\text{minimizar } D(S) \\ &\text{sujeito a } R(S) = R* \end{aligned} \tag{4.1}$$

Utilizamos o Método do Multiplicador de Lagrange para encontrar a solução S* deste problema. Define-se então a função custo Lagrangeano:

$$J(S) = D(S) + \lambda R(S) \tag{4.2}$$

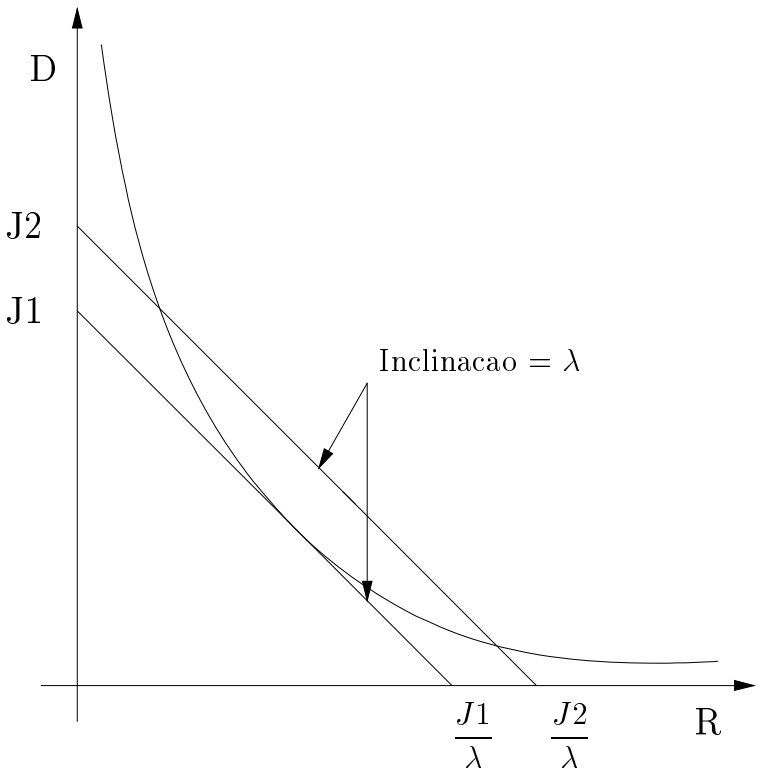


Figura 4.1:

onde $J(S)$, $D(S)$ e $R(S)$ representam, respectivamente, o custo, distorção e taxa para transmissão da árvore de segmentação S (sequência de flags e índices). Minimizar a distorção equivale a minimizar o custo Lagrangeano na equação acima. Temos agora um problema de minimização sem restrições:

$$\text{minimizar } J(S) = D(S) + \lambda R(S) \tag{4.3}$$

Minimizar o custo da Equação 4.3 para um valor de λ igual a 0 equivale a minimizar apenas a distorção. Para valores de λ muito grandes, apenas a taxa será minimizada. Valores intermediários de λ levam em consideração a taxa e a distorção.

A Figura 4.1 mostra uma curva RD para um determinado codificador. Podemos ver, para um valor de λ fixo, que quanto mais a reta $D = J - \lambda R$ estiver afastada da origem, maior será o custo J. O ponto na curva RD que minimiza o custo J é aquele no qual a reta $D = J - \lambda R$ tangencia a curva RD.

A distorção total $D(S)$ associada à árvore é encontrada somando-se as distorções $D(l)$ associadas a cada nodo folha:

$$D(S) = \sum_{l \in S_f} D(l), \tag{4.4}$$

onde $\mathbf{S_f}$ denota o conjunto de nodos folhas de S.

A distorção D(l) associada a um segmento X^l, como mostrado no capítulo 3, é obtida fazendo-se:

$$D(l) \ = \| \ \mathbf{X^l} - \mathbf{S_{i_l}} \ \|, \tag{4.5}$$

sendo $\mathbf{S_{i_l}}$ o elemento do dicionário escolhido para representar $\mathbf{X^l}$.

A taxa necessária para transmitir o conjunto de flags e índices que descrevem a árvore de segmentação S* é obtida somando-se a taxa para codificar os flags com a taxa para codificar os índices do dicionário, como mostra a equação a seguir:

$$R(S) = R_A(S) + \sum_{l \in S_f} R(l), \tag{4.6}$$

onde $R_A(S)$ é a taxa para transmitir todos os flags '0' ou '1' que representam a árvore e R(l) é o número de bits necessário para representar os índices do dicionário associados aos nodos folha. O número de bits para representar um flag é:

$$R(\text{flag}) = \log_2 \frac{1}{Pr_{\text{flag}}}, \tag{4.7}$$

onde Pr_{flag} é a probabilidade de ocorrência do flag. Nas implementações realizadas neste trabalho, ela é estimada através dos modelos do codificador aritmético. Da mesma forma, o número de bits necessário para representar o índice i_l do dicionário é:

$$R(l) = \log_2 \frac{1}{Pr(i_l|k_l)}, \tag{4.8}$$

onde $Pr(i_l|k_l)$ é a probabilidade de ocorrência do índice i_l no dicionário de escala k_l. Nas implementações realizadas neste trabalho, ela é estimada através dos modelos do codificador aritmético.

O custo para transmissão de S* é descrito pela equação (4.2). Utilizando as equações 4.4 a 4.6, temos:

$$\begin{aligned} J(S) &= D(S) + \lambda R(S) \\ &= \left(\sum_{l \in S_f} D(l) \right) + \lambda \left(R_A(S) + \sum_{l \in S_f} R(l) \right) \\ &= \lambda R_A(S) + \sum_{l \in S_f} (D(l) + \lambda R(l)) \\ &= \lambda R_A(S) + \sum_{l \in S_f} (J(l)), \end{aligned} \tag{4.9}$$

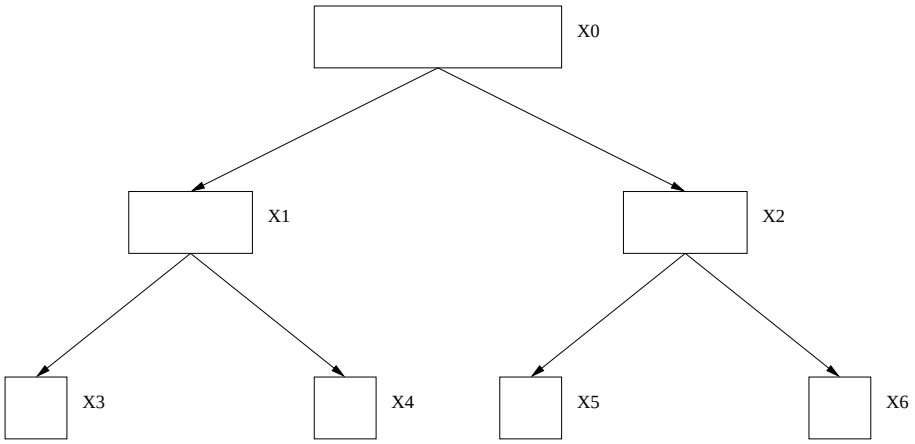


Figura 4.2: Árvore de segmentação completa

onde $J(l)$, o custo associado ao nodo l, é definido por:

$$\begin{aligned} J(l) &= D(l) + \lambda R(l) \\ &= \| \mathbf{X^l} - \mathbf{S^{i_l}} \| - \lambda \log_2 \frac{1}{Pr(i_l|k_l)} \end{aligned} \tag{4.10}$$

Vamos chamar $S(l)$ a sub-árvore binária composta por todos os nodos que têm l como raiz. O custo Lagrangeano associado à $S(l)$ é:

$$J(S(l)) = \lambda R_A(S(l)) + \sum_{l \in S_F \cap S(l)} J(l) \tag{4.11}$$

Se o nodo l for um nodo folha, o custo associado à sub-árvore $S(l)$ é:

$$J(S(l)) = D(l) + \lambda(R_l^1 + R(l)), \tag{4.12}$$

onde R_l^1 é a taxa necessária para codificar o flag 1.

Se o nodo l não for um nodo folha, o custo associado à sub-árvore $S(l)$ é:

$$J(S(l)) = \lambda R_l^0 + J(S(2l+1)) + J(S(2l+2)), \tag{4.13}$$

onde R_l^0 é a taxa necessária para codificar o flag 0 e $J(S(2l+1))$ e $J(S(2l+2))$ são os custos associados aos dois nodos filhos do nodo l.

Para implementar esta otimização, parte-se da árvore de segmentação completa, onde existem todos os nodos possíveis para cada nível (cada escala). Para isto divide-se o bloco $\mathbf{X}^j$ até chegar à menor escala, onde o bloco tem apenas 1 amostra, como mostra a Figura 4.2, para um bloco inicial de 4 amostras.

À medida que a árvore é criada, calcula-se o custo para cada sub-árvore $S(l)$ considerando que o nodo l é folha (nodo que não tem filhos). Esse custo é obtido utilizando-se a Equação 4.12:

Quando o MMP busca no dicionário o elemento que melhor representa o bloco que está sendo codificado, ele procura o elemento que fornece o menor custo $D + \lambda R$, onde D é descrito pela equação (4.5) e R, pela equação (4.8). É importante lembrar que no algoritmo simples, o elemento escolhido para representar um bloco $\mathbf{X}^j$ era aquele que fornecia a menor distorção.

Agora que temos a árvore com todos os nodos e seus respectivos custos, o algoritmo deve decidir se o nodo deve ser podado, isto é, se é mais vantajoso, em relação ao custo, fazer com que o nodo seja um nodo folha (sem filhos). Para isto, utiliza-se as Equações (4.12) e (4.13), verificando se o custo $J(S(l))$ considerando que o nodo l é folha é menor ou igual ao custo $J(S(l))$ considerando que o nodo l é segmentado:

$$J(l) \leq J(2l+1) + J(2l+2) + \lambda R_l^0 \tag{4.14}$$

Caso seja menor, é mais vantajoso podar os filhos do nodo pai. Caso contrário, decide-se manter os filhos. Nesse caso, o custo do nodo pai deve ser atualizado para

$$J(l) = J(2l+1) + J(2l+2) + \lambda R_l^0 \tag{4.15}$$

Podemos facilmente perceber que quando calcula-se o custo mínimo para um determinado nodo considerando que ele é um nodo folha, acha-se a tangente à sua curva RD, com inclinação -λ. Cada nodo terá uma distorção e taxa diferentes e, consequentemente, um custo diferente. Com base nos custos dos nodos pai e filhos, o MMP decide se segmenta ou não o bloco. Essa idéia é exemplificada na Figura 4.3.

Além disso, adiciona-se ao dicionário a concatenação das representações dos dois nodos filhos $\mathbf{X}^{2j+1}$ e $\mathbf{X}^{2j+2}$. Isso é importante porque este novo elemento passa a ser considerado no cálculo do custo dos nodos que estão à direita do nodo atual, podendo ser usado para representar algum desses nodos. No entanto, se em um nível superior, o nodo l for podado, a concatenação das representações de seus dois nodos filhos deve ser retirada do dicionário.

Vamos considerar que no exemplo da Figura 4.2 o custo $J(1) > J(3) + J(4) + \lambda R_l^0$. Nesse caso, a concatenação das representações dos nodos filhos de $\mathbf{X}^1$, $[\hat{\mathbf{X}}^{\mathbf{3}}\ \hat{\mathbf{X}}^{\mathbf{4}}]$ deve ser inserido no dicionário. Consideremos também que o custo $J(2) < J(5) + J(6) + \lambda R_l^0$ e que

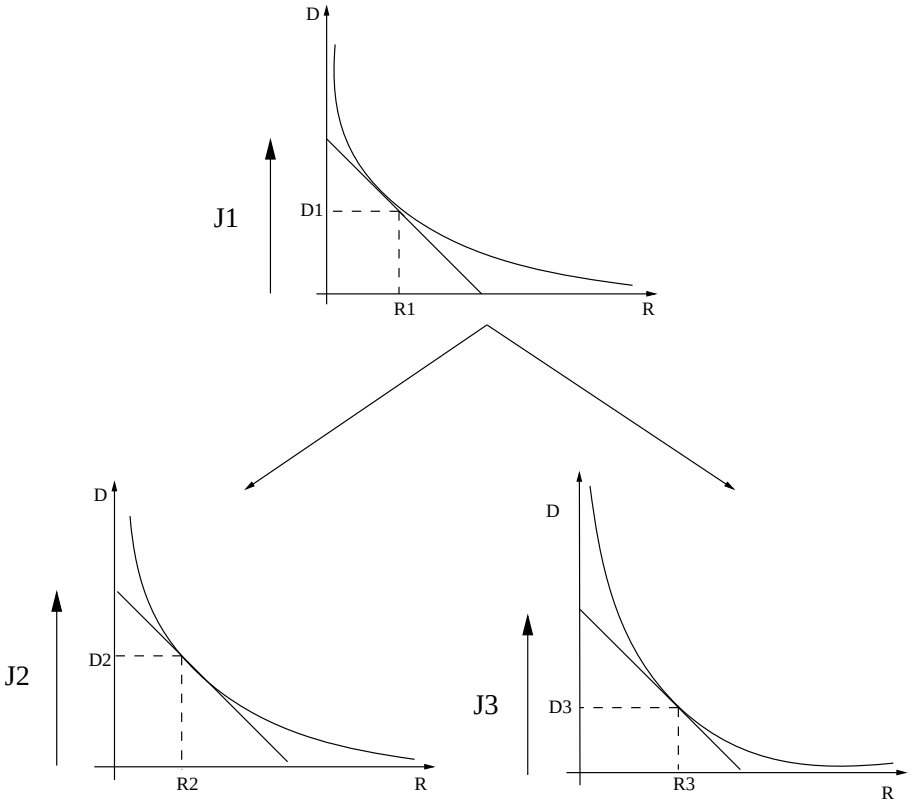


Figura 4.3:

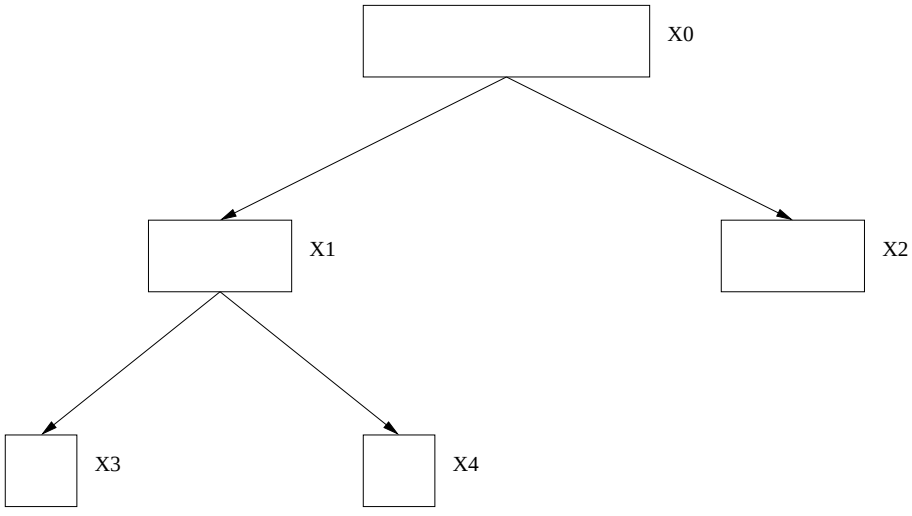


Figura 4.4: Árvore de segmentação resultante.

o custo $J(0) > J(1) + J(2) + \lambda R_l^0$. A concatenação das representações dos filhos de $\mathbf{X}^0$, $[\hat{\mathbf{X}}^{\mathbf{1}}\ \hat{\mathbf{X}}^{\mathbf{2}}]$ também deve ser inserida no dicionário. A árvore resultante é representada pela Figura 4.4

Apresentaremos um pseudo-código do algoritmo:

Considere que temos um vetor de dicíonários $\boldsymbol{D}$, com L coordenadas, uma para cada escala, e que cada dicionário possui um conjunto de elementos $S_1^l, S_2^l, S_3^l, ..., S_K^l$. Seja um vetor de entrada X^j, de tamanho $l(X^j)$, uma função $T(X^j, l(X^j), L)$ de transformação de escala, que transforma o tamanho $l(X^j)$ de X^j para L, e um multiplicador de Lagrange λ.

```
{S*, X̂^j} = otimizaRD(indPai, X^j, l(X^j), λ){
    S^l = D[l(X^j)]
    k = tamanhoDicionario(S^l)
    For i = 1:k{
        distorcao[i] = || X^j − S_i^l ||
        J[i] = distorcao + λR_i
    }
    indice = min(J)
    J[indPai] = J[indice] + λR_flag1
    {S*, X̂^(2j+1)} = otimizaRD(2indPai+1, X^j, l(X^j)/2, λ)
    {S*, X̂^(2j+2)} = otimizaRD(2indPai+2, X^j, l(X^j)/2, λ)
    If(l(X^j) == 1){
        return;
    }
    If (J[indPai] ≤ J[2indPai+1] + J[2indPai+2] + λR_flag0){
        S*[indPai] = 1;
        i = l(X^j)
        While(i > 1){
            If(S*[2indPai+1]==0)
                retiraDicionario(X̂^(2j+1))
            If(S*[2indPai+2]==0)
                retiraDicionario(X̂^(2j+2))
            i--
        }
        X̂^j = S_indice
    }
    Else{
        S*[indPai] = 0
        J[indPai] = J[2indPai+1] + J[2indPai+2] + λR_flag0
        X̂^j = [X̂^(2j+1) X̂^(2j+2)]
        L = 1
```

```
        For i = 1:  log M + 1{
             D(i) = T(X̂^j, l(X̂^j), L)
             L = L*2
        }
    }
}
```

```
X̂^j = codificaRD(indPai, X^j, l(X^j), λ, S*){
    if (S*[indPai] == 1){
        S^l = D[l(X^j)]
        k = tamanhoDicionario(S^l)
        For i = 1:k{
            distorcao[i] = || X^j − S_i^l ||
            J[i] = distorcao + λR_i
        }
        indice = min(J)
        codifica(Flag1)
        codifica(indice)
        X̂^j = S_indice
    }
    Else{
        codifica(Flag0)
        X^{2j+1} = codificaRD(2*indPai+1, X^j, l(X^j), λ, S*)
        X^{2j+2} = codificaRD(2*indPai+2, X^j, l(X^j), λ, S*)
        X̂^j = [X̂^{2j+1} X̂^{2j+2}]
        L = 1
        For i = 1: log M + 1{
            D(i) = T(X̂^j, l(X̂^j), L)
            L = L*2
        }
    }
}
```

O pseudo-código do decodificador é igual ao apresentado no capítulo 3.

Como vimos, esse algoritmo descobre a árvore de segmentação ótima a partir do nível mais baixo da árvore (escala de 1 amostra). Assim, é possível que em um nível superior (escalas maiores) decida-se podar o nodo $\mathbf{X}^j$, o que implica em retirar do dicionário elementos que foram adicionados em níveis inferiores.

Para facilitar esta implementação criamos um dicionário rascunho que irá conter os

elementos que são criados nessa fase de otimização. Quando um elemento deve ser retirado do dicionário rascunho, um flag indica que este elemento não é mais válido e, conseqüentemente, não deve ser considerado na busca de um elemento que melhor aproxima o bloco.

Durante esse procedimento também considera-se a variação da freqüência de ocorrência dos flags e dos índices dos dicionários. Até que o algoritmo encontre a árvore de segmentação ótima, a seqüência de flags que representa esta árvore e a ocorrência dos índices dos dicionários variam, sendo necessário atualizar constantemente os vetores de freqüência relativa. Para isso utiliza-se vetores auxiliares.

A atualização do dicionário rascunho segue o procedimento já explicado na seção 3.2, no entanto, deve ser verificada a existência do elemento novo tanto no dicionário rascunho como no dicionário normal.

4.1.1 Funções Principais

A seguir apresentaremos as principais funções do MMP otimizado taxa-distorção, além da descrição dos vetores de frequência auxiliares.

Vetores de frequência auxiliares:

- **unsigned long int **contComp**

  Tamanho: Número de escalas x (tamanho máximo do dicionário+1)

  Descrição: Armazena o número de ocorrências dos índices do dicionário ao longo da função otimizaRD (da posicão 1 à posição tamanhoDicMax+1) e o número total de ocorrências (posição 0). Para cada bloco que é codificado, todos os índices são inicializados como tendo 1 ocorrência.

- **unsigned long int **contNewDic**

  Tamanho: Número de escalas x (número de nodos - 1)

  Descrição: Armazena o número de ocorrências dos índices do dicionário rascunho ao longo da função otimizaRD (da posicão 1 a número de nodos-1) e o número total de ocorrências (posição 0). Para cada bloco que é codificado, todas as posições são inicializadas com valor nulo.

- **unsigned long int **contFlag**

Tamanho: Número de escalas x 3

Descrição: Armazena o número de ocorrências dos flags durante a função otimizaRD (da posição 1 à 3) e o número total de ocorrências na posição 0. Para cada bloco que é codificado, o flag 0 e o flag 1 são inicializados com 1 ocorrência.

Funções:

- **índice, context = buscaDicionarioOtimiza(**X^j**,** $l(X^j)$**,** λ**)**

  1 - Acessa o dicionário na escala $l(X^j)$.

  2 - Para todos os elementos do dicionário calcula-se a distorção $\| \mathbf{X^j - S_i} \|$ e a taxa R para codificar o índice. Em seguida calcula-se o custo J = D + λ R.

  3 - Caso $l(X^j)$ seja diferente de 1, repete-se os passos 1 e 2 para o dicionário rascunho. Caso contrário, vai para o passo 4.

  4 - O elemento que fornece o menor custo é escolhido para representar X^j.

  5 - Retorna-se o índice deste elemento e o dicionário ao qual pertence (context = 'I' indica o dicionário "normal"e context = 'A' indica o dicionário rascunho).

- **indice = buscaDicionarioCodifica(**X^j**,** $l(X^j)$**,** λ**)**

  1 - Acessa o dicionário na escala $l(X^j)$.

  2 - Para todos os elementos do dicionário calcula-se a distorção $\| \mathbf{X^j - S_i} \|$ e a taxa R para codificar o índice. Em seguida calcula-se o custo $J = D + \lambda R$.

  3 - O elemento que fornece o menor custo é escolhido para representar X^j.

  4 - Retorna-se o índice deste elemento.

- **atualizaDicionario(**X_0^j**,** $l(X_0^j)$**, L)**

  1 - X^j = transformaçãoEscala(X_0^j, $l(X_0^j)$, L)

  2 - Acessa o dicionário na escala L.

  3 - Verifica-se se X^j já existe no dicionário. Caso não exista e o dicionário não esteja com tamanho máximo, vai para o passo 4. Caso contrário, a função finaliza.

  4 - O elemento X^j é inserido no dicionário. O modelo dos índices é expandido.

- **atualizaDicionarioRascunho(X_0^j, $l(X_0^j)$, L)**

  1 - X^j = transformaçãoEscala(X_0^j, $l(X_0^j)$, L)

  2 - Acessa o dicionário rascunho na escala L.

  3 - Acessa o dicionário na escala L.

  4 - Verifica-se se X^j já existe no dicionário e no dicionário rascunho. Caso não exista em nenhum dos dicionários, vai para o passo 5. Caso contrário, a função finaliza.

  5 - O elemento X^j é inserido no dicionário.

- **S*, $X^j_{rascunho}$ = otimizaRD(X^j, indPai, $l(X^j)$)**

  1 - indice, context = buscaDicionarioOtimiza(X^j, $l(X^j)$, λ)

  2 - Calcula-se o custo considerando-se que o nodo indPai é folha: $J = D + \lambda(R_{indice} + R_{flag1})$ e armazena-se este custo no vetorCusto: vetorCusto[indPai] = J

  3 - Calcula-se a taxa do flag0, que é armazenada num vetor: vetorFlag0[indPai] = R_{flag0}

  4 - Caso o tamanho do bloco seja 1 amostra, incrementa-se a frequência do índice no vetor de frequência auxiliar referente ao dicionário utilizado (contComp ou contNewDic) e a frequência do flag1 no vetor de frequência auxiliar referente aos flags (contFlag).

  5 - $X^j_{rascunho} = S_{indice}$

  6 - Divide-se o vetor em X^{2j+1} e X^{2j+2}, de tamanhos iguais

  $$\text{S*, } X^{2j+1}_{rascunho} = \text{otimizaRD}(X^{2j+1}, \tfrac{l(Xj)}{2}, \lambda)$$

  $$\text{S*, } X^{2j+2}_{rascunho} = \text{otimizaRD}(X^{2j+2}, \tfrac{l(Xj)}{2}, \lambda)$$

  7 - Verifica se vetorCusto[indPai] $\leq$ vetorCusto[2indPai+1] + vetorCusto[2indPai+2] + λR_{flag0}. Caso seja menor vai para o passo 8. Caso contrário, vai para o passo 12.

  8 - S*[indPai] = 1

  9 - Verifica se os nodos descendentes (até o último nível) deste nodo têm filhos. Caso tenham filhos, os elementos que foram inseridos como concatenção destes filhos são retirados do dicionário rascunho.

  10 - $X^j_{rascunho} = S_{indice}$

  11 - Retorna $X^j_{rascunho}$

12 - S*[indPai] = 0

13 - vetorCusto[indPai] = vetorCusto[2indPai+1] + vetorCusto[2indPai+2] + λR_{flag0}

14 - Para todas as escalas possíveis:

atualizaDicionarioRascunho($X_{rascunho}$, $l(X_{rascunho})$, L)

- $\hat{X^j}$ = **codifica(S*, indPai,** X^j**,** $l(X^j)$**)**

1 - Se S*[indPai] for 1, vai para o passo 2. Caso contrário, vai para o passo 7.

2 - indice = buscaDicionarioCodifica(X^j, $l(X^j)$, λ)

3 - Codifica o flag1

4 - Codifica o índice do elemento encontrado.

5 - $\hat{X^j} = S_{indice}$

6 - Retorna $\hat{X^j}$

7 - O vetor X^j será segmentado em dois de mesmo tamanho e será chamada a função codifica para cada um deles:

$\hat{X^{2j+1}}$ = codifica(vetorFlag, 2*indPai+1, $\frac{l(Xj)}{2}$)

$\hat{X^{2j+2}}$ = codifica(vetorFlag, 2*indPai+2, $\frac{l(Xj)}{2}$)

8 - $\hat{X^j} = [\hat{X^{2j+1}}\hat{X^{2j+2}}]$

9 - Para todas as escalas:

atualizaDicionario(X^j, $l(X^j)$, escala)

- $\hat{X^j}$ = **decodifica(**$l(X^j)$**)**

1 - Decodifica o flag. Caso o flag seja 1 vai para o passo 2. Caso o flag seja 0, vai para o passo 5.

2 - Decodifica o índice.

3 - Acessa o dicionário na escala $l(X^j)$.

4 - retorna $\hat{X^j} = S_{indice}$

5 - Divide-se o bloco em X^{2j+1} e X^{2j+2} e chama-se a função para os 2 blocos:

$\hat{X^{2j+1}}$ = decodifica ($\frac{l(X^j)}{2}$)

$\hat{X^{2j+2}}$ = decodifica ($\frac{l(X^j)}{2}$)

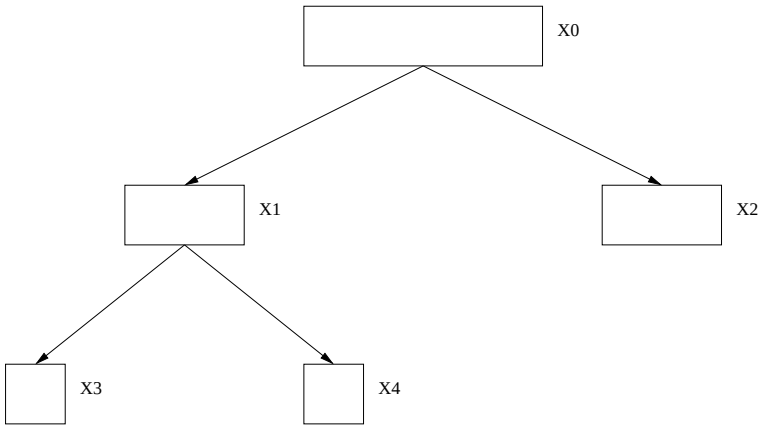


Figura 4.5: Árvore de segmentação resultante

6 - $\hat{X}^j = [\hat{X}^{2j+1}\hat{X}^{2j+2}]$

7 - Para todas as escalas:

atualizaDicionario($\hat{X}^j$, l($\hat{X}^j$), escala)

4.2 Dicionário de Deslocamentos

O Dicionário de Deslocamentos é utilizado para explorar melhor a característica "quase periódica"do sinal de eletrocardiograma. Como vimos nos capítulos 3 e 4, o MMP aprende os padrões típicos do sinal a partir da criação de um dicionário que contém versões contraídas/dilatadas de padrões previamente codificados. No entanto, como o tamanho do bloco usado para codificação dificilmente coincide com o período do sinal de ECG, a periodicidade do sinal não é bem explorada.

A proposta de [2] foi criar um outro tipo de dicionário, chamado de dicionário de deslocamentos, que consiste em um trecho do sinal que contém as últimas N amostras que foram codificadas. Esse tamanho N deve ser tal que compreenda vários períodos do sinal. Neste trabalho, assim como em [2], $N = 16M$, onde M é o tamanho do bloco.

Consideremos que a árvore de segmentação da Figura 4.5 seja gerada para um determinado bloco. Quando $\mathbf{X}^3$ tiver sido codificado, $\hat{\mathbf{X}}^3$ será inserido no dicionário de deslocamentos, seguido por $\hat{\mathbf{X}}^4$ e $\hat{\mathbf{X}}^2$. À medida que novos blocos são aplicados ao algoritmo e novos padrões estão disponíveis, estes são acrescentados ao dicionário de deslocamentos. Quando o dicionário estiver totalmente preenchido, para que um segmento de tamanho n seja inserido, as n amostras iniciais são descartadas e as novas amostras são adicionadas ao dicionário.

Com o Dicionário de Deslocamentos completo, ele também será utilizado para buscar uma representação para um segmento $\mathbf{X^j}$. Para fazer a busca no dicionário de deslocamentos, o segmento $\mathbf{X^j}$ será deslocado de uma amostra ao longo do dicionário, conforme a Figura 4.6. Os índices **i** mostrados na figura indicam o deslocamento que foi feito. A escolha do segmento que melhor representa o vetor $\mathbf{X^j}$ acontece de acordo com os critérios apresentados na seção 4.1. O índice do dicionário de deslocamentos que será codificado corresponde ao deslocamento **i** que forneceu a melhor representação para X^j. Neste exemplo, será codificado o índice 2, que foi o deslocamento que forneceu a melhor representação do bloco sendo codificado.

Podemos ver em [2] que se o sinal for rigorosamente periódico, o índice associado ao deslocamento que proporciona o melhor casamento será sempre o mesmo, para todos os segmentos, em todas as escalas, isto é, $i_L = i*$, para $L = L_1, L_2, ..., L_k$. Para sinais que não são rigorosamente periódicos, como o sinal de ECG, o algoritmo passa a codificar predominantemente os índices em torno de i*, índices em torno do período aproximado do sinal. Em qualquer caso, haverá uma grande redução na taxa.

Para que o decodificador possa decodificar corretamente o elemento, ele precisa saber a que dicionário se refere o índice que foi codificado. Para fazer a distinção entre o dicionário D e o Dicionário de Deslocamentos B, utiliza-se mais um flag. Após codificação do flag que indica *match*, codifica-se o flag 1, para indicar que o índice refere-se ao dicionário de deslocamentos, ou o flag 0, para indicar que o índice se refere ao dicionário D.

Assim como fazemos com os outros tipos de dicionário, também utilizamos um modelo estatístico para cada escala do Dicionário de Deslocamentos.

4.2.1 Funções Principais

Vetores de frequência auxiliares:

- **unsigend long int **contComp**

  Tamanho: número de escalas x (tamanho máximo do dicionário+1)

  Descrição: armazena o número de ocorrências dos índices do dicionário ao longo da função otimizaRD (da posicão 1 à posição tamanhoDicMax+1) e o número total de ocorrências (posição 0). Para cada bloco que é codificado, todos os índices são inicializados como tendo 1 ocorrência.

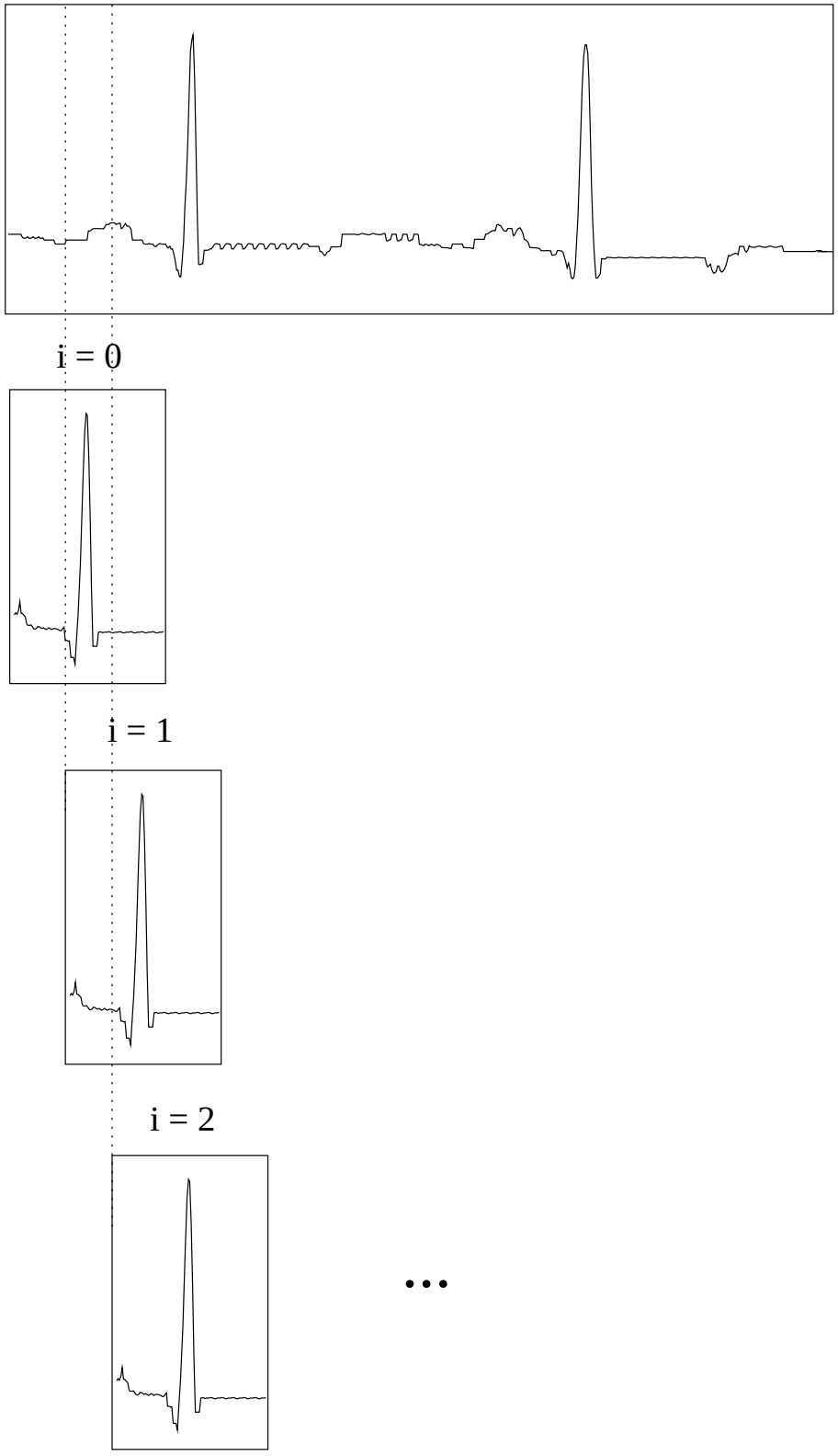


Figura 4.6: Exemplo do uso do Dicionário de Deslocamentos

- **unsigned long int **contNewDic**

  Tamanho: número de escalas x (número de nodos + 1)

  Descrição: armazena o número de ocorrências dos índices do dicionário rascunho ao longo da função otimizaRD (da posicão 1 a número de nodos-1) e o número total de ocorrências (posição 0). Para cada bloco que é codificado, todas as posições são inicializadas com valor nulo.

- **unsigned long int **contFlag**

  Tamanho: número de escalas x 3

  Descrição: armazena o núemero de ocorrências dos flags durante a função otimizaRD (da posição 1 à 3) e o número total de ocorrências na posição 0. Para cada bloco que é codifcado, o flag 0 e o flag 1 são inicializados com 1 ocorrência.

- **unsigned long int **contDesloc**

  Tamanho: número de escalas x (16*tamBloco+1)

  Descrição: armazena o número de ocorrências dos índices do dicionário de deslocamentos (da posição 1 à 16*tamBloco+1) e o número total de ocorrências na posição 0. Para cada bloco que é codificado, todos os índices são inicializados com 1 ocorrência.

Funções:

- **índice, context = buscaDicionarioOtimiza(**X^j**,** $l(X^j)$**,** λ**)**

  1 - Acessa o dicionário na escala $l(X^j)$

  2 - Para todos os elementos do dicionário calcula-se a distorção $\| \mathbf{X^j - S_i} \|$ e a taxa R para codificar o índice. Em seguida calcula-se o custo J = D + λ R.

  3 - Caso $l(X^j)$ seja diferente de 1, repete-se os passos 1 e 2 para o dicionário rascunho. Caso contrário, vai para o passo 4.

  4 - Se o dicionário de deslocamentos já estiver completo, repete-se os passos 1 e 2 para este dicionário.

  5 - O elemento que fornece o menor custo é escolhido para representar X^j.

6 - Retorna-se o índice deste elemento e o dicionário ao qual pertence (context = 'I' indica o dicionário "normal"e context = 'A' indica o dicionário rascunho e context = 'D' indica o dicionário de deslocamentos).

- **indice, context = buscaDicionarioCodifica(X^j, $l(X^j)$,λ)**

  1 - Acessa o dicionário na escala $l(X^j)$.

  2 - Para todos os elementos do dicionário calcula-se a distorção $\| \mathbf{X^j - S_i} \|$ e a taxa R para codificar o índice. Em seguida calcula-se o custo J = D + λ R.

  3 - Se o dicionário de deslocamentos já estiver completo, repete-se os passos 1 e 2 para este dicionário. Caso contrário, vai para o passo 4.

  4 - O elemento que fornece o menor custo é escolhido para representar X^j.

  5 - Retorna-se o índice deste elemento e o dicionário ao qual pertence (context = 'I' indica o dicionário "normal"e context = 'D' indica o dicionário de deslocamentos).

- **atualizaDicionario(X_0^j, $l(X_0^j)$, L)**

  1 - X^j = transformaçãoEscala(X_0^j, $l(X_0^j)$, L)

  2 - Acessa o dicionário na escala L

  3 - Verifica-se se X^j já existe no dicionário. Caso não exista e o dicionário não esteja com tamanho máximo, vai para o passo 4. Caso contrário, a função finaliza.

  4 - O elemento X^j é inserido no dicionário. O modelo dos índices é expandido.

- **atualizaDicionarioRascunho(X_0^j, $l(X_0^j)$), L)**

  1 - X^j = transformaçãoEscala(X_0^j, $l(X_0^j)$, L)

  2 - Acessa o dicionário rascunho na escala L

  3 - Verifica-se se X^j já existe no dicionário. Caso não exista, vai para o passo 4. Caso contrário, a função finaliza.

  4 - O elemento X^j é inserido no dicionário.

- **atualizaDicionarioDeslocamentos(X^j, $l(X^j)$)**

  1 - Verifica se o tamanho do dicionário de deslocamentos é 16*tamBloco. Caso seja, vai para o passo 2. Caso contrário, vai para o passo 3.

2 - Adiciona-se X^j ao dicionário de deslocamentos

3 - Retira-se as $l(X^j)$ amostras mais antigas e adiciona-se X^j.

- **S*, $X^j_{rascunho}$ = otimizaRD(indPai, X^j, $l(X^j)$)**

1 - indice, context = buscaDicionarioOtimiza(X^j, $l(X^j)$, λ)

2 - Calcula-se o custo considerando-se que o nodo indPai é folha: $J = D + \lambda(R_{indice} + R_{flag1})$ e armazena-se este custo no vetorCusto: vetorCusto[indPai] = J

3 - Calcula-se a taxa do flag0, que é armazenada num vetor: vetorFlag0[indPai] = R_{flag0}

4 - Caso o tamanho do bloco seja 1 amostra, incrementa-se a frequência do índice no vetor de frequência auxiliar referente ao dicionário utilizado (contComp, contNewDic ou contDesloc) e a frequência do flag1 no vetor de frequência auxiliar referente aos flags (contFlag).

5 - $X^j_{rascunho} = S_{indice}$

6 - Divide-se o vetor em X^{2j+1} e X^{2j+2}, de tamanhos iguais

$$\text{S*, } X^{2j+1}_{rascunho} = \text{otimizaRD}(X^{2j+1}, \frac{l(Xj)}{2}, \lambda)$$

$$\text{S*, } X^{2j+2}_{rascunho} = \text{otimizaRD}(X^{2j+2}, \frac{l(Xj)}{2}, \lambda)$$

7 - Verifica se vetorCusto[indPai] $\leq$ vetorCusto[2indPai+1] + vetorCusto[2indPai+2] + λR_flag0. Caso seja menor vai para o passo 8. Caso contrário, vai para o passo 12.

8 - S*[indPai] = 1

9 - $X^j_{rascunho} = S_{indice}$

10 - Verifica se os nodos descendentes (até o último nível) deste nodo têm filhos. Caso tenham filhos, os elementos que foram inseridos como concatenção destes filhos são retirados do dicionário rascunho.

11 - Retorna $X^j_{rascunho}$

12 - S*[indPai] = 0

13 - vetorCusto[indPai] = vetorCusto[2indPai+1] + vetorCusto[2indPai+2] + λR_{flag0}

14 - Para todas as escalas possíveis:

atualizaDicionarioRascunho($X^j_{rascunho}$, $l(X^j_{rascunho})$, escala)

- $\hat{X^j}$ = **codifica(S*, indPai,** X^j**,** $l(X^j)$**)**

  1 - Se S*[indPai] for 1, vai para o passo 2. Caso contrário, vai para o passo 9.

  2 - indice = buscaDicionarioCodifica(X^j, $l(X^j)$, λ)

  3 - Codifica o flag1 de *match*

  4 - Caso o elemento encontrado seja do dicionário de deslocamentos, codifica-se o flag1. Caso contrário, codifica-se o flag0.

  5 - Codifica o índice do elemento encontrado.

  6 - $\hat{X}^j = S_{indice}$

  7 - atualizaDicionarioDeslocamentos(X^j, $l(X^j)$)

  8 - Retorna $\hat{X}^j$

  9 - Codifica-se o flag 0 de segmentação

  10 - O vetor X^j será segmentado em dois de mesmo tamanho e será chamada a função codifica para cada um deles:

  $$X^{\hat{2j+1}} = \text{codifica(vetorFlag, 2*indPai+1, } \tfrac{l(Xj)}{2})$$

  $$X^{\hat{2j+2}} = \text{codifica(vetorFlag, 2*indPai+2, } \tfrac{l(Xj)}{2})$$

  11 - $X^j = [X^{\hat{2j+1}} X^{\hat{2j+2}}]$

  12 - Para todas as escalas:

  atualizaDicionario(X^j, $l(X^j)$, escala)

- $\hat{X^j}$ = **decodifica(**$l(X^j)$**)**

  1 - Decodifica o flag. Caso o flag seja 1 vai para o passo 2. Caso o flag seja 0, vai para o passo 12.

  2 - Decodifica o flag de deslocamento.

  3 - Decodifica o índice.

  4 - Caso o flag de deslocamento seja 1, vai para o passo 5. Caso contrário vai para o passo 8.

  5 - $\hat{X}^j = S^B_{indice}$

  6 - atualizaDicionarioDeslocamentos(X^j, $l(X^j)$)

7 - Retorna $\hat{X}^j$

8 - Acessa o dicionário D na escala $l(X^j)$.

9 - $\hat{X}^j = S^D_{indice}$

10 - atualizaDicionarioDeslocamentos(X^j, $l(X^j)$)

11 - Retorna $\hat{X}^j$

12 - Divide-se o bloco em X^{2j+1} e X^{2j+2} e chama-se a função para os 2 blocos:

$$\hat{X}^{2j+1} = \text{decodifica } (\tfrac{l(X^j)}{2})$$

$$\hat{X}^{2j+2} = \text{decodifica } (\tfrac{l(X^j)}{2})$$

13 - $\hat{X}^j = [\hat{X}^{2j+1}\hat{X}^{2j+2}]$

14 - Para todas as escalas:

atualizaDicionario($\hat{X}^j$, l($\hat{X}^j$), L)

Capítulo 5

MMP aplicado ao sinal de ECG

Esse capítulo irá apresentar as modificações propostas a fim de melhorar o desempenho do MMP para compressão de sinais de eletrocardiograma.

5.1 Tratando ECG como Imagem

O sinal de ECG, como pode ser verificado pela Figura 5.1 tem uma característica "quase periódica", que deve ser bem explorada a fim de conseguir uma compressão mais eficiente. Como o MMP obtém bons resultados com compressão de imagens, testamos a compressão de sinais de ECG utilizando o MMP, comsiderando o sinal de ECG como uma imagem (sinal bidimensional).

5.1.1 Gerando a Imagem

Para converter o sinal de ECG unidimensional para um sinal de 2 dimensões, é necessário dividir o sinal em blocos, onde cada bloco será uma linha da matriz que representa a imagem. A fim de manter uma maior correlação entre blocos vizinhos, dividimos o sinal de tal forma que cada linha dessa matriz tivesse aproximadamente um múltiplo do período do sinal de ECG.

Como o MMP impõe restrições quanto ao tamanho do bloco (codificação bloco a bloco), que deve ser uma potência de 2, o número de colunas da matriz deve ser um múltiplo desse tamanho. Assim, após estimar o período do sinal, encontramos o múltiplo do tamanho do bloco mais próximo deste valor. Para isso foi necessário completar o sinal com zero, de forma que o tamanho do sinal fosse múltiplo do número de colunas, evitando que algumas

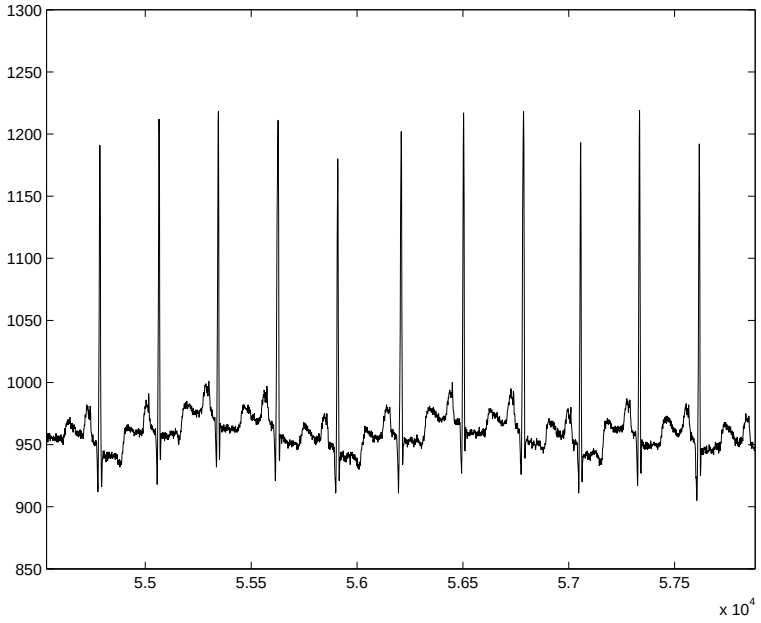


Figura 5.1: Trecho de um sinal de ECG

amostras fossem descartadas.

A Figura 5.2 exemplifica o que foi explicado acima, mostrando como seria a imagem gerada através do sinal de ECG.

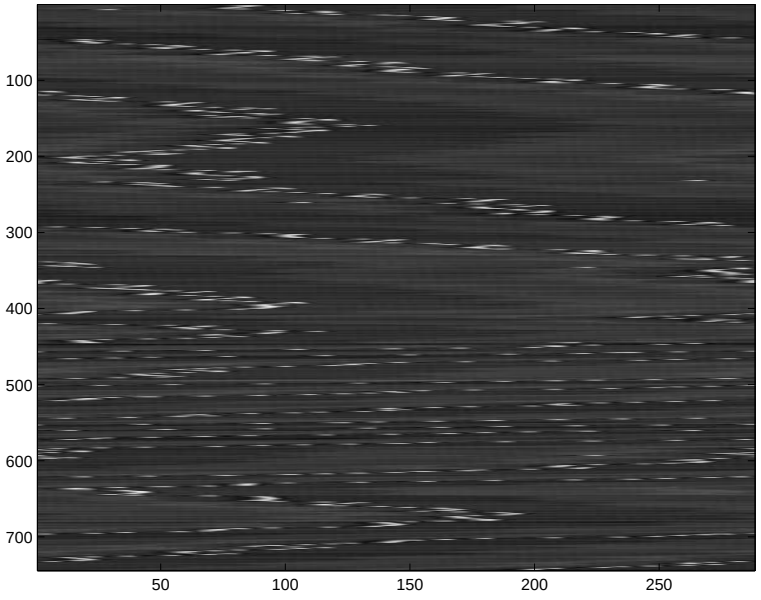


Figura 5.2: ECG como imagem

A estimativa do período do sinal foi feita através da autocorrelação do sinal de ECG. A autocorrelação pode ser obtida através da transformada de Fourier inversa da densidade espectral de potência do sinal x[n].

A densidade espectral de potência fazendo-se:

$$S_X(w) = |X(\omega)|^2, \tag{5.1}$$

onde X(w) é a transformada de Fourier de x[n].

Assim, a autocorrelação é calculada como:

$$R_{xx}(\tau) = \mathcal{F}^{-1}\{S_X(\omega)\}, \tag{5.2}$$

considerando-se que o sinal x[n] é ergódico.

Para sinais periódicos, a cada período, as amostras se repetem e portanto têm correlação alta. Dessa forma, a autocorrelação também será periódica, com período igual ao período do sinal. Como o sinal de ECG é quase periódico, a autocorrelação fornece uma boa estimativa do período. Apesar do período poder variar ao longo do sinal, consideramos como sendo o período a diferença entre o primeiro e o segundo pico do sinal de autocorrelação.

5.1.2 Resultados

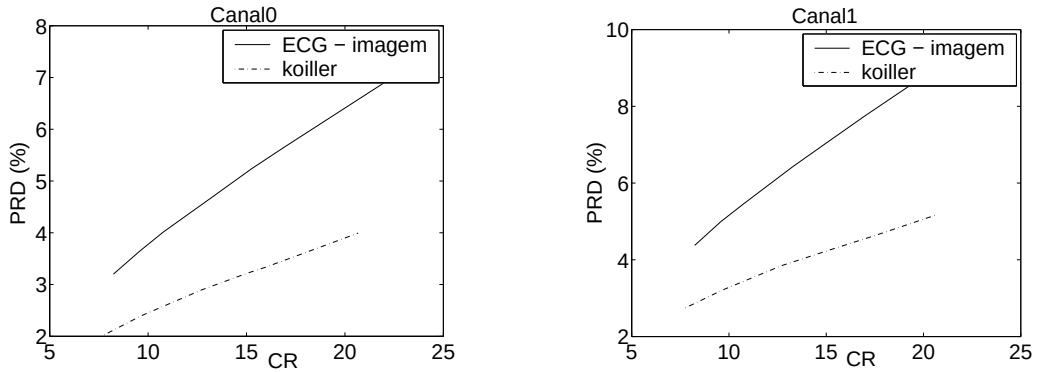
Apresentaremos os resultados obtidos utilizando o algoritmo proposto, que foi aplicado ao sinal 100.dat, variando-se a forma de gerar a imagem a partir do sinal de ECG. Fizemos simulações onde o número de colunas da imagem gerada foi igual a:

- T (Figura 5.3)
- 2*T (Figura 5.4)
- 3*T (Figura 5.5),

onde T é o período estimado do sinal.

Tabela 5.1: Resultados fazendo-se o número de colunas da imagem igual ao período estimado do sinal.

CR	**PRD0**	**PRD1**
8.239	3.201	4.375
9.585	3.648	4.994
10.776	4.014	5.465
13.272	4.693	6.423
15.275	5.242	7.140
17.025	5.681	7.762
23.060	7.160	9.826

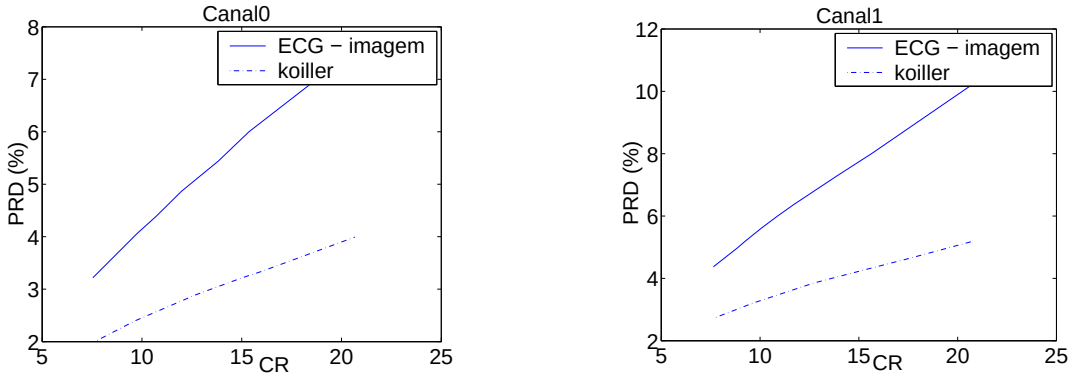


(a) (b)

Figura 5.3: Curva Taxa-Distorção para o sinal 100, fazendo-se o número de colunas igual ao período estimado do sinal: (a) Canal0, (b) Canal1

Tabela 5.2: Resultados para o sinal 100.dat fazendo-se o número de colunas da imagem igual ao dobro do período estimado do sinal.

CR	PRD0	PRD1
7.643	3.218	4.372
8.860	3.683	4.979
9.912	4.062	5.534
10.885	4.386	5.995
12.236	4.864	6.599
14.110	5.441	7.372
15.639	5.998	7.999
21.605	7.689	10.594

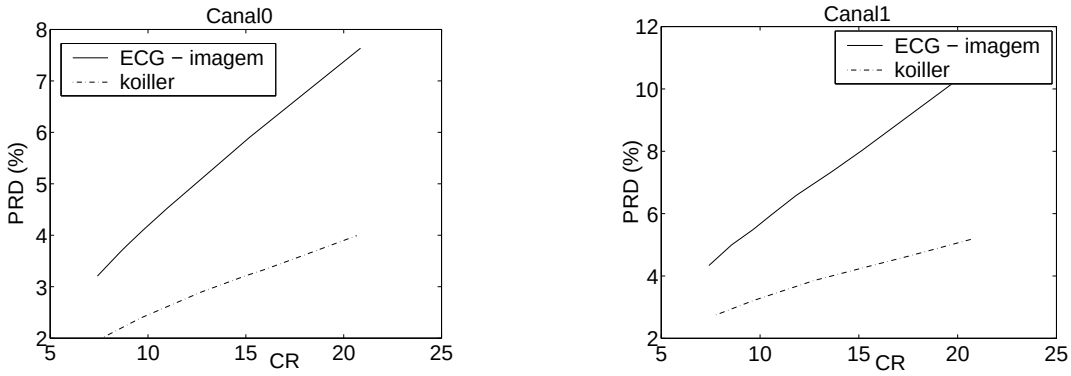


(a) (b)

Figura 5.4: Curva Taxa-Distorção para o sinal 100, fazendo-se o número de colunas da imagem igual ao dobro do período estimado do sinal: (a) Canal0, (b) Canal1

Tabela 5.3: Resultados para o sinal 100.dat fazendo-se o número de colunas da imagem igual ao triplo do período estimado do sinal.

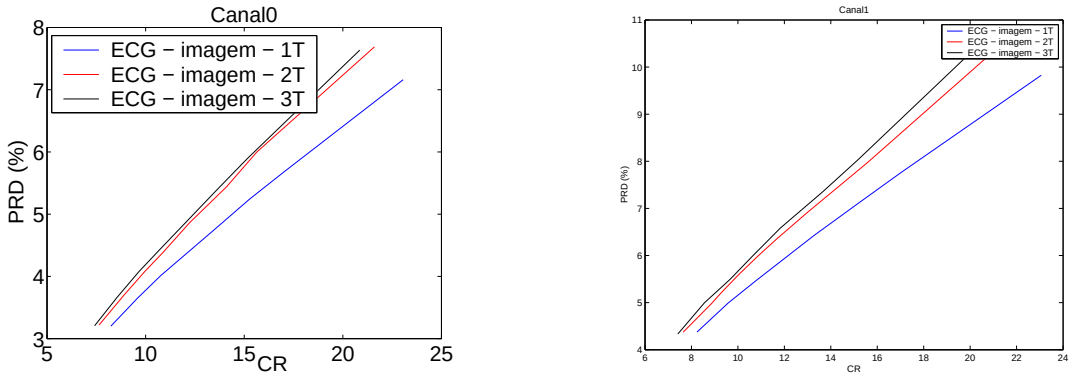
CR	**PRD0**	**PRD1**
7.415	3.207	4.334
8.564	3.667	4.998
9.658	4.075	5.491
10.598	4.392	5.972
11.834	4.800	6.586
13.640	5.399	7.347
15.174	5.897	8.033
20.860	7.636	10.700



(a) (b)

Figura 5.5: Curva Taxa-Distorção para o sinal 100, fazendo-se o número de colunas da imagem igual ao triplo do período estimado do sinal: (a) Canal0, (b) Canal1

A Figura 5.6 mostra uma comparação entre os resultados apresentados acima.



(a) (b)

Figura 5.6: Curva Taxa-Distorção para o sinal 100, comparando-se os resultados apresentados acima: (a) Canal0, (b) Canal1

Podemos ver pelos resultados apresentados que o método proposto para compressão de sinais de ECG, apesar de eficiente, não foi superior ao algoritmo proposto por [2]. Certamente, o sinal de ECG tratado como sinal unidimensional apresenta características que são melhor aproveitadas pelo MMP.

Dessa forma, nas próximas seções iremos propor outras alterações ao algoritmo de [2] para melhorar a eficiência do MMP para sinais de ECG.

5.2 Retirando o nível DC

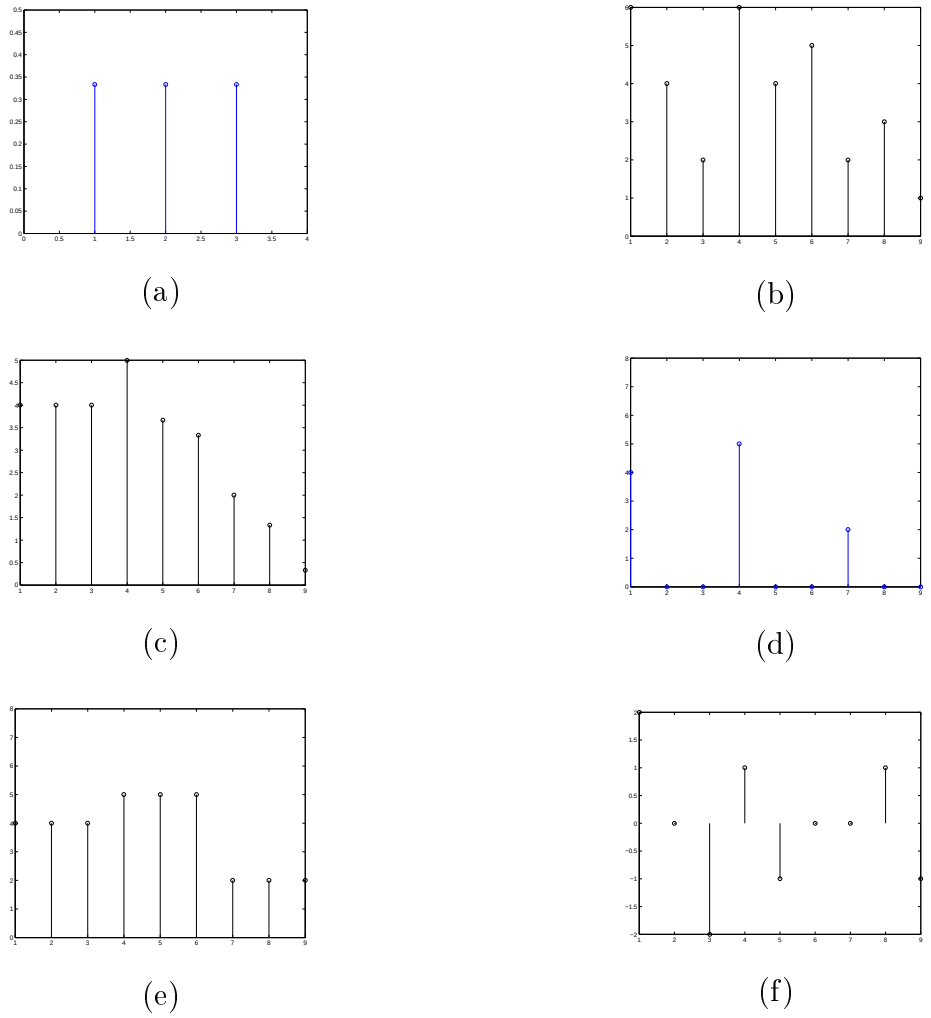
Apesar da característica "quase periódica"do sinal de ECG, existem variações no nível DC do sinal a cada período, como pode ser visto na Figura 5.1. Como vimos no capítulo 4 o MMP utiliza um dicionário de deslocamentos que contém $16M$ amostras (M = tamanho do bloco para codificação) referentes às últimas $16M$ amostras que foram codificadas. Para encontrar uma representação para o bloco que está sendo codificado, também utiliza-se esse dicionário.

O bloco é deslocado ao longo do dicionário e é escolhido o trecho que fornece a melhor representação deste bloco. As pequenas variações no nível DC do sinal podem fazer com que o dicionário não seja bem utilizado. Assim, se for possível equalizar o sinal, reduzindo essas variações, poderemos explorar melhor o dicionário e aumentar a eficiência do algoritmo na compressão de sinais eletrocardiográficos.

Uma forma de reduzir essas variações seria retirando o nível DC do sinal. Para isso, aplicamos um filtro média-móvel $h_m[n]$ (filtro passa-baixa), com comprimento igual a B. Em

seguida, o sinal resultante é decimado de B amostras, isto é, existe uma amostra diferente de zero a cada B amostras. Finalmente, interpola-se este sinal de B amostras e aplica-se um filtro passa-baixa. Como resultado, todas as amostras pertencentes a um bloco de B amostras terão o mesmo valor, igual à média aritmética deste bloco . Esse processo é esquematizado na Figura 5.11.

Considerando um sinal x[n] = {6 4 2 6 4 5 2 3 1} e um filtro $h[n] = \{\frac{1}{3}\ \frac{1}{3}\ \frac{1}{3}\}$, o sinal resultante do processo descrito acima seria $y[n] = \{4\ 4\ 4\ 5\ 5\ 5\ 2\ 2\ 2\}$, como descrito na Figura 5.7.



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

Figura 5.7: A Figura (a) representa o filtro média-móvel e a Figura (b), o sinal. A Figura (c) é o resultado da filtragem do sinal por este filtro. Na Figura (d) vemos o resultado da decimação de B amostras deste sinal resultante , na Figura (e) vemos o sinal após interpolação de B amostras e finalmente na Figura (f) podemos ver o sinal sem a média.

Na prática, o que fizemos foi dividir o sinal original em blocos de tamanho B e calcular a média aritmética para cada bloco. Para todas as amostras de um determinado bloco sub-

traímos a média obtida para aquele bloco. Para que o sinal fosse decodificado corretamente, foi necessário codificar as médias encontradas. Após etapa de transformação e quantização, as médias foram codificadas utilizando o codificador aritmético, com um modelo adaptativo à parte para estimar as probabilidades dos símbolos.

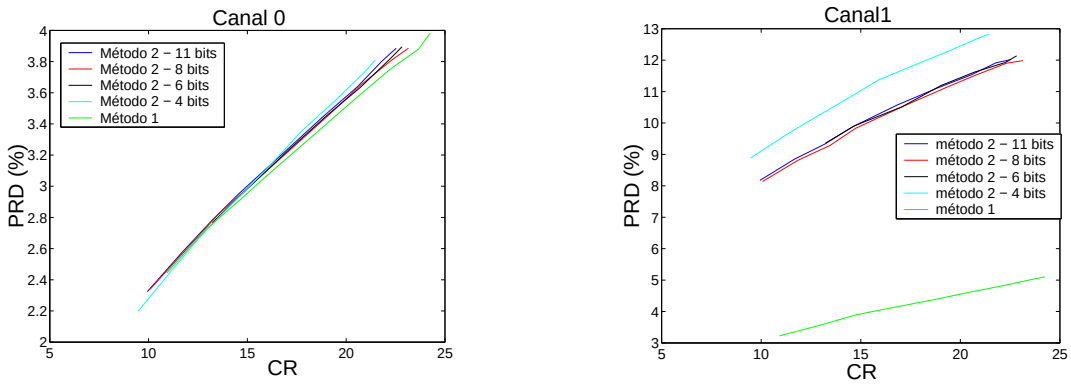
Para transmitir a média fizemos dois testes distintos, onde variamos a etapa de transformação. Esta etapa transforma o conjunto de valores que deseja-se codificar em um conjunto de valores com correlação menor (menos redundância entre os dados) e portanto, mais adequado para serem quantizados. A etapa de quantização reduz o número de símbolos que serão codificados e consequentemente, o número de bits que será necessário para representá-los.

O primeiro método consistiu em levar o intervalo de valores da média ($[\bar{X}_{min}\ \ \bar{X}_{max}]$) para o intervalo $[0\ \ (\bar{X}_{max} - \bar{X}_{min})]$. Após essa etapa de transformação, cada valor (do tipo *double*) foi atribuído a uma variável do tipo *int*, caracterizando uma etapa de quantização implícita. Esses valores quantizados foram então codificados por um codificador de entropia, o codificador aritmético.

No segundo método a etapa de transformação levou o intervalo de valores da média ($[\bar{X}_{min}\ \ \bar{X}_{max}]$) para o intervalo $[0\ \ 1]$. Em seguida cada valor foi multiplicado por 2^n, onde n é $\log_2(L+1)$ e L é o número de níveis de reconstrução. Cada valor (do tipo *double*) foi atribuído a uma variável do tipo *int*. Para chegar ao melhor resultado, fizemos testes com o sinal 101.dat, quantizando a média com 4, 6, 8 e 11 bits. Os valores resultantes foram codificados com o codificador aritmético.

Após os dois métodos aplicamos o MMP ao sinal sem a média.

Podemos ver na Figura 5.8 que o primeiro método proposto apresentou resultados melhores e portanto, foi utilizado no restante das simulações.



(a) (b)

Figura 5.8: Comparação entre os diferentes métodos de codificação da média: (a) Canal0, (b) Canal1

Seguindo a idéia apresentada acima, aplicamos um filtro $h_c[n]$ mais suave, que ao contrário do filtro média-móvel, faria uma média ponderada. Em relação ao filtro média móvel, com comprimento igual a B, o filtro cosseno $h_c[n]$ possui comprimento igual a 2*B.

$$y_c[n] = \frac{1}{2} - \frac{1}{2}cos(\pi\frac{n}{B}) \tag{5.3}$$

Para entender as vantagens do filtro cosseno sobre o filtro média-móvel devemos lembrar que o MMP é aplicado a cada bloco de 64 amostras do sinal. No decodificador, o sinal é reconstruído bloco a bloco, isto é, cada bloco é decodificado e concatenado com os blocos anteriores. Caso as últimas amostras de um bloco sejam muito diferentes das primeiras amostras dos blocos vizinhos, o sinal apresentará efeito de blocos, caracterizado por essa discontinuidade entre blocos vizinhos. O filtro média-móvel apresenta grande descontinuidade no tempo e portanto, irá gerar descontinuidade quando aplicado a um sinal. Já o filtro cosseno é mais suave, o que reduz o efeito de blocos que seria produzido pelo filtro média-móvel.

A filtragem utilizando o filtro $h_c[n]$ foi feita multiplicando-se cada amostra de um bloco de 2*B amostras do sinal pela amostra correspondente de $h_c[n]$ e após somarmos os resultados e dividirmos pelo somatório dos valores do filtro utilizados no cálculo, obtivemos o valor da média ponderada para o bloco de B amostras do sinal correspondente. O filtro $h_c[n]$ não altera a média quando aplicado a um sinal constante. Essa relação entre as amostras utilizadas para calcular a média e o bloco ao qual essa média se refere pode ser visualizada na Figura 5.9. O filtro foi deslocado de B amostras , repetindo-se este processo até o final do sinal. É

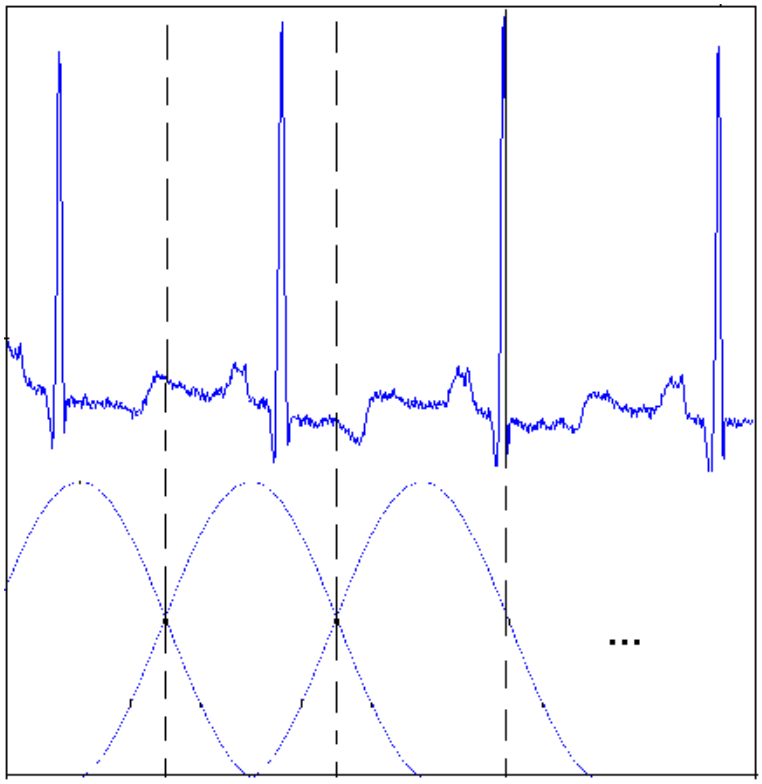


Figura 5.9: Filtro cosseno aplicado ao sinal de ECG

importante destacar que para calcular a média do primeiro bloco do sinal utilizamos apenas $\frac{3B}{2}$ amostras do sinal e de $h_c[n]$, que assumiu valores no intervalo $[\frac{-B}{2}\ B]$. Na Figura 5.10 podemos ver a relação entre o filtro cosseno e o filtro média-móvel.

O primeiro desafio encontrado nos dois algoritmos propostos foi a escolha do comprimento $l(y[n])$ desse filtro, que deveria ter um valor que otimizasse a codificação. Primeiro pensamos em fazer $l(y[n])$ igual ao período estimado do sinal. Mas como esse período é variável, achamos razoável que essa não seria uma boa forma de fazer a filtragem. Então decidimos fazer testes onde $l(y[n])$ seria dependente do tamanho do bloco usado para codificação. Assim, para um determinado sinal, testamos 4 valores diferentes $l(y[n])$. Utilizamos o valor que forneceu o melhor resultado para fazer os testes com esse algoritmo nos outros sinais ($l(y[n]) = 3tamBloco$).

5.2.1 Resultados

A fim de verificar qual dos filtros propostos seriam mais eficientes, aplicamos os dois algoritmos ao sinal 101.dat, para filtros com $l(y[n]) = 9 * tamBloco$.

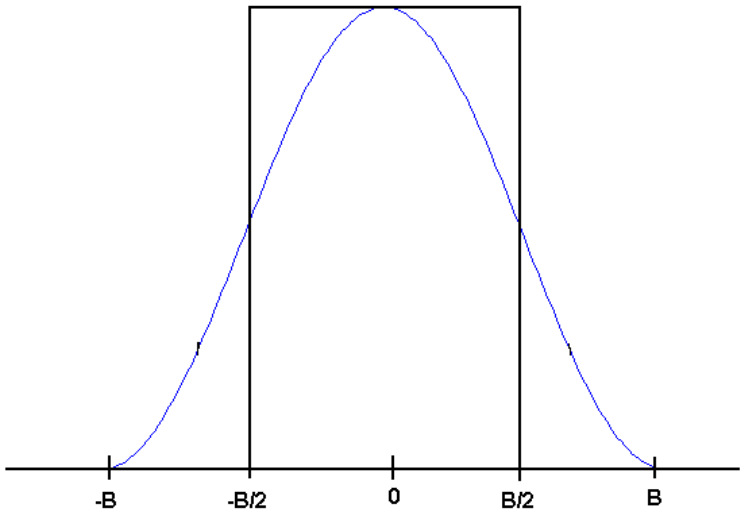


Figura 5.10: Relação entre o filtro cosseno e o filtro média-móvel

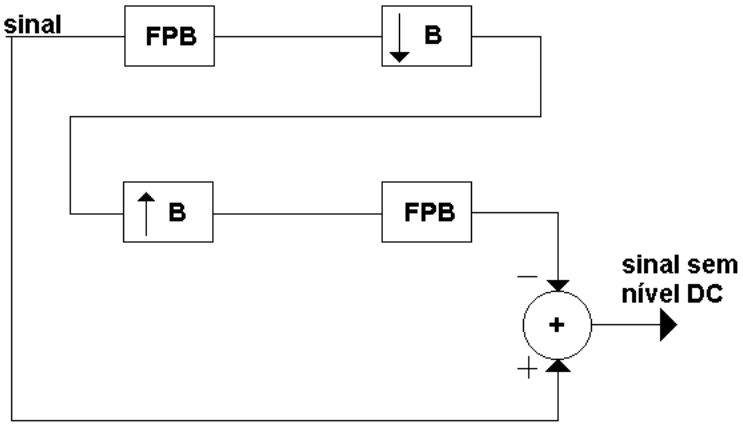
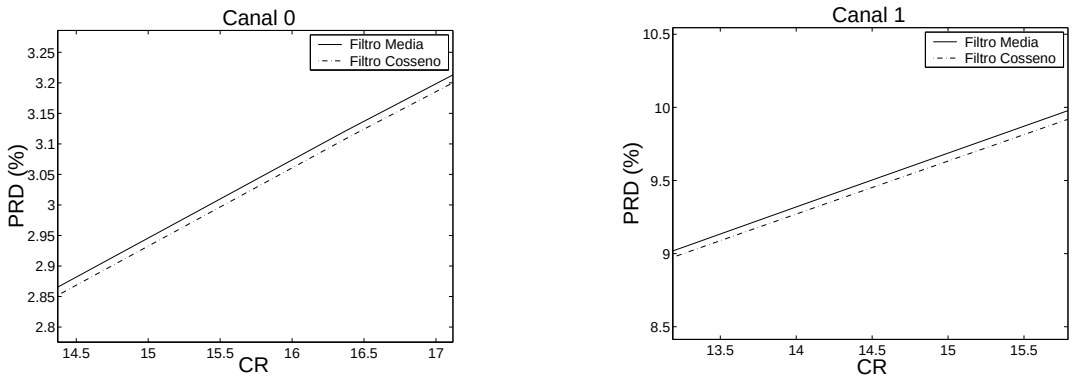


Figura 5.11: Esquema para retirar o nível DC do sinal de ECG



(a) (b)

Figura 5.12: Comparação entre o filtro média e o filtro cosseno: (a) Canal0, (b) Canal1

Podemos ver pelos gráficos 5.12 que as curvas que descrevem o comportamento do codificador utilizando o filtro média e o filtro cosseno estão bastante próximas. Um "zoom" aplicado a essas figuras permitem que visualizemos a curva referente ao filtro cosseno um pouco abaixo da outra curva. Dessa forma optamos por utilizar apenas o filtro cosseno no restante das simulações.

Apresentaremos os resultados obtidos utilizando o sinal 101.dat, para quatro $l(y_c[n])$ diferentes e dependentes do tamanho do bloco utilizado para codificação: B = tamBloco, B = 3*tamBloco, B = 5*tamBloco, B = 9*tamBloco. O tamanho do bloco foi de 64 amostras.

Tabela 5.4: 101.dat - l(y[n]) = Tamanho do bloco = 64

CR	**PRD0**	**PRD1**
9.682	2.160	7.514
11.494	2.449	8.255
12.945	2.668	8.809
14.226	2.859	9.302
16.394	3.156	9.946
18.057	3.372	10.397
19.456	3.594	10.873
20.647	3.745	11.162
21.257	3.846	11.437

Tabela 5.5: 101.dat - l(y[n]) = 3*Tamanho do bloco = 192

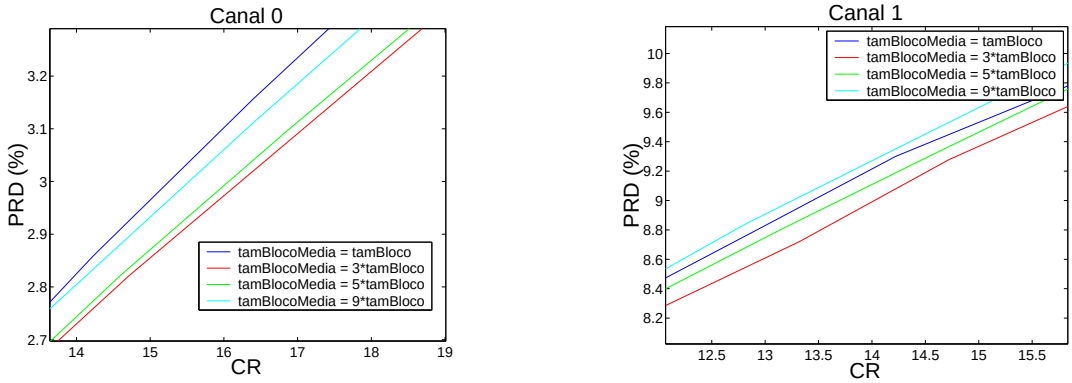
CR	**PRD0**	**PRD1**
10.012	2.160	7.466
11.764	2.428	8.180
13.320	2.643	8.720
14.721	2.822	9.278
17.026	3.094	10.026
19.103	3.339	10.574
20.657	3.536	11.047
22.148	3.707	11.444
22.910	3.788	11.551

Tabela 5.6: 101.dat - l(y[n]) = 5*Tamanho do bloco = 320

CR	**PRD0**	**PRD1**
9.831	2.141	7.438
11.582	2.416	8.219
13.153	2.633	8.805
14.601	2.822	9.325
16.859	3.096	10.119
18.875	3.332	10.697
20.484	3.529	11.158
21.987	3.699	11.493
22.904	3.798	11.727

Tabela 5.7: 101.dat - l(y[n] = 9*Tamanho do bloco = 576

CR	PRD0	PRD1
9.603	2.174	7.463
11.390	2.446	8.260
12.808	2.653	8.837
14.250	2.837	9.361
16.407	3.113	10.142
18.278	3.342	10.786
19.815	3.523	11.265
21.384	3.704	11.639
21.944	3.803	11.836



(a) (b)

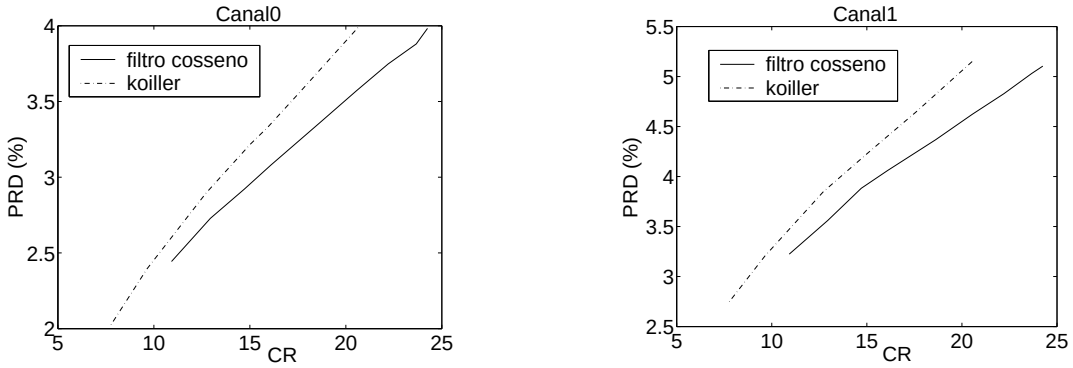
Figura 5.13: Comparação entre os diferentes valores de $y_c[n]$: (a) Canal0, (b) Canal1

Podemos ver pelos resultados apresentados que utilizar $l(y[n]) = 3*tamBloco$ forneceu resultados melhores.

Os resultados que serão apresentados para os demais sinais foram obtidos utilizando $l(y[n] = 3 * tamBloco$.

Tabela 5.8: 100.dat

CR	**PRD0**	**PRD1**
10.922	2.443	3.224
12.945	2.729	3.560
14.709	2.921	3.883
16.099	3.080	4.061
18.623	3.357	4.369
20.555	3.571	4.624
22.203	3.748	4.828
23.669	3.880	5.030
24.256	3.983	5.104

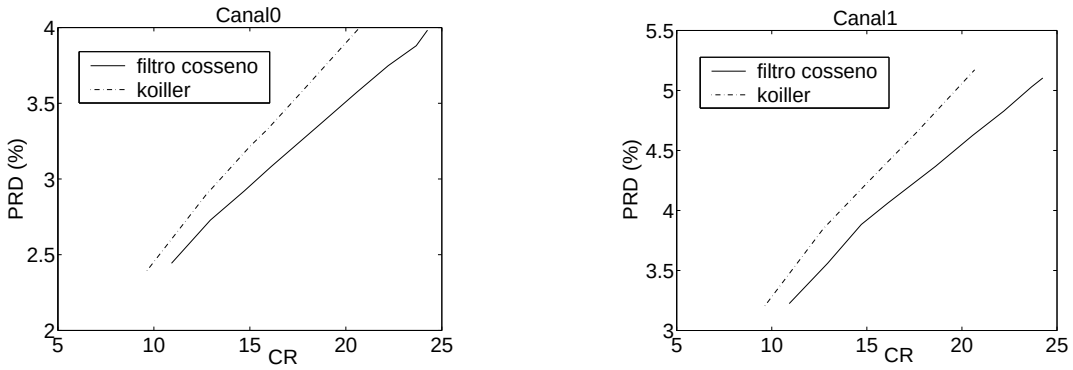


(a) (b)

Figura 5.14: Curva Taxa-Distorção para o sinal 100: (a) Canal0, (b) Canal1

Tabela 5.9: 101.dat

CR	**PRD0**	**PRD1**
10.922	2.443	3.224
12.945	2.729	3.560
14.709	2.921	3.883
16.099	3.080	4.061
18.623	3.357	4.369
20.555	3.571	4.624
22.203	3.748	4.828
23.669	3.880	5.030
24.256	3.983	5.104

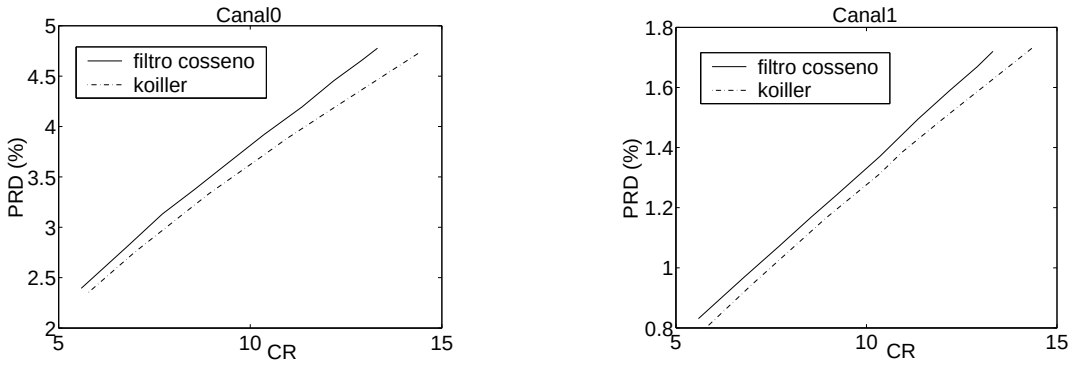


(a) (b)

Figura 5.15: Curva Taxa-Distorção para o sinal 101: (a) Canal0, (b) Canal1

Tabela 5.10: 102.dat

CR	PRD0	PRD1
5.591	2.395	0.831
6.782	2.806	0.968
7.708	3.132	1.072
8.549	3.376	1.168
9.206	3.574	1.241
10.354	3.918	1.371
11.354	4.195	1.495
12.179	4.454	1.589
12.910	4.657	1.669
13.320	4.778	1.720

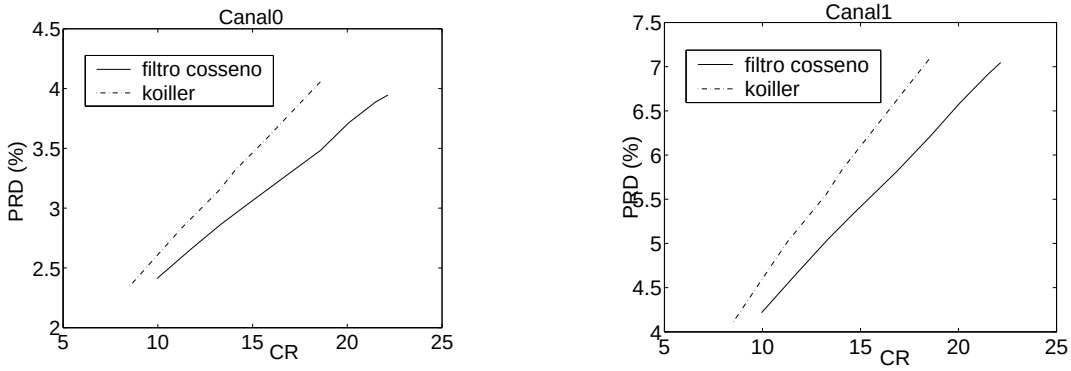


(a) (b)

Figura 5.16: Curva Taxa-Distorção para o sinal 102: (a) Canal0, (b) Canal1

Tabela 5.11: 103.dat

CR	PRD0	PRD1
9.958	2.411	4.217
11.788	2.662	4.670
13.321	2.864	5.043
14.576	3.013	5.323
16.874	3.282	5.819
18.587	3.483	6.219
20.080	3.714	6.591
21.501	3.889	6.914
22.154	3.946	7.047

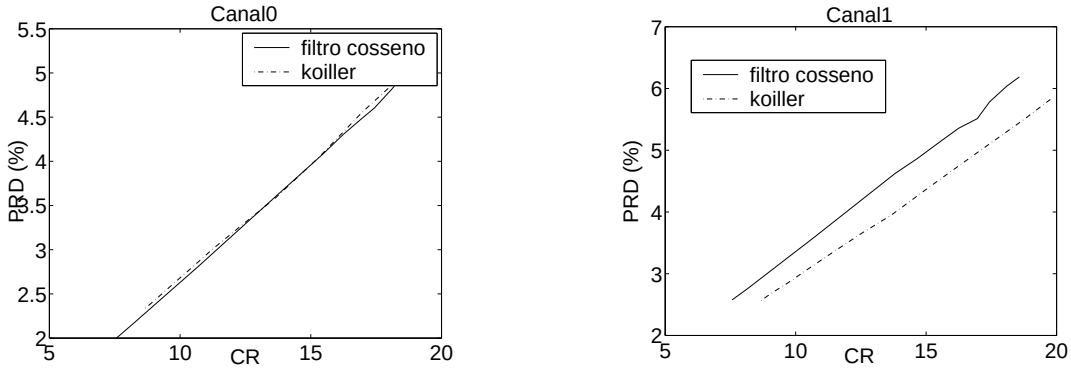


(a) (b)

Figura 5.17: Curva Taxa-Distorção para o sinal 103: (a) Canal0, (b) Canal1

Tabela 5.12: 107.dat

CR	PRD0	PRD1
7.565	2.000	2.576
8.147	2.148	2.755
10.713	2.815	3.590
13.807	3.642	4.622
14.678	3.874	4.869
15.479	4.087	5.121
16.250	4.306	5.357
16.961	4.486	5.511
17.436	4.604	5.784
18.099	4.817	6.040
18.572	4.986	6.187

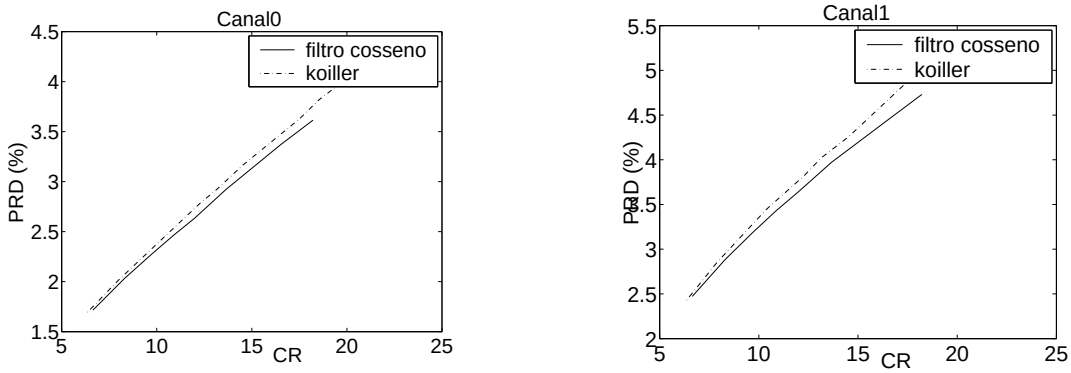


(a) (b)

Figura 5.18: Curva Taxa-Distorção para o sinal 107: (a) Canal0, (b) Canal1

Tabela 5.13: 109.dat

CR	PRD0	PRD1
6.649	1.716	2.473
8.310	2.032	2.886
9.734	2.274	3.195
10.894	2.464	3.434
11.970	2.630	3.636
13.657	2.924	3.971
16.536	3.372	4.451
18.220	3.615	4.731

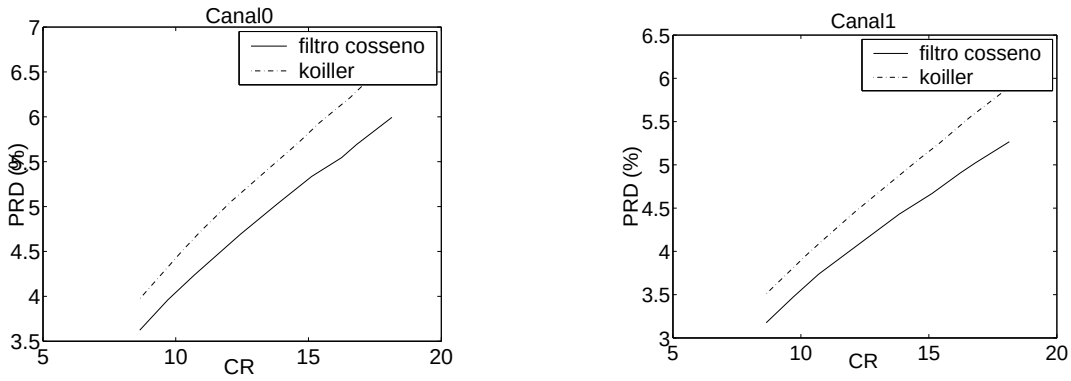


(a) (b)

Figura 5.19: Curva Taxa-Distorção para o sinal 109: (a) Canal0, (b) Canal1

Tabela 5.14: 111.dat

CR	PRD0	PRD1
8.645	3.623	3.175
9.699	3.960	3.472
10.695	4.236	3.736
12.437	4.691	4.122
13.846	5.035	4.431
15.124	5.336	4.669
16.247	5.545	4.910
16.811	5.691	5.021
18.141	5.994	5.267

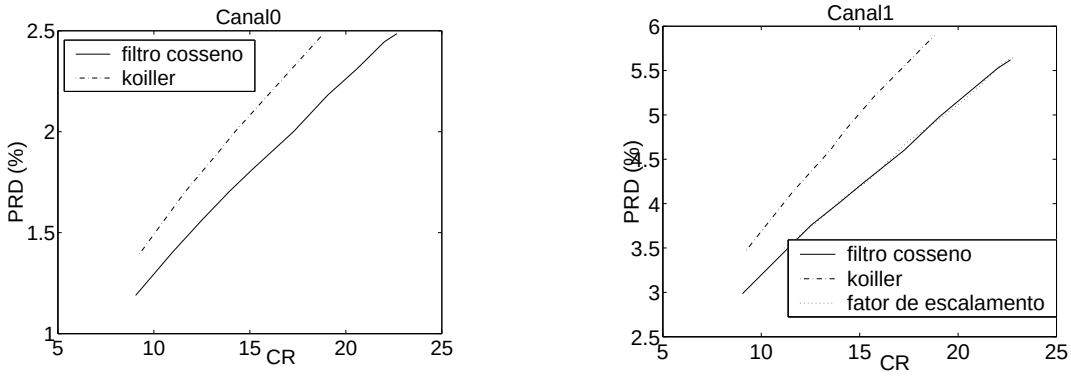


(a) (b)

Figura 5.20: Curva Taxa-Distorção para o 111: (a) Canal0, (b) Canal1

Tabela 5.15: 115.dat

CR	PRD0	PRD1
9.049	1.189	2.986
10.975	1.404	3.412
12.534	1.565	3.757
13.983	1.709	4.013
15.147	1.814	4.227
17.280	2.001	4.604
19.044	2.180	4.977
20.590	2.312	5.263
22.015	2.446	5.527
22.666	2.486	5.619

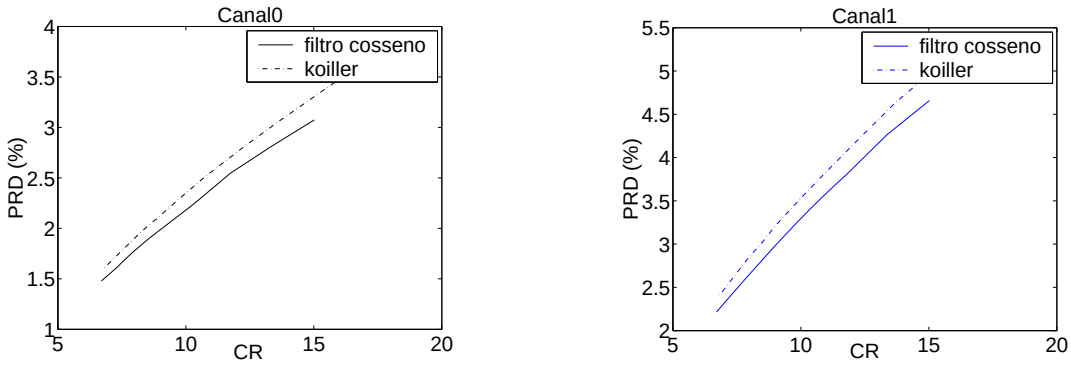


(a) (b)

Figura 5.21: Curva Taxa-Distorção para o 115: (a) Canal0, (b) Canal1

Tabela 5.16: 118.dat

CR	PRD0	PRD1
6.702	1.479	2.214
7.327	1.617	2.429
7.877	1.752	2.611
8.403	1.865	2.784
8.611	1.912	2.861
9.799	2.140	3.237
10.789	2.344	3.527
11.748	2.548	3.794
13.367	2.816	4.264
15.023	3.075	4.658

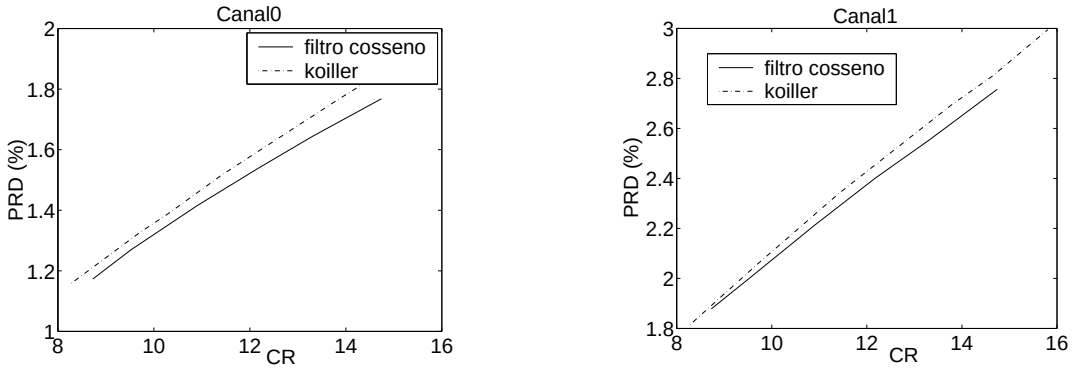


(a) (b)

Figura 5.22: Curva Taxa-Distorção para o 118: (a) Canal0, (b) Canal1

Tabela 5.17: 119.dat - B = 3*Tamanho do bloco = 192

CR	PRD0	PRD1
8.728	1.173	1.878
9.518	1.269	1.999
10.907	1.415	2.213
12.176	1.539	2.401
13.310	1.644	2.553
14.742	1.768	2.756



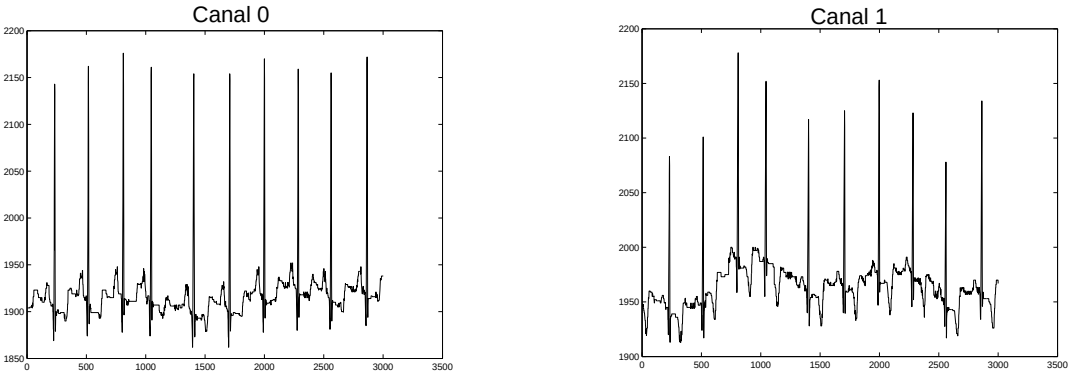
(a) (b)

Figura 5.23: Curva Taxa-Distorção para o 119: (a) Canal0, (b) Canal1

Podemos ver pelos resultados obtidos que, para a maioria dos sinais utilizados, a eficiência do MMP aumentou quando aplicamos a ele o sinal sem o nível DC, que era transmitido utilizando-se o codificador aritmético.

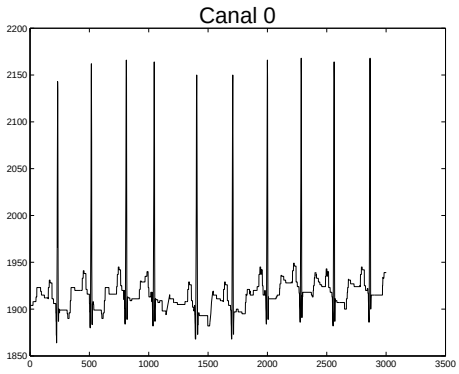
Quando o nível DC é removido, o sinal torna-se mais redundante, e isso faz com que o dicionário de deslocamentos seja utilizado com mais frequência. Além disso, um determinado índice i* passa a ser mais utilizado também, aumentando a eficiência da codificação já que o modelo adaptativo do codificador aritmético atribui probabilidades cada vez mais altas ao índice i*, de forma que menos bits são necessários para codificá-lo.

Apresenteremos alguns sinais resultantes da decodificação do arquivo gerado pelo MMP.

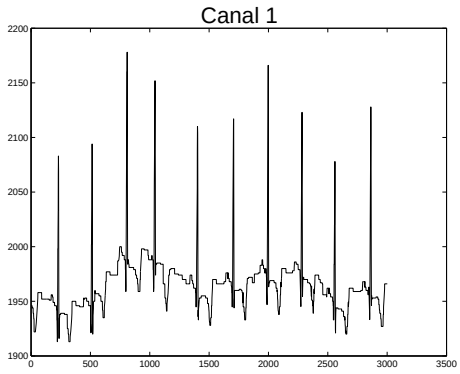


(a) (b)

Figura 5.24: Sinal 100 com CR = 24.256: (a) Canal0 - PRD = 3.983%, (b) Canal1 - PRD = 5.104%

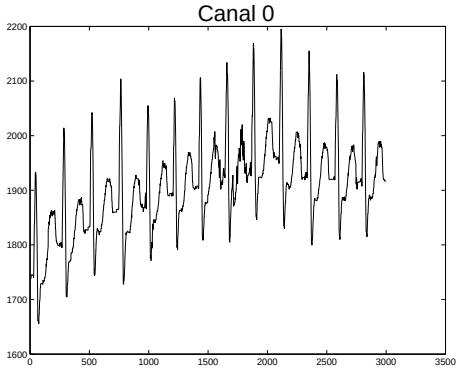


(a)

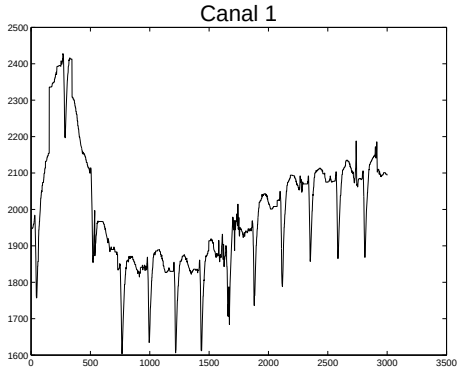


(b)

Figura 5.25: Sinal 100 com CR = 10.922: (a) Canal0 - PRD = 2.443%, (b) Canal1 - PRD = 3.224%

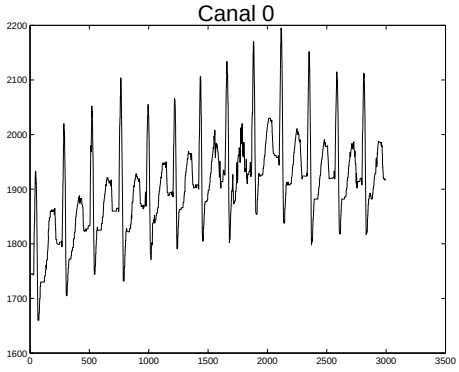


(a)

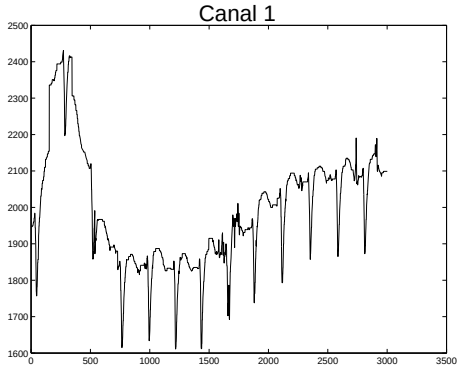


(b)

Figura 5.26: Sinal 109 com CR = 18.220: (a) Canal0 - PRD = 3.615%, (b) Canal1 - PRD = 4.731%



(a)



(b)

Figura 5.27: Sinal 109 com CR = 9.734: (a) Canal0 - PRD = 2.274%, (b) Canal1 - PRD = 3.195%

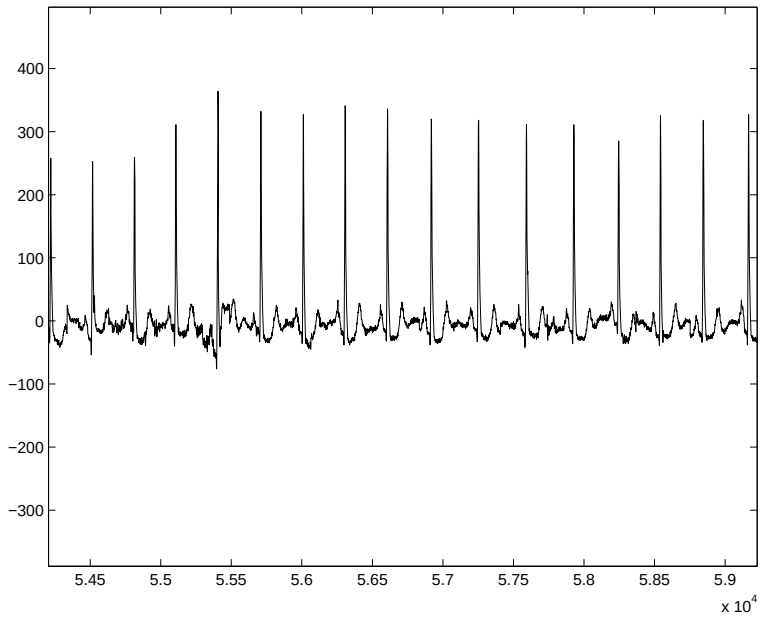


Figura 5.28: Trecho do sinal 101 sem o nível DC

5.3 Equalizando os Picos

Vimos na seção anterior como aumentamos o desempenho do algoritmo para sinais de ECG retirando o nível DC do sinal. Ainda assim, podemos observar nos sinais de ECG sem o nível DC (Figura 5.28) que existe uma variação na altura dos picos. Assim, se fosse possível equalizar os picos do sinal, este seria mais "periódico"e consequentemente, a codificação seria mais eficiente.

Uma forma de equalizar os picos seria utilizando um fator de escalamento, isto é, a altura de um dos picos seria utilizada como referência e os outros picos seriam escalados para que tivessem um valor igual ao valor de referência. O sinal resultante seria aplicado ao MMP e os fatores de escalamento seriam transmitidos utilizando o codificador aritmético, após etapa de transformação e quantização, como fizemos com o nível DC. Essa equalização do sinal é feita no sinal já sem o nível DC.

Como a ocorrência dos picos é quase periódica, dividimos o sinal em blocos de tamanho igual ao período estimado do sinal. Optamos por fazer uma estimativa simples do período do sinal, apenas verificando a distância entre dois picos e tomando o valor encontrado como o valor do período.

Observando o sinal de ECG verifica-se que ele possui picos superiores e picos inferiores, em relação à média do sinal, como mostra a Figura 5.29 .

Para fazer a estimativa do período do sinal, foi necessário primeiro decidir se usaríamos

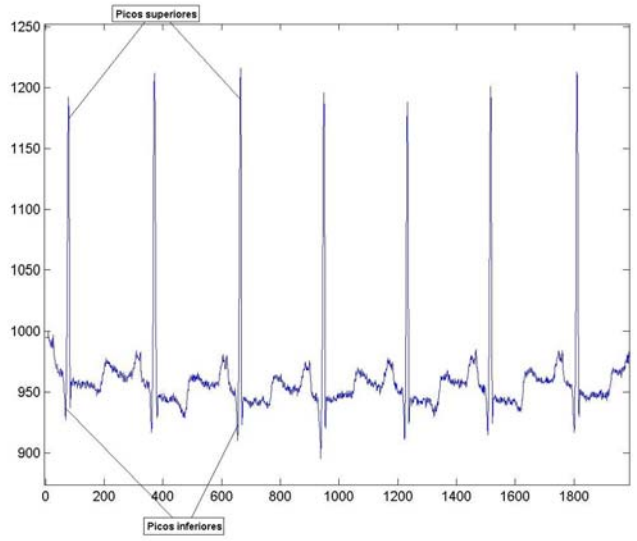


Figura 5.29: Neste trecho do sinal 100.dat podemos observar os picos superiores e os picos inferiores

os picos superiores ou os picos inferiores (em relação à média do sinal). Para isso, verificamos se:

$$X_{max} - \bar{X} > \bar{X} - X_{min}, \tag{5.4}$$

onde X_{max} é o valor máximo do sinal, $\bar{X}$ é a média e X_{min} é o valor mínimo do sinal. Esses valores foram calculados em uma janela de 1000 amostras do sinal. Esse cálculo é aplicado ao sinal original (ainda com o nível DC).

Caso fosse maior, considerávamos os picos superiores para cálculo do período. Caso contrário, considerávamos os picos inferiores.

Assim, conhecendo o valor máximo do sinal (X_{max}) na janela de 1000 amostras e o índice onde se encontra esse máximo (imax), para calcular o período usando os picos superiores fizemos:

```
patamar = X_max − (20/100)(X_max − X_min)
for(int j = imax + 25 ; j < imax + 25 + 720 ; j++){
    if (X[j] > patamar){
        while(X[j+1] − X[j] > 0){
            j++;
        }
        indice = j+1;
        break;
    }
}
T = indice-imax;
```

De forma análoga, para obter o período usando os picos inferiores, conhecendo o valor mínimo do sinal (X_{min}) na janela de 1000 amostrras e o índice onde se encontra esse mínimo (imin), fizemos:

```
patamar = X_min + (20/100)(X_max − X_min)
for(int j = imin + 25 ; j < imin + 25 + 720 ; j++){
    if (X[j] < patamar){
        while(X[j+1] − X[j] < 0){
            j++;
        }
        indice = j+1;
        break;
    }
}
T = indice-imin;
```

Esses valores (25 e 720) e também o patamar foram obtidos empiricamente, utilizando os sinais que tínhamos disponíveis e utilizaríamos para os testes. Na Figura 5.30 o período

é calculado como sendo a diferença entre os índices dos dois últimos picos da janela de 1000 amostras.

Após retirada do nível DC, para cada bloco de T amostras calculamos o valor do pico (o valor máximo dentro do bloco) e tomando o primeiro pico como referência, calculamos os fatores de escalamento para cada bloco i como sendo:

$$fator(i) = \frac{pico_{ref}}{pico(i)}, \tag{5.5}$$

onde $pico_{ref}$ corresponde ao valor da maior amostra do primeiro bloco e $pico(i)$ é o valor máximo de cada bloco i.

O pico de referência e os picos que devem ser ajustados dependem do método utilizado para cálculo do período. Assim, se utilizamos os picos superiores para cálculo do período, os picos superiores serão alinhados, de forma que a parte superior do sinal ficará homogênea, com pouca oscilação. Caso o período tenha sido calculado com base nos picos inferiores, a parte inferior do sinal ficará alinhada, como podemos ver nas Figuras 5.32 e 5.31.

Como o período varia ao longo do sinal, em alguns blocos não havia nenhum pico, produzindo um resultado ruim. Para contornar este problema, optamos por estabelecer um patamar e verificar se o pico estaria acima desse patamar. Caso não estivesse, assumimos que ali haveria um pico de valor igual ao pico do bloco anterior.

Para transmitir os fatores de escalamento codificando-os utilizando o codificador aritmético e permitir que o sinal fosse decodificado, foi necessário quantizar esses fatores.

A transmissão dos fatores de escalamento irá aumentar a taxa, a razão entre o número

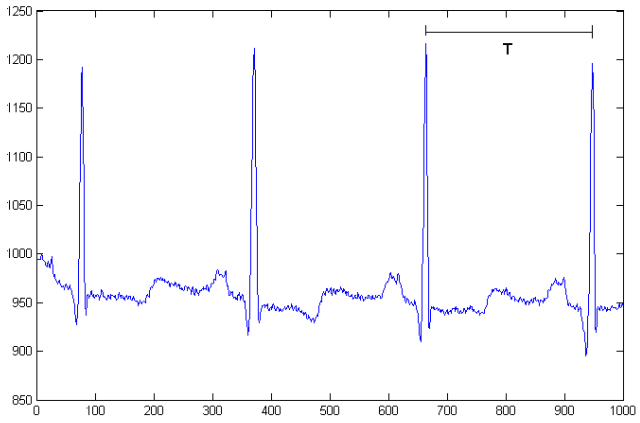


Figura 5.30: Cálculo do periodo para um sinal de ECG

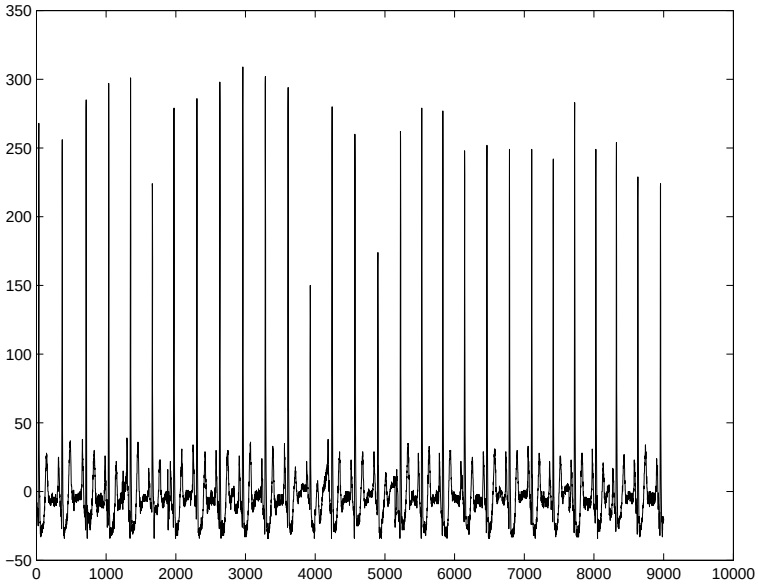


Figura 5.31: Canal0 do sinal 101, com os picos inferiores alinhados

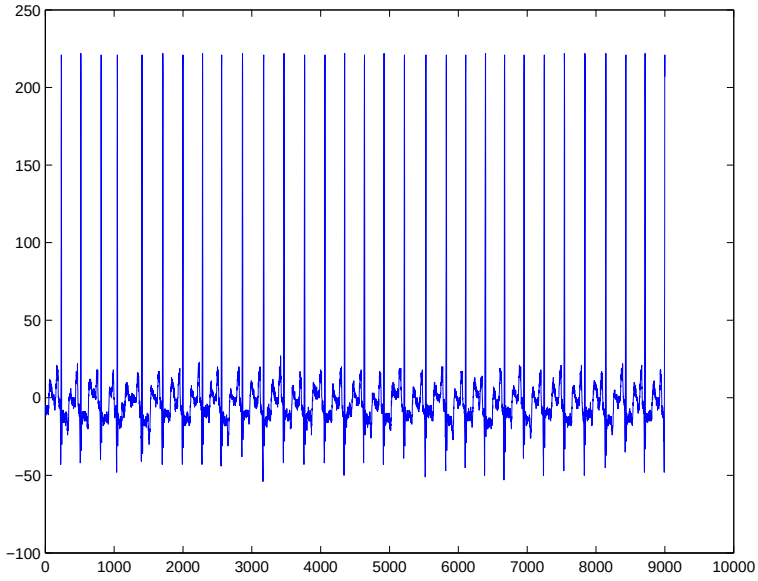


Figura 5.32: Canal0 do sinal 100, com os picos superiores alinhados

de bits do sinal codificado e o número de bits do sinal original. Para que esse método produza resultados acima daqueles obtidos pelo primeiro método proposto, a equalização do sinal deve ser tal que aumente bastante a eficiência do MMP, isto é, reduza significativamente a distorção.

Embora diferentes, os picos de cada bloco não variam muito. Da mesma forma, os fatores de escalamento também apresentam uma pequena variação. Assim, não é necessário utilizar muitos níveis de quantização para esses fatores.

Com o objetivo de encontrar o número de bits ideal para a quantização dos fatores de escalamento, utilizamos o sinal 101.dat, ao qual aplicamos o algoritmo proposto para diferentes fatores de quantização.

5.3.1 Resultados

Apresentaremos os resultadoos obtidos aplicando-se o algoritmo proposto, ao sinal 101.dat, para quatro fatores de quantização diferentes: 4, 6, 8 e 11 bits.

Tabela 5.18: 101.dat - 11 bits

CR	**PRD0**	**PRD1**
8.645	3.623	3.175
9.699	3.960	3.472
10.695	4.236	3.736
12.437	4.691	4.122
13.846	5.035	4.431
15.124	5.336	4.669
16.247	5.545	4.910
16.811	5.691	5.021
18.141	5.994	5.267

Tabela 5.19: 101.dat - 8 bits

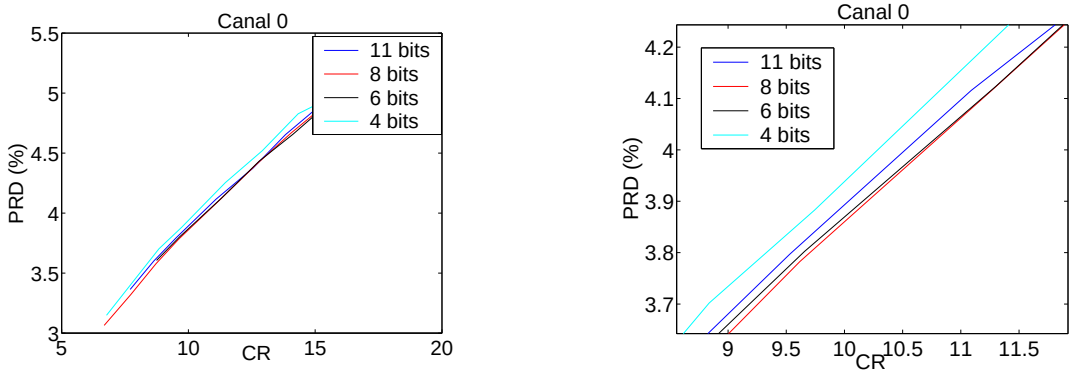
CR	**PRD0**	**PRD1**
6.678	3.064	4.887
7.726	3.320	5.488
8.727	3.579	5.962
9.617	3.783	6.422
11.248	4.112	7.094
12.666	4.402	7.665
14.056	4.669	8.113
15.324	4.889	8.500
15.903	5.027	8.770

Tabela 5.20: 101.dat - 6 bits

CR	**PRD0**	**PRD1**
8.747	3.606	6.064
9.657	3.802	6.439
11.301	4.124	7.160
12.794	4.433	7.722
14.127	4.657	8.211
15.405	4.893	8.589
15.922	4.955	8.780

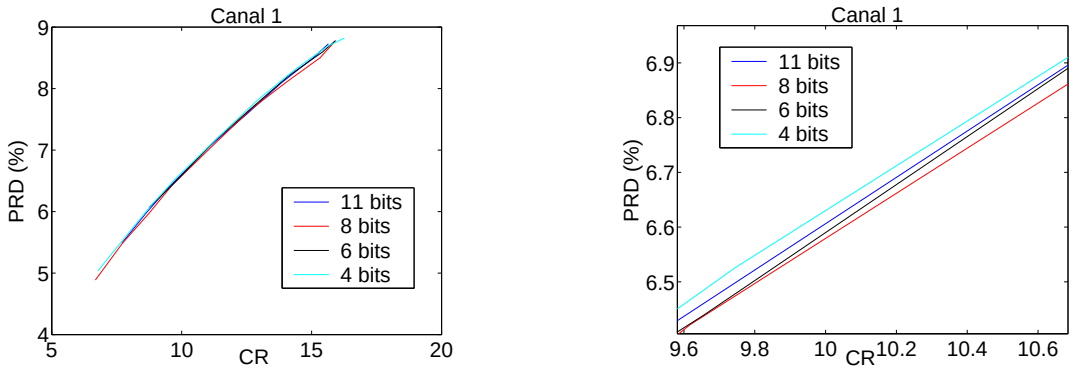
Tabela 5.21: 101.dat - 4 bits

CR	PRD0	PRD1
6.772	3.148	5.034
8.838	3.702	6.107
9.744	3.883	6.525
11.457	4.253	7.226
12.919	4.520	7.813
14.330	4.827	8.298
15.600	4.962	8.686
16.264	5.127	8.821



(a) (b)

Figura 5.33: Comparação entre diferentes fatores de quantização para o canal0 do sinal 101.dat: (a) Curva taxa-distorção , (b) Zoom da curva taxa-distorção



(a) (b)

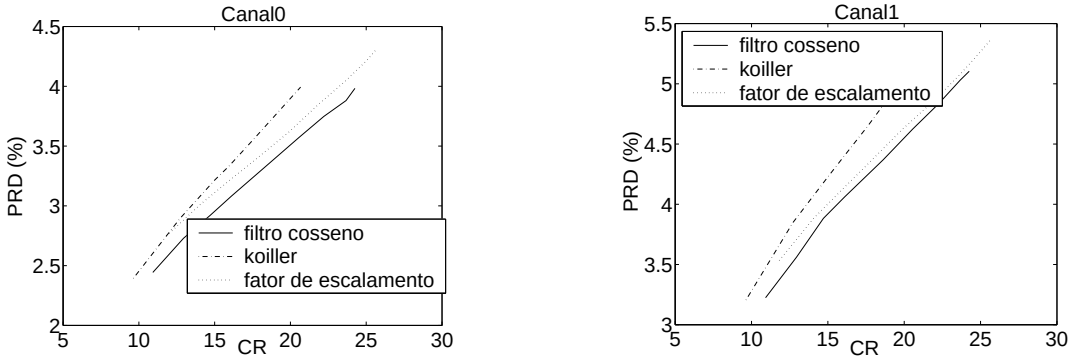
Figura 5.34: Comparação entre diferentes fatores de quantização para o canal1 do sinal 101.dat: (a) Curva taxa-distorção , (b) Zoom da curva taxa-distorção

Os gráficos 5.33 e 5.34 mostram as curvas taxa-distorção para os diferentes fatores aplicados. Podemos ver que a quantização com 8 bits resultou em uma curva mais próxima do eixo horizontal, e irá fornecer uma relação taxa-distorção melhor.

Determinado o melhor fator de quantização, aplicamos o algoritmo proposto para os demais sinais.

Tabela 5.22: 100.dat

CR	**PRD0**	**PRD1**
11.815	2.748	3.533
14.082	3.014	3.885
15.980	3.213	4.128
17.417	3.359	4.308
19.858	3.613	4.620
21.916	3.856	4.857
23.635	4.044	5.075
25.040	4.216	5.272
25.687	4.311	5.369

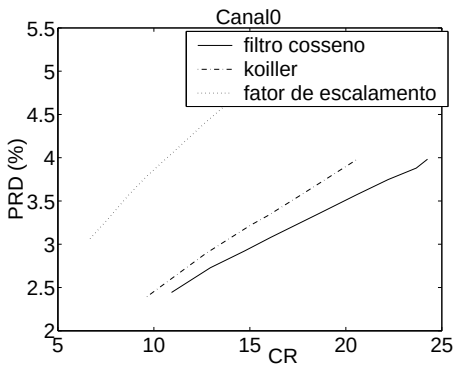


(a) (b)

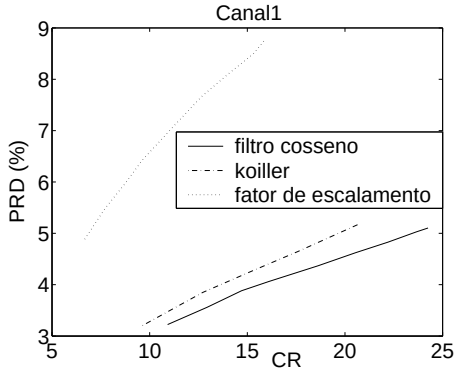
Figura 5.35: Curva taxa-distorção para o canal 100.dat : (a) Canal0 , (b) Canal1

Tabela 5.23: 101.dat

CR	PRD0	PRD1
6.678	3.064	4.887
7.726	3.320	5.488
8.727	3.579	5.962
9.617	3.783	6.422
11.248	4.112	7.094
12.666	4.402	7.665
14.056	4.669	8.113
15.324	4.889	8.500
15.903	5.027	8.770



(a)

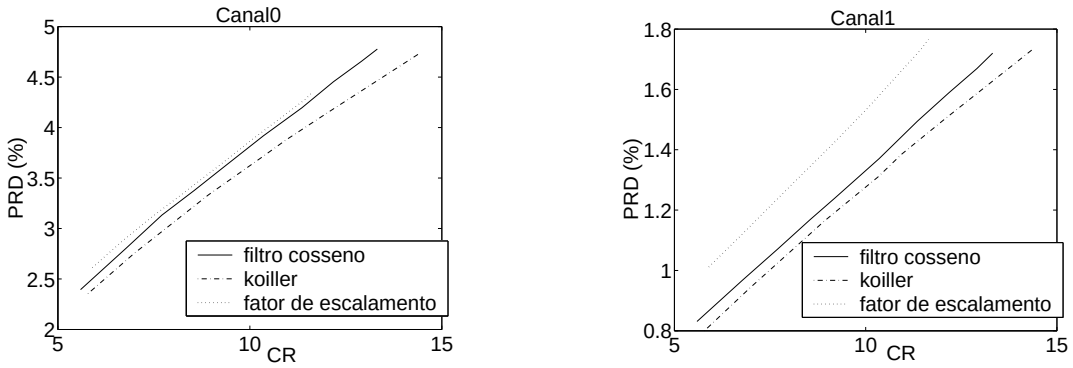


(b)

Figura 5.36: Curva taxa-distorção para o canal 101.dat : (a) Canal0 , (b) Canal1

Tabela 5.24: 102.dat

CR	PRD0	PRD1
5.591	2.395	0.831
6.782	2.806	0.968
7.708	3.132	1.072
8.549	3.376	1.168
9.206	3.574	1.241
10.354	3.918	1.371
11.354	4.195	1.495
12.179	4.454	1.589
12.910	4.657	1.669
13.320	4.778	1.720

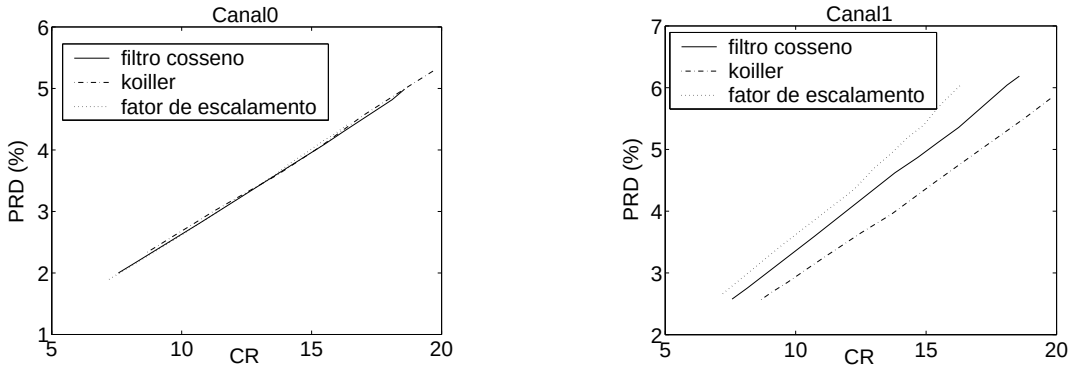


(a) (b)

Figura 5.37: Curva taxa-distorção para o canal 102.dat : (a) Canal0 , (b) Canal1

Tabela 5.25: 107.dat

CR	PRD0	PRD1
7.565	2.000	2.576
8.147	2.148	2.755
10.713	2.815	3.590
13.807	3.642	4.622
14.678	3.874	4.869
15.479	4.087	5.121
16.250	4.306	5.357
16.961	4.486	5.511
17.436	4.604	5.784
18.099	4.817	6.040
18.572	4.986	6.187

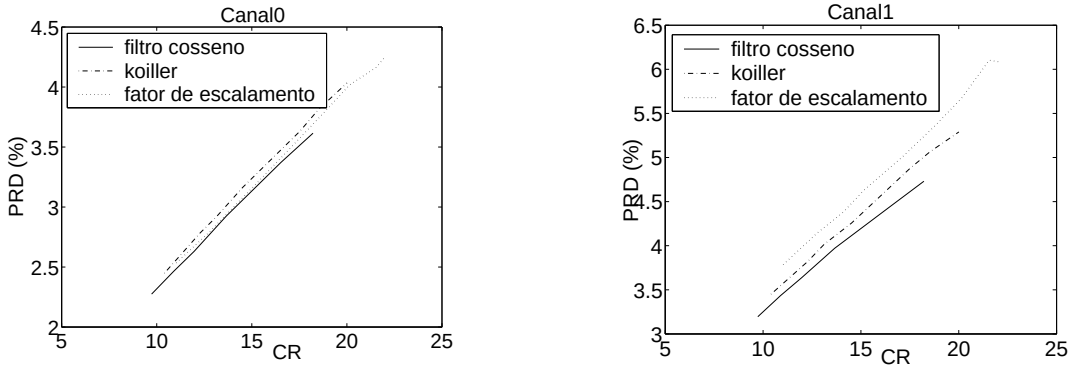


(a) (b)

Figura 5.38: Curva taxa-distorção para o canal 107.dat : (a) Canal0 , (b) Canal1

Tabela 5.26: 109.dat

CR	**PRD0**	**PRD1**
6.649	1.716	2.473
8.310	2.032	2.886
9.734	2.274	3.195
10.894	2.464	3.434
11.970	2.630	3.636
13.657	2.924	3.971
16.536	3.372	4.451
18.220	3.615	4.731

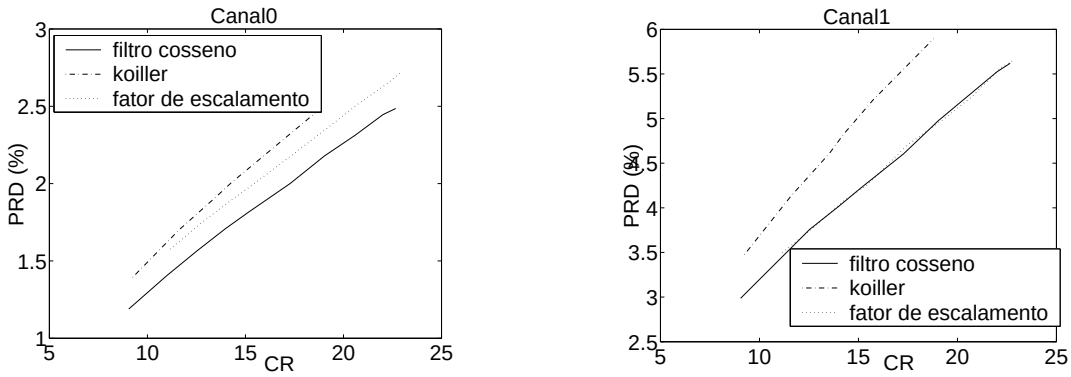


(a) (b)

Figura 5.39: Curva taxa-distorção para o canal 109.dat : (a) Canal0 , (b) Canal1

Tabela 5.27: 115.dat

CR	PRD0	PRD1
9.049	1.189	2.986
10.975	1.404	3.412
12.534	1.565	3.757
13.983	1.709	4.013
15.147	1.814	4.227
17.280	2.001	4.604
19.044	2.180	4.977
20.590	2.312	5.263
22.015	2.446	5.527
22.666	2.486	5.619

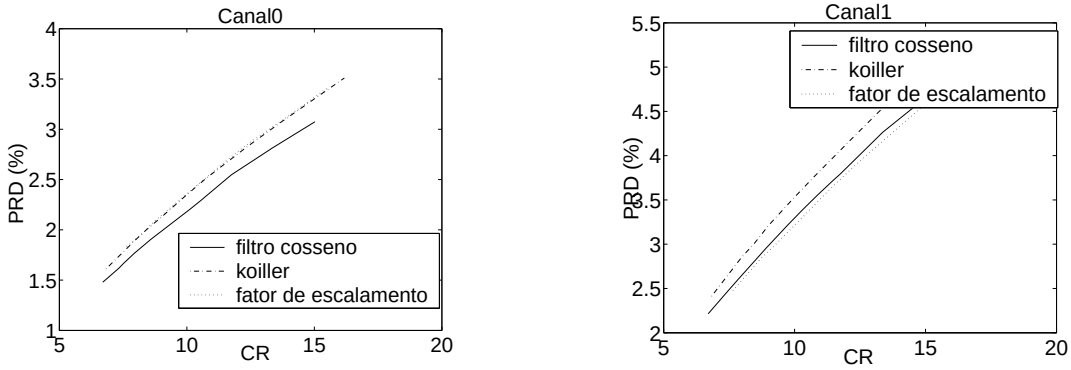


(a) (b)

Figura 5.40: Curva taxa-distorção para o canal 115.dat : (a) Canal0 , (b) Canal1

Tabela 5.28: 118.dat

CR	PRD0	PRD1
6.702	1.479	2.214
7.327	1.617	2.429
7.877	1.752	2.611
8.403	1.865	2.784
8.611	1.912	2.861
9.799	2.140	3.237
10.789	2.344	3.527
11.748	2.548	3.794
13.367	2.816	4.264
15.023	3.075	4.658

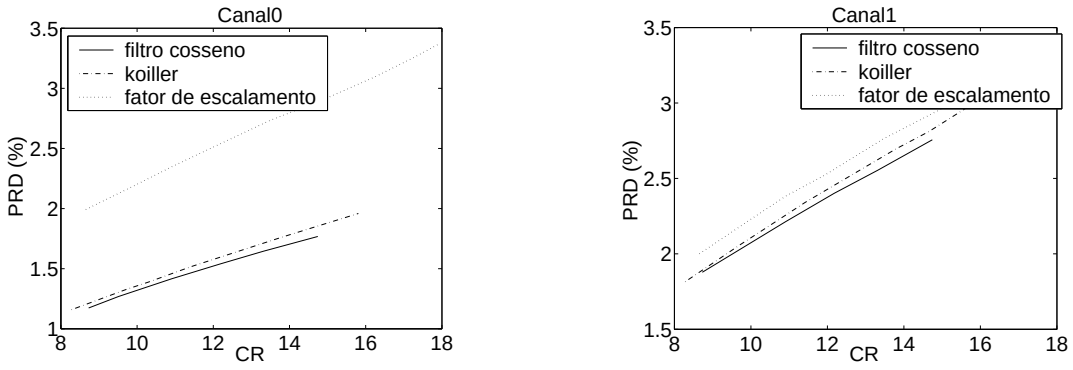


(a) (b)

Figura 5.41: Curva taxa-distorção para o canal 118.dat : (a) Canal0 , (b) Canal1

Tabela 5.29: 119.dat

CR	PRD0	PRD1
8.728	1.173	1.878
9.518	1.269	1.999
10.907	1.415	2.213
12.176	1.539	2.401
13.310	1.644	2.553
14.742	1.768	2.756



(a) (b)

Figura 5.42: Curva taxa-distorção para o canal 119.dat : (a) Canal0 , (b) Canal1

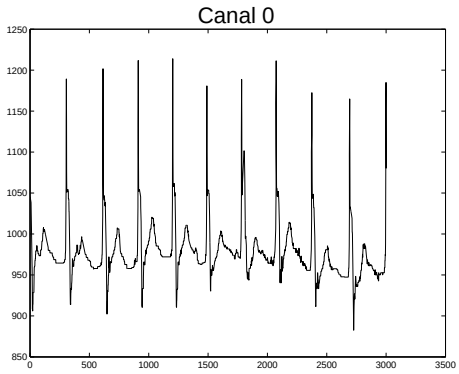
Os resultados obtidos mostram que a equalização dos picos do sinal apesar de deixá-lo mais redundante, não produziu resultados melhores. Para praticamente todos os sinais utilizados para teste a curva taxa-distorção ficou acima da curva obtida no MMP aplicado ao sinal somente sem o nível DC. Em alguns casos, conseguimos uma melhora em relação ao MMP simples implementado por [2]. A única exceção foi para o canal 1 do sinal 118.

O aumento da taxa pela transmissão dos fatores de escalamento não foi compensada por uma grande redução na distorção, como esperávamos que ocorresse quando aplicássemos um sinal mais redundante ao MMP.

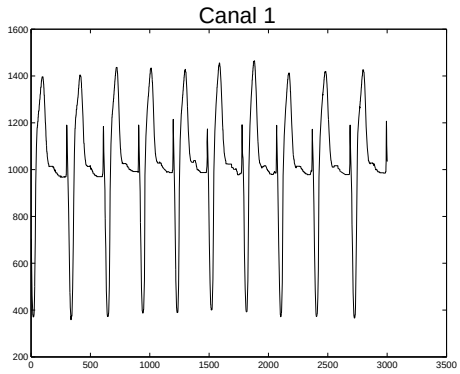
Certamente a variação entre a altura dos picos, que consideramos como ponto a ser melhorado, não era tão grande que reduzisse a eficiência do Dicionário de Deslocamentos. Assim, a equalização implementada implicou em um aumento da taxa (mais bits para serem transmitidos), mas não alterou o comportamento do Dicionário de Deslocamentos, ao contrário do que havíamos previsto.

É possível que em sinais que variem mais esse algoritmo apresente resultados melhores.

Apresenteremos alguns sinais resultantes da decodificação do arquivo gerado pelo MMP.

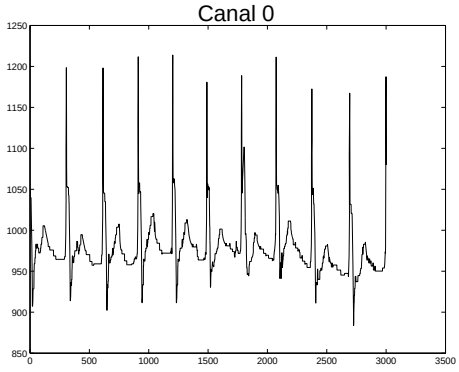


(a)

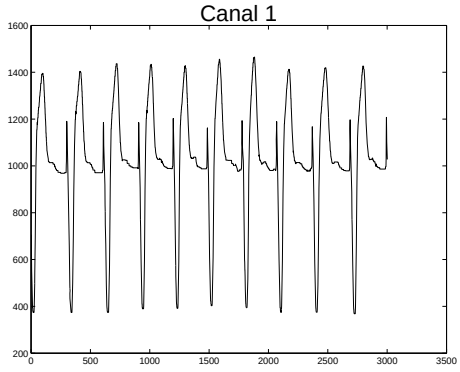


(b)

Figura 5.43: Sinal 102 com CR = 12.179: (a) Canal0 - PRD = 4.454%, (b) Canal1 - PRD = 1.589%

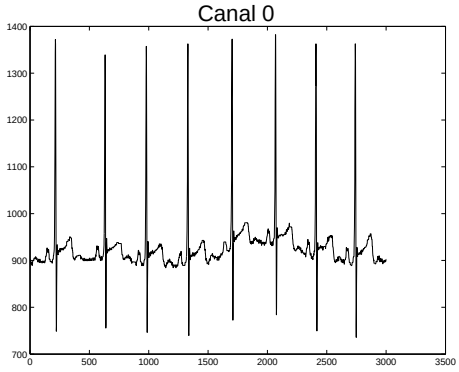


(a)

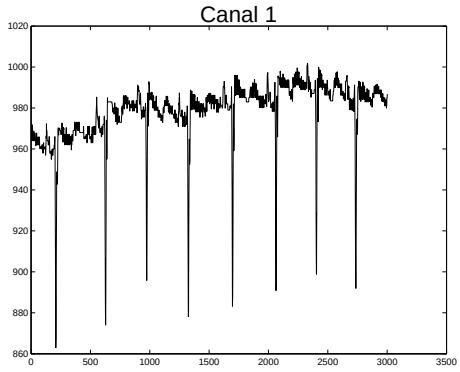


(b)

Figura 5.44: Sinal 102 com CR = 8.549: (a) Canal0 - PRD = 3.376%, (b) Canal1 - PRD = 1.168%

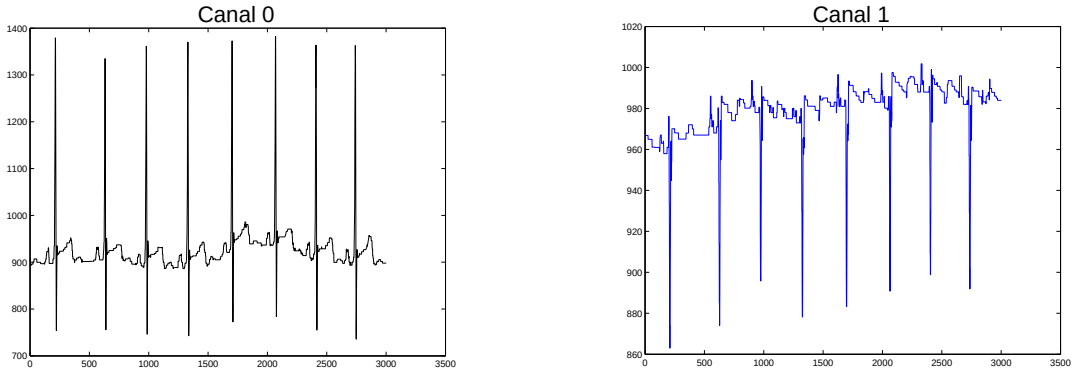


(a)



(b)

Figura 5.45: Sinal 115 com CR = 22.015: (a) Canal0 - PRD = 2.446%, (b) Canal1 - PRD = 5.527%



(a) (b)

Figura 5.46: Sinal 115 com CR = 9.049: (a) Canal0 - PRD = 1.189%, (b) Canal1 - PRD = 2.986%

Capítulo 6

Conclusão

O trabalho descreveu o algoritmo MMP, em diversas versões. Desde a mais simples, desenvolvida com o intuito de comprimir imagens até as alterações que implementamos a fim de tornar o MMP mais adequado para compressão de sinais de eletrocardiograma.

O MMP é um algoritmo bastante complexo, baseado na recorrência de padrões, em diversas escalas, de um sinal. Enquanto codifica o sinal, o MMP cria um dicionário com versões contraídas, dilatadas e concatenadas dos padrões previamente codificados.

O trabalho desenvolvido envolveu toda a implementação do MMP. Começamos com o MMP simples, depois implementamos o MMP com otimização taxa-distorção e em seguida implementamos o Dicionário de Deslocamentos, proposto por [2]. A partir desta base, implementamos as alterações propostas no Capítulo 5. Todo o sistema foi desenvolvido utilizando uma linguagem orientada objeto - C++.

A primeira idéia foi tratar o sinal de ECG como imagem, uma vez que o MMP era considerado um método bastante eficiente de comprimir imagens. O Capítulo 5 mostrou que os resultados obtidos não comprovaram nossas expectativas. Apesar de comprimir com sucesso o sinal de ECG, a compressão obtida não superou os resultados de [2]. Certamente, o sinal de ECG tratado como sinal unidimensional apresenta características que são melhor aproveitadas pelo MMP.

Em seguida decidimos tratar o sinal de ECG em sua forma original (sinal unidimensional), fazendo modificações que o deixasse mais adequado para ser aplicado ao MMP.

Como foi apresentado no projeto, quando busca-se uma representação para um trecho do sinal, utiliza-se também o Dicionário de Deslocamentos, que possui uma sequência de 16M amostras no sinal codificado (onde M é o tamanho do bloco usado para codificação). Caso

sinais periódicos fossem comprimidos utilizando o MMP com Dicionário de Deslocamentos, seria obtido alto desempenho de compressão, uma vez que um determinado índice i* seria codificado repetidamente para representar grandes blocos do sinal.

O sinal de ECG assemelha-se a um sinal periódico e portanto, o Dicionário de Deslocamento também é utilizado com sucesso. Se o sinal de ECG fosse mais redundante, isto é, se os blocos fosse mais parecidos, o Dicionário de Deslocamentos seria melhor utilizado.

A primeira tentativa de aprimorar o algoritmo desenvolvido por [2], aproveitando melhor o Dicionário de Deslocamentos surgiu após verificarmos que o nível DC do sinal de ECG varia ao longo do tempo. Assim, implementamos um algoritmo para retirar o nível DC do sinal e então aplicar o MMP ao sinal sem o nível DC. Os valores do nível DC foram transmitidos após etapa de transformação, quantização e codificação. Para esta última estapa utilizamos o codificador aritmético, com um modelo adaptativo à parte para estimar as probabilidade de ocorrência dos valores quantizados.

Apesar da necessidade de transmitir mais bits, o sinal se tornou mais redundante, levando o Dicionário de Deslocamentos a ser melhor aproveitado. Vimos que aplicar o MMP ao sinal sem o nível DC, na média, aumentou a eficiência do codificador, obtendo resultados melhores que [2].

Apesar dos bons resultados, acreditamos ainda ser possível aumentar mais o desempenho do MMP. Verificamos que mesmo sem o nível DC havia irregularidade nos picos do sinal, com característica "quase periódica". Implementamos um método para equalizar os picos e transmitir os fatores de escalamento. Os fatores foram codificados utilizando também o codificador aritmético, com outro modelo adaptativo à parte para estimar as probabilidade de ocorrência dos fatores de escalamento quantizados. Os testes revelaram que apesar do sinal ficar mais redundante, a necessidade de transmitir mais bits (fatores de escalamento) não foi compensada por uma redução significativa da distorção (PRD), de forma que para a quase totalidade dos sinais, o resultado foi inferior ao obtido com o primeiro método proposto.

Para aprimorar os algoritmos desenvolvidos seria importante fazer uma análise mais aprofundada da influência dos parâmetros utlizados, como o tamanho de bloco para codificação e o comprimento do filtro usado para estimar o nível DC de cada bloco. A escolha do melhor parâmetro surgiu a partir de testes com um único sinal. Fazer testes com esse objetivo com todos os sinais poderia ser mais eficiente, mas demandaria muito mais tempo, sendo inviável para este trabalho.

Outro ponto que poderia ser melhorado é a estimativa do período do sinal, que foi feita de maneira simples, utilizando métodos empíricos, que têm validade para os sinais que utilizamos para teste e, possivelmente, para sinais com características semelhantes.

Referências Bibliográficas

[1] JALALEDDINE, S. M. S., HUTCHENS, C. G., STRATTAN, R. D., *et al.*, "ECG Data Compression Techniques - A Unified Approach", *IEE Transactions on Biomedical Enginnering*, v. 37, n. 4, Abril 1990.

[2] KOILLER, J., *Codificação de Sinais Eletrocardiográficos Usando Recorrência de Padrões Multiescalas*. Projeto final de graduação, UFRJ, Rio de Janeiro, RJ, Brasil, Abril 2002.

[3] CARVALHO, M. B. D., *Compressão de Sinais Multi-Dimensionais Usando Recorrência de Padrões Multiescalas*. Tese de doutorado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, Abril 2001.

[4] TOBIAS, N. M. M. D. O., MOFFA, P. J., GRUPI, C. J., "Aplicações Clínicas do Eletrocardiograma de Alta Resolução", *Revista da Sociedade Cardiológica do Estado de São Paulo*, , 1999.

[5] CHAGAS, A. V., *Compressão de Sinais Eletrocardiográficos Utilizando Wavelets Biortogonais e Codificação Aritmética Adaptativa*. Tese de mestrado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, Abril 2002.

[6] BELL, T. C., CLEARY, J. G., *Text Compression*. Englewood Cliffs, New Jersey, Prentice Hall, 1990.

[7] SAYOOD, K., *Introduction to Data Compression*. Morgan Kaufman, 1996.

[8] COVER, T. M., THOMAS, J. A., *Elements of Information Theory*. John Wiley & Sons, 1991.

Apêndice A

Princípios da Teoria da Informação

Queremos neste apêndice fazer uma breve introdução à teoria da informação, a fim de que os conceitos trabalhados ao longo deste projeto, que fornecem a base para a compressão de sinais, fiquem claros para o leitor.

A Teoria da Informação teve início em 1948, com um artigo publicado por Shannon, um engenheiro elétrico do Bell Labs.

A idéia fundamental é o conceito de entropia, que é uma medida da ordem ou redundância em uma mensagem. A entropia será pequena quando a mensagem for muito redundante e será grande na situação inversa. A medida da entropia de uma mensagem é dependente da probabilidade dos símbolos desta mensagem. A entropia indica o valor mínimo de bits/amostra que pode ser utilizado para codificar uma mensagem.

A.1 Informação e Entropia

Uma medida da quantidade de informação que uma mensagem carrega é diferente da quantidade de dados necessária para representar esta mensagem. A idéia da compressão de dados é encontrar uma nova representação de uma mensagem de forma que ela carregue a mesma quantidade de informação, porém seja representada utilizando um menor número de *bits*.

A quantidade de informação carregada por uma mensagem está relacionada à probabilidade de ocorrência da mensagem. Quanto menor for a probabilidade de ocorrência da mensagem, maior será a quantidade de informação associada a ela. Da mesma forma, mensagens com probabilidade de ocorrência maior, carregam menos informação. Uma mensagem

que tem 100 % de probabilidade de ocorrer não carrega nenhuma informação.

Suponha que um experimento probabilístico envolve a observação da saída emitida por uma fonte discreta. A saída dessa fonte é uma variável aleatória S, que pode assumir valores de um alfabeto finito $\mathcal{S} = \{s_0, s_1, ..., s_{N-1}\}$, com probabilidades $P(S = s_k) = p_k, k = 0, 1, ..., N-1$. Esse conjunto de probabilidades deve satisfazer à codição:

$$\sum_{k=0}^{N-1} p_k = 1. \tag{A.1}$$

A quantidade de informação após o evento $S = s_k$ ocorrer é:

$$I(s_k) = \log_2(\frac{1}{p_k}) \tag{A.2}$$

A unidade de informação é o *bit*.

A entropia da fonte S é a média da informação de todos os símbolos que ela pode emitir. Assim, podemos calcular a entropia $H(S)$ como:

$$H(S) = \sum_{k=0}^{N-1} p_k I(s_k) \tag{A.3}$$

Shannon mostrou que o número de *bits* médio necessário para codificar a saída de uma fonte não pode ser menor que a entropia. Como já vimos, a compressão sem perdas permite que o sinal reconstruído seja exatamente igual ao sinal original e por isso não é possível obter qualquer taxa de compressão. Isto é, existe um limite além do qual não é possível comprimir sem perdas. Esse limite é dado pela entropia da fonte uma vez que a menor taxa (*bits*/amostra) possível de ser atingida é igual à entropia.

A.2 Teoria Taxa-Distorção

Mostramos acima que o limite para compressão sem perdas é a entropia da fonte. Sistemas de compressão com perdas também têm restrições, que são estudadas na teoria taxa-distorção. Ao contrário do que ocorre quando um sinal é comprimido sem perdas, o sinal não é exatamente igual ao original quando ele é comprimido com perdas, isto é, existe uma distorção no sinal reconstruído.

Na compressão com perdas deseja-se obter a menor taxa e distorção possíveis. No entanto, existe um compromisso entre a taxa e a distorção, que pode ser representado por uma

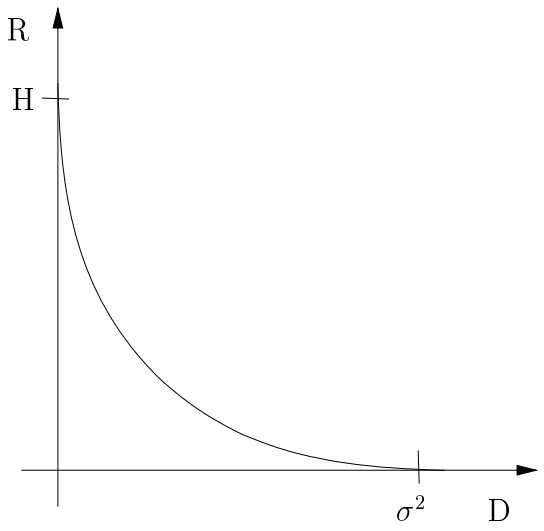


Figura A.1: Curva RD para uma fonte

função R(D). Essa função define a menor taxa R que podemos obter mantendo a distorção menor ou igual a D. É um mínimo teórico definido para cada fonte.

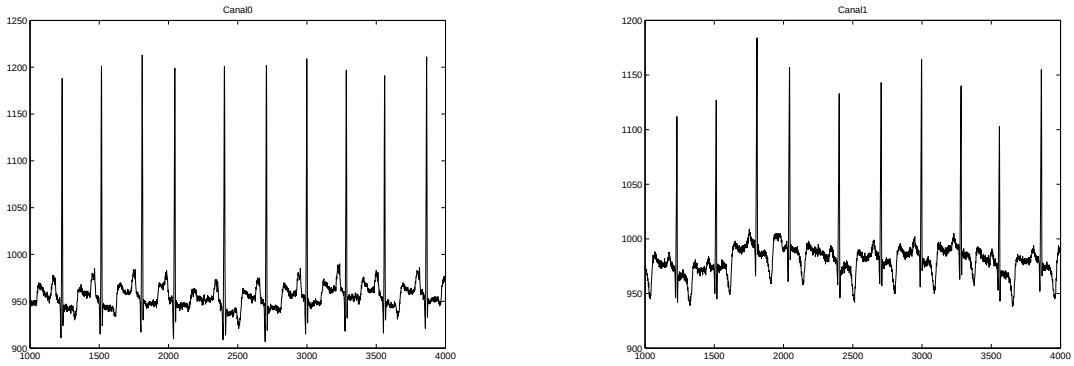
A função R(D) é uma função decrescente e convexa definida no intervalo [0,Dmax], onde Dmax é a menor distorção obtida quando representa-se qualquer saída da fonte com 0 *bits*. Podemos ver na Figura A.1 que quanto maior a distorção, menor será a taxa e quanto menor a distorção, maior a taxa. Quando a distorção for nula, a taxa será igual à entropia da fonte e quando a taxa for nula, a distorção será igual à variância da fonte, como pode ser visto em [7] e em [8].

O que se almeja em compressão é codificar um sinal com uma relação entre taxa e distorção próxima daquela dada por R(D).Isto de um modo geral não é possível. O objetivo das técnicas de compressão é se aproximar o máximo possível da R(D).

Apêndice B

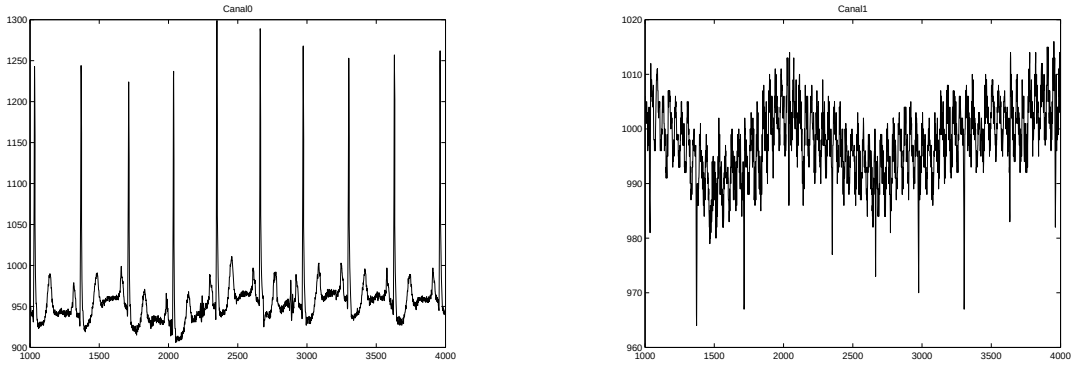
Sinais de ECG Utilizados para Teste

Apresentaremos os sinais, que fazem parte do banco de dados MIT-BIH, utilizados para testes neste trabalho. Cada sinal possui dois canais, correspondentes à duas derivações do ECG. Mostraremos apenas 3000 amostras de cada canal (1000 a 4000).



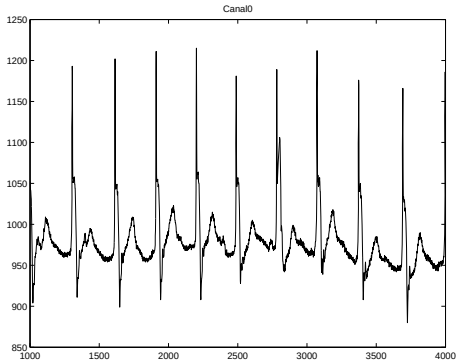
(a) (b)

Figura B.1: Sinais da Arquivo 100.dat: (a) Canal 0, (b) Canal1

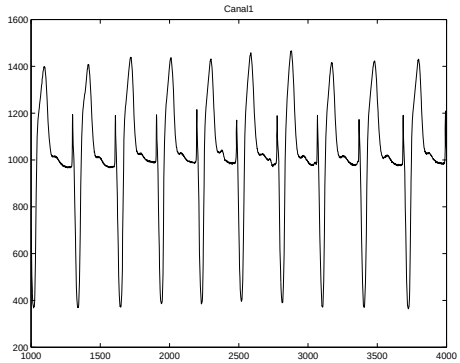


(a) (b)

Figura B.2: Sinais da Arquivo 101.dat: (a) Canal 0, (b) Canal1

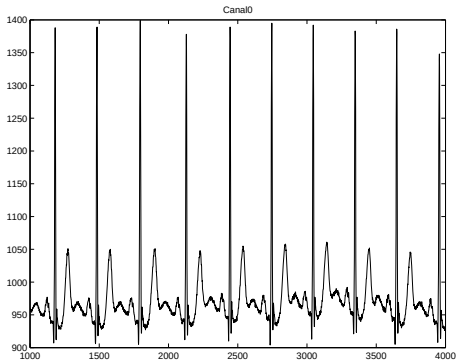


(a)

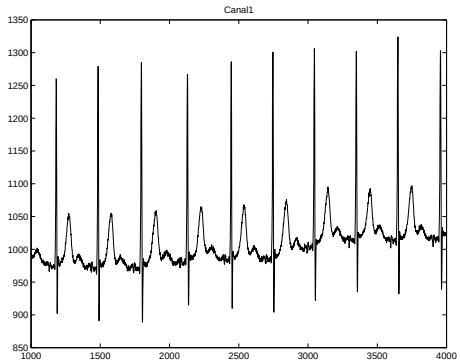


(b)

Figura B.3: Sinais da Arquivo 102.dat: (a) Canal 0, (b) Canal1

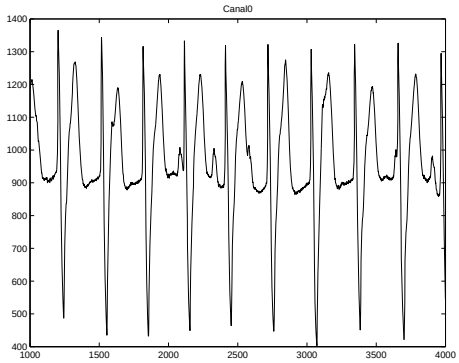


(a)

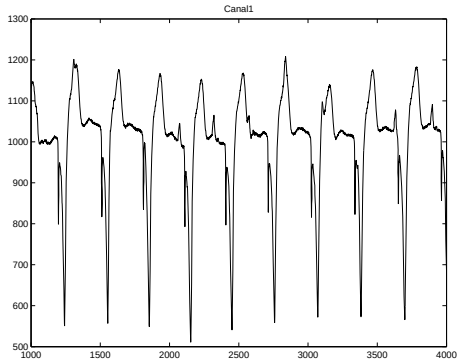


(b)

Figura B.4: Sinais da Arquivo 103.dat: (a) Canal 0, (b) Canal1

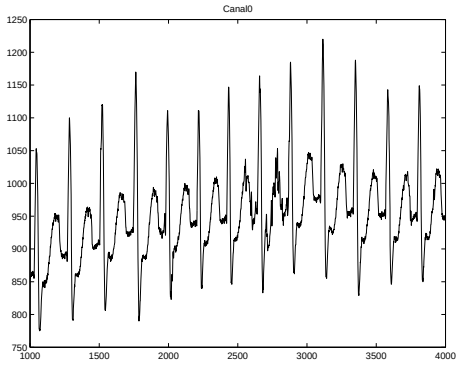


(a)

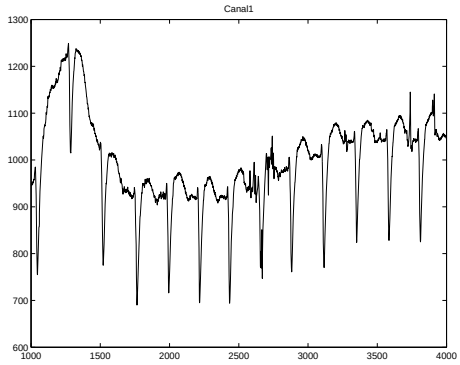


(b)

Figura B.5: Sinais da Arquivo 107.dat: (a) Canal 0, (b) Canal1

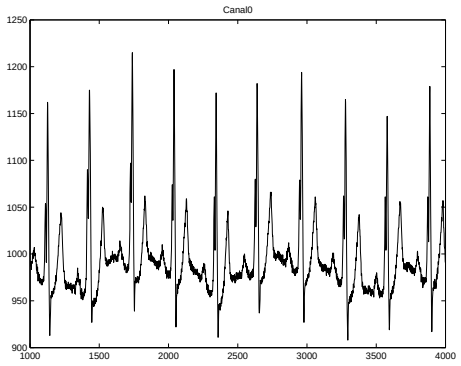


(a)

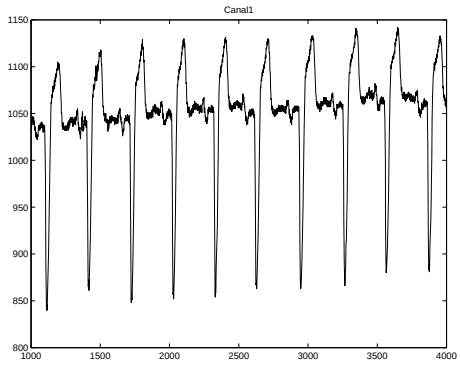


(b)

Figura B.6: Sinais da Arquivo 109.dat: (a) Canal 0, (b) Canal1

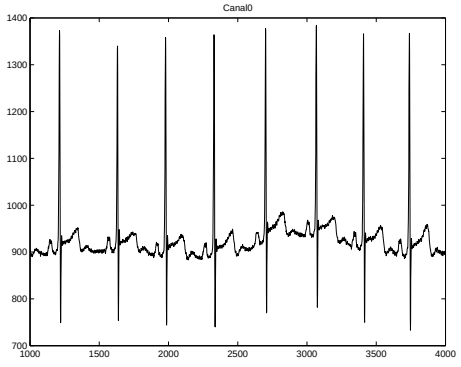


(a)

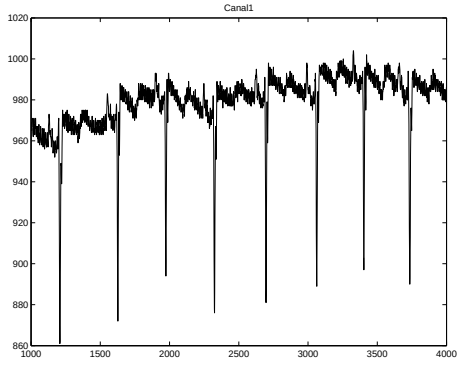


(b)

Figura B.7: Sinais da Arquivo 111.dat: (a) Canal 0, (b) Canal1

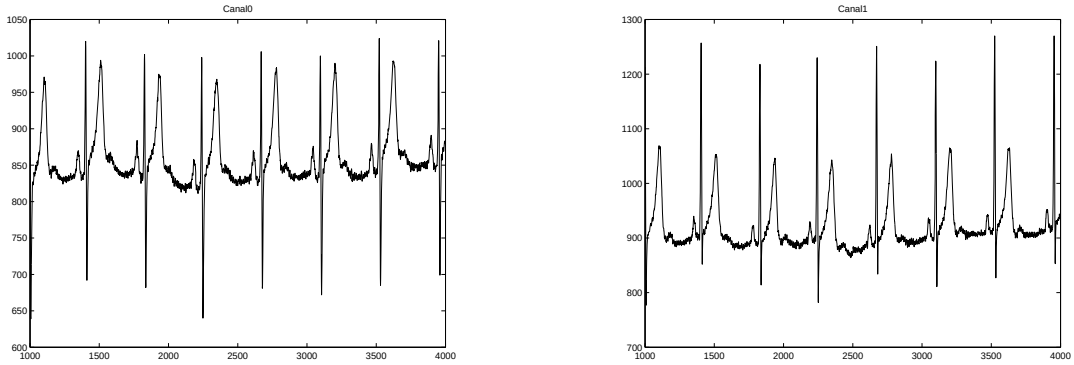


(a)



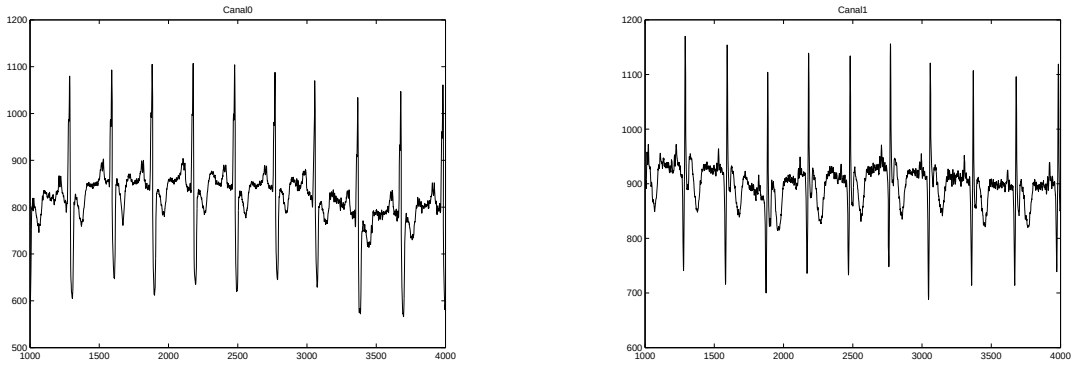
(b)

Figura B.8: Sinais da Arquivo 115.dat: (a) Canal 0, (b) Canal1



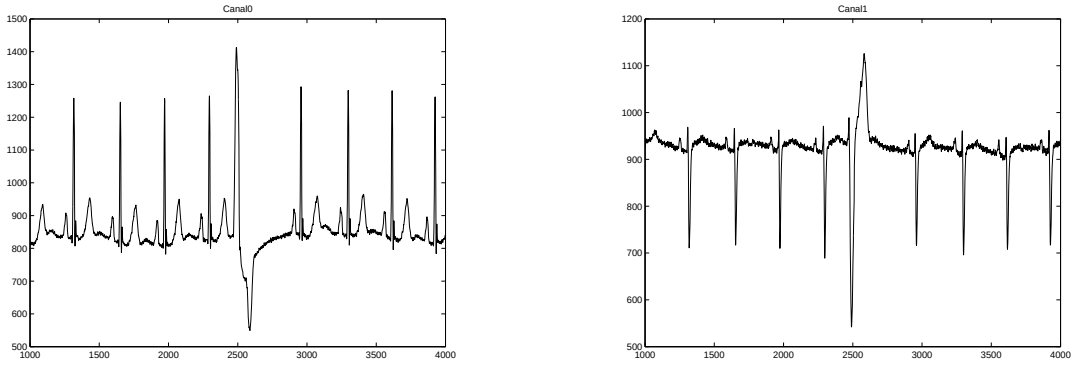
(a) (b)

Figura B.9: Sinais da Arquivo 117.dat: (a) Canal 0, (b) Canal1



(a) (b)

Figura B.10: Sinais da Arquivo 118.dat: (a) Canal 0, (b) Canal1



(a) (b)

Figura B.11: Sinais da Arquivo 119.dat: (a) Canal 0, (b) Canal1