Codec para sinais fisológicos (EMG)

Madalena Silva
Departamento de Engenharia Informática, Universidade de Coimbra

Maria Paula Viegas
Departamento de Engenahria Informática, Universidade de Coimbra

Vitalina Holubenko
Departamento de Engenharia Informática, Universidade de Coimbra

Resumo **– Este trabalho apresenta técnicas unidimensionais sem perdas para a compressão de sinais S-EMG adquiridos durante contrações dinâmicas de modo a reduzir o número de bits necessários para representar a informação, mantendo o máximo de qualidade possível. O sinal primeiro sofre um pré-processamento de modo a aumentar a correlação entre os valores da matriz(que não vai ser aprofundado neste trabalho). De seguida são utilizados vários preditores de forma a diminuir a entropia e aumentar a redundância, de seguida são utilizados codificadores entrópicos. Os preditores que são apresentados neste trabalho são os preditores gerais, como o lpc(linear prediction filter coefficients), levinson, e o delta. Os codificadores entrópicos, por sua vez são, o codificador de Huffman, Golomb Rice, aritmético e o Run Length Encoder.**

Palavras-chave — ***compressão de dados, S-EMG, técnicas unidimensionas, preditores, lpc, linear prediction filter coefficients, levinson, delta, Codificador de Huffman, RLE, Delta, Codificador Aritmético, codificadores entrópicos.***

Introdução

A eletromiografia é utilizada na área da saúde maioritariamente para diagnosticar pacientes que sofram de doenças neuromusculares; este teste pode ser feito ao paciente através de agulhas em contacto direto com o nervo ou de uma forma superficial e não evasiva através de elétrodos, em contacto com a pele (S-EMG). Em ambos, os nervos são estimulados através de pequenos choques elétricos para ver se respondem de forma normal. Na maioria dos casos os dados provenientes de sinais eletromiográficos precisam de ser armazenados ou transmitidos para outros centros hospitalar para serem vistas por um especialista. Para isto são necessárias técnicas para facilitar o armazenamento e a transmissão. Uma maneira de resolver este problema é através de métodos de compressão. Assim, é de interesse o desenvolvimento de técnicas eficientes de codificação que possam apresentar um sinal codificado que seja o mais fiel possível ao sinal original, na menor quantidade de bytes. Há maioritariamente três critérios de avaliação de compressão de dados: a taxa de compressão para avaliar o ganho resultante, a eficiência do algoritmo, ou seja, se a compressão é feita com perdas ou sem perdas e por fim a complexidade do algoritmo. Ao longo do tempo, foram feitos vários estudos sobre métodos de compressão para EMG entre os quais: ZeroTrees, incorporado da transformada de Wavelet feito por Norris (2001) com uma taxa de compressão de 60-95%; modelo autoregressivo (AR) com 95% de compressão feito por Carotti; transformada de Wavelet proposto por Berger (2006,2007); transformada discreta de Wavelet (Brechet em 2007); método de quantização vetorial combinado com Wavelet (Neelu Jain e Vig Renu 2007); otimização de filtros Wavelet (Paiva 2008); a técnica de compressão de sinais de S-EMG baseada em padrões recorrentes, proposta por Filho et al. (2008a), apresentou desempenho bastante satisfatório em termos de ganho de compressão; técnicas que envolvem inteligência artificial relacionadas com a transformada de wavelet levaram a altas taxas de compressão; nos trabalhos mais recentes temos trabalhos de Trabuco (2013) como a compressão de sinais EMG por Transformadas e Perfil Espectral para alocação de bits e a codificação bidimensional.

Neste trabalho vamos explorar estas técnicas mais recentes.

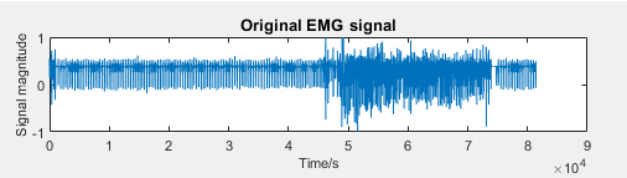


Fig.1 – Sinal EMG a ser comprimido

I. Compressão de sinais

A. Compressão com perdas

Para a compressão com perdas utilizadas em sinais biomédicos temos dois tipos principais de técnicas: os métodos diretos e métodos de compressão por transformada.

A.1.Métodos diretos

Os métodos diretos dependem da morfologia do sinal original, por esta razão os sinais são codificados no domínio do tempo. A maioria destes métodos são complexos e geram pouca eficiência de compressão em comparação com os métodos com as transformadas.

B.1.Transformadas

Os mais viáveis dentre os vários métodos de compressão de sinais, são as técnicas derivadas de transformadas, que alcançam o melhor desempenho em termos de ganho de compressão e de fidelidade ao formato do sinal. Para um dado vetor de dados, é possível determinar uma transformada ortogonal com uma operação linear, seja T uma transformada linear, tal que:

$$y = Tx,$$

onde y é um vetor dos coeficientes da transformada.

A compressão por transformada consiste no processamento da informação que foi distribuída ao longo

das amostras pode ser representada por um pequeno número de coeficientes transformados.

A figura 3 demonstra este conceito, no qual temos um sinal bidimensional com os seus coeficientes correspondentes no domínio da transformada ao lado. Neste exemplo foi aplicada a DCT (transformada discreta de cossenos).

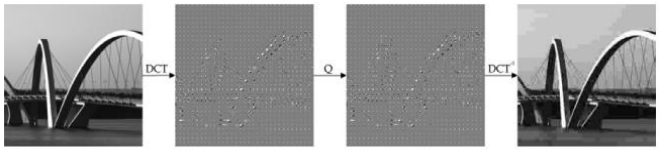


Figura 2 – Compressão de uma imagem, com a utilização do DCT, aplicada em blocos de 8x8 pixeis.

Por outro lado, temos uma transformada usada com mais frequência, a DWT (transformada discreta de wavelets). Esta transformada consiste na subdivisão sucessiva do espectro espacial da imagem em duas metades, nos sentidos horizontal e vertical. Cada banda passa a ter N/2 amostras.

Para podermos aplicar a TWD na imagem, esta terá que passar do domínio espacial para o domínio da wavelet, em um ou vários níveis. Quando é aplicada a TWD bidimensional obtém-se como resultado a matriz de coeficientes em quatro partes ou detalhes.

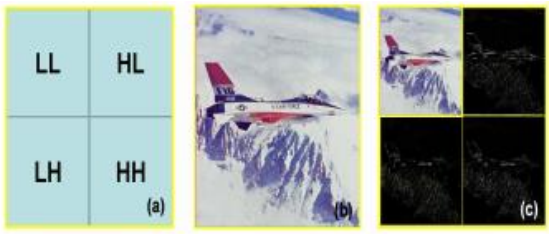


Figura 3 - Quadrantes de frequência da TWD aplicada a uma imagem; a) quatro quadrantes; b) imagem original; c) TWD da imagem em um nível.

Na figura 4 foi demostrado a decomposição da imagem após um nível de decomposição wavelet. De seguida, a cada nível a LL se divide em outros quatro quadrantes. É mais comum realizar três, quatro ou cinco níveis para a compressão de imagens.

II. COMPRESSÃO DE SINAIS UNIDIMENSIONAIS

A. Preditores

B.1. LPC

O LPC é uma operação matemática em que valores futuros de um sinal de tempo discreto são estimados como uma função linear de amostras anteriores, extraindo a informação essencial do sinal orignal.

$$\hat{x}(n) = \sum_{i=1}^{p} a_i x(n-i)$$

Utilizando a função lpc do matlab, [a,g] = lpc(x,p) encontra os coeficientes do preditor linear de ordem p que prevê o valor atual do sinal x de acordo com amostras anteriores.

$$\begin{bmatrix} r(1) & r(2)^* & \cdots & r(p)^* \\ r(2) & r(1) & \ddots & \vdots \\ \vdots & \ddots & \ddots & r(2)^* \\ r(p) & \cdots & r(2) & r(1) \end{bmatrix} \begin{bmatrix} a(2) \\ a(3) \\ \vdots \\ a(p+1) \end{bmatrix} = \begin{bmatrix} -r(2) \\ -r(3) \\ \vdots \\ -r(p+1) \end{bmatrix}$$

$$\hat{x}(n) = -a(2)x(n-1) - a(3)x(n-2) - \cdots - a(p+1)x(n-p)$$

Fig 4 – algoritmo lpc, sendo ^x(n) os coeficientes resultantes.

Após calculados os coeficientes é calculado o sinal estimado através da função filter, e o valor do erro (erro = sinal estimado - sinal original) para podermos calcular a quantida de de bits de diferença entre o sinal original e o sinal estimado.

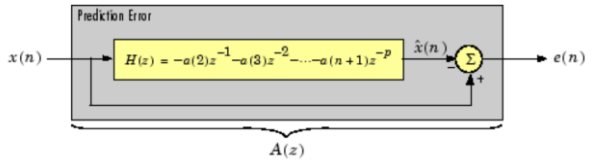


Fig 5 - *e*(*n*) – prediction error
A(z) – filter
x(n) – sinal original
x^(n) – predicted signal

B.3.Delta de primeira ordem

Função que começa por introduzir num array vazio um 0, de seguida são introduzidos os valores, do sinal original, até chegar ao penúltimo valor da matriz original, sendo o último descartado.

B. Compressão sem perdas

No que diz respeito às taxas de compressão as técnicas de compressão de sinais sem perda (entre elas o codificador run-length, o codificador de Huffman, o codificador aritmético) são menos eficientes do que as técnicas com perda mas podem vir a ser uteis em casos em que precisamos da reconstrução total do sinal original após a codificação e a descodificação.

C.1.Codificador Run-length

Arquivos de dados apresentam frequentemente caracteres repetidos sequencialmente. O comprimento dos caracteres repetidos pode ser representado de forma a evitar a redundância, como é ilustrado na figura 4.

Original code: 5 7 7 7 8 8 8 8 8 3 3 9 4 4 1 2 2 2 2 2 2

Run-Length: (1 5) (3 7) (5 8) (2 3) (1 9) (2 4) (1 1) (6 2)

Figura 4 – Exemplo de codificador run-length.

Se o número de zeros repetidos nos dados de entrada for superior a dois, o codificador terá alcançado compressão de dados.

A regras para a codificação Run-Length são:

-Quando três elementos, ou mais, se repetem consecutivamente, utiliza-se o Run-Length

-Senão, um carácter de controlo (00) é inserido, seguido do número de elementos da cadeia não comprimida e seguidamente desta última

-Se o número de elementos da cadeia for ímpar, o carácter de controlo (00) é acrescentado no fim;

-Os caracteres de controlo específicos para codificar:

- Fim de linha (00 01)
- Fim da imagem (00 00)
- Uma deslocação do ponteiro na imagem de XX colunas e YY linhas no sentido da leitura (00 02 XX YY).

Consideramos analisar esse tipo de codificação por ser um dos mais simples e não ter perdas de informação.

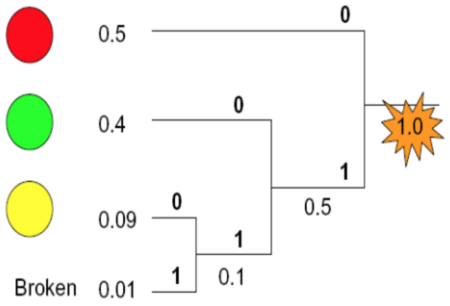
C.2.Codificador de Huffman

A codificação de Huffman é um método de compressão que usa as probabilidades de ocorrência dos símbolos no conjunto de dados a ser comprimido para determinar códigos de tamanho variável para cada símbolo.

Uma árvore binária completa, chamada de árvore de Huffman é construída recursivamente a partir da junção dos dois símbolos de menor probabilidade, e estes, são então os dois somados em símbolos auxiliares e estes símbolos auxiliares são recolocados no conjunto de símbolos. O processo vai terminar quando todos os símbolos forem unidos em símbolos auxiliares, formando uma árvore binária. A árvore depois vai ser percorrida, atribuindo-se valores binários de 1 ou 0 para cada aresta, para as arestas do lado esquerdo atribui-se o valor 1, e para as arestas da direita atribui-se o valor 0. Os códigos são então gerados a partir desse percurso.

O resultado do algoritmo de Huffman pode ser visto como uma tabela de códigos de tamanho variável para codificar um símbolo da fonte. Assim como em outros métodos de codificação, os símbolos mais comuns são geralmente representados usando-se uma menor quantidade de dígitos do que os símbolos que aparecem com menos frequência, que são representados por digitos menores, de forma a ocupar o menor númeor de bits possível.

Ao longo do sinal EMG, é possível encontrar repetições de eventos ou eventos de probabilidade igual. Por isso, uma boa solução para a compressão da fonte é a Codificação de Huffman. Ao agrupar eventos de mesma probabilidade, acaba por reduzir a variância e, consequentemente, aumenta a resistência a erros na codificação. É também capaz de reduzir o comprimento médio em bits da fonte, pois representa símbolos menos frequentes com códigos mais extensos com menos bits e símbolos mais frequentes com códigos mais pequenos usando poucos bits.



C.3.Golomb Rice

O sistema é baseado em codificação de entropia e tem como objetivo codificar números inteiros não negativos em que se verifica que a probabilidade de ocorrência é menor quanto maior for o número.

$$q = \left\lfloor \frac{n-1}{b} \right\rfloor, r = n - qb - 1$$

A codificação de Golomb-Rice tem como vantagem ser simples e de entendimento fácil. Esta codificação aplica-se a todos os números ‘n’ inteiros não negativos e que vão depender de um parâmetro ‘m’ que deve ser previamente calculado. Primeiro começa-se por obter a sequência de bits que queremos codificar usando a codificação de Golomb-Rice. De seguida vamos calcular o parâmetro ‘m’ usando a probabilidade ‘p’ de encontrar um bit ‘0’ na sequência anteriormente obtida.

$$m = \left\lceil -\frac{\log(1+p)}{\log(p)} \right\rceil$$

Nota: No Golomb-Rice, o ‘m’ vai ser arredondado á potência de 2 mais próxima do valor original.
Obter ‘n’ e calcular o coeficiente ‘q’ usando a fórmula:

$$q = \left\lfloor \frac{n}{m} \right\rfloor,$$

Transformar ‘q’ em unário (ex. 4=11110, 1=10, 2=110)
Calcular o resto ‘r’ e o parâmetro ‘c’ usando as seguintes fórmulas.

$$r = n - qm, \quad \text{and} \quad c = \left\lceil \log_2 m \right\rceil$$

Na codificação de Golomb original existe um passo extra. Esse passo consiste numa verificação:

- Se r < 2c − m,r é codificado como um inteiro sem sinal em c − 1 bits.
- Se r ⩾ 2c − m, r é representado como o inteiro sem sinal r+2c − m' em c bits.

Por fim, o código será uma concatenação do coeficiente em unário q com r → (r+2c –m).

Sequência Bits -> 00000100110001010000111010001

Subsequência de 0's -> 5 2 0 3 1 5 0 0 1 3 0 (ultimo '0' usado para indicar que a sequência acaba com bit '1')

$p(0) = 20/30 = 0,7$

$m = \lceil -\log(1+p)/\log(p) \rceil = \lceil -\log(1,7)/\log(0,7) \rceil = 2$

$q = \lfloor n/m \rfloor = \lfloor 5/2 \rfloor = 2$ -> 110 (unário)

$r = n - qm = 5 - 2 * 2 = 1$

$c = \lceil \log_2 m \rceil = \lceil \log_2 2 \rceil = 1$

Como 'm' é uma potência de 2 então $r \geq 2^c - m$

$r + 2^c - m = 1 + 0 = 1$ -> (1)1

$n=5$ vai ser codificado com a concatenação de 'q' e 'r' -> 110 1

E assim sucessivamente para o resto das subsequências de 0's

No final vamos ter a seguinte sequencia codificada -> 1101l100l00l101l01l1101l00l00l01l101l00

Fig.6 – Exemplo de uma codificação de uma sequência de bits com o codificador de Golomb Rice

C.4.Codificador Aritmético

Para podermos proceder a codificação de um fluxo de entrada de dados é necessária a existência de um alfabeto com a probabilidade de ocorrência de cada símbolo nele contido, para isso criamos um array com a contagem de cada símbolo do alfabeto. O fluxo de dados de entrada no início do processo encontra-se no intervalo [0, 1], este intervalo vai ser dividido em sub intervalos que correspondem a cada um dos símbolos do alfabeto da seguinte forma:

O limite inferior (fechado ([ )) é a probabilidade cumulativa até, mas deixando de fora o símbolo;

O limite superior(aberto ([ )) de cada subintervalo cumulativa até incluindo o símbolo;

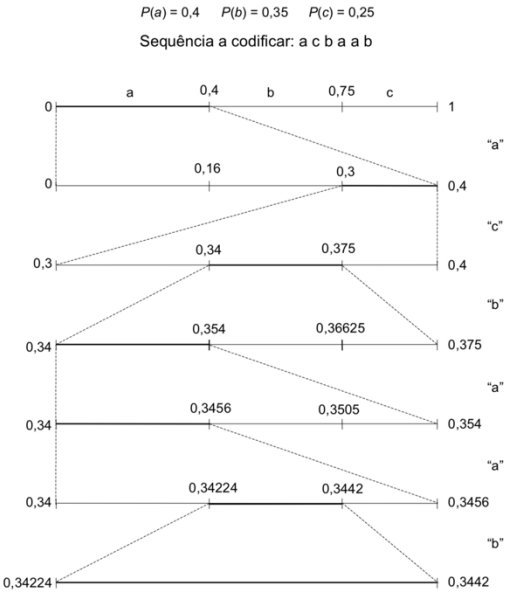
O primeiro passo do codificador Aritmético consiste na leitura e processamento do primeiro símbolo do fluxo de dados de entrada, em seguida vai detetar o sub intervalo correspondente ao símbolo lido e fazer com que este seja o novo intervalo atual que por sua vez vai fazer com que os restantes símbolos do alfabeto sejam recalculados com os novos limites do intervalo.

Para calcular para cada símbolo do fluxo de entrada aplicamos o algoritmo da codificação aritmética abaixo apresentado:

Codificador_aritmético (fluxo de dados de entrada)

```
1   anterior_baixo <- 0
2   anterior_alto <- 1
3   largura <- 1
4   do
5       ler novo símbolo do fluxo de dados de entrada
6       anterior_alto <- anterior_baixo + largura * subintervalo_alto (símbolo)
7       anterior_baixo <- anterior_baixo + largura * subintervalo_baixo (símbolo)
8       largura <- anterior_alto - anterior_baixo
9   while(simbolo <> terminador (EOF))
10  escrever na saida um código binário tal que
11      anterior_baixo <= código < anterior_alto
```

Exemplo:



O código deve ser um número real incluído no intervalo [0,34224; 0,3442[.

III. APLICAÇÃO DOS MÉTODOS

A. Critérios de avaliação

Para avaliar a taxa de compressão, torna-se necessário quantificar o ganho resultante da sua utilização. Normalmente, tal é conseguido através da razão entre o número de bits necessário para representar o sinal original e o número de bits necessários para representar o sinal na sua forma comprimida.

$$CR = \frac{nBitsOriginal}{nBitsComprimido}$$

O desempenho do algoritmo de compressão dos sinais de EMG é avaliado usando o critério quantitativo: o fator de compressão, CF.

O fator de compressão é definido por:

$$CF(\%) = \frac{Os - Cs}{Os} \times 100\% ,$$

em que Os é a quantidade de bits necessária para armazenar os dados originais e Cs é a quantidade de bits necessária para armazenar os dados comprimidos.

1. *Sem preditores*

Sem preditores observamos que a entropia do sinal EMG é 8.480 e o número total de bits do sinal é 1302944.

2. *Com preditores*

a. LPC + Codificador de Huffman

Após a aplicação do preditor LPC (Linear Prediction Filter Coefficient) podemos verificar uma descida na entropia em relação ao sinal original, houve uma descida de 8.48 para 8.26, no que resultou no aumento da redundância do sinal. No entanto ao aplicarmos o codificador de Huffman podemos concluir que houve uma taxa de compressão de 45%, passando a razão de compressão para 1:1.829. Passando o sinal a ter 712283 bits necessários para representar a informação do sinal original.

b. LPC + Codificador de Golomb

A compressão do sinal juntando o lpc e o codificador de Golomb resultou numa taxa de compressão de 24,44%, e uma razão de compressão de 1: 1.323, tendo o sinal que se obteve no final um tamanho de 984507 bits.

c. LPC + Codificador Aritmético

A utilização do codificador aritmético após a aplicação do Linear Prediction Filter Coefficient, tendo o sinal que se obteve no final um tamanho de 710077 bits o que causou uma diminuição de 592867 bits em relação ao array do sinal original, traduzindo-se numa taxa de compressão de 45.6% e uma RC de 1: 1.835.

d. LPC + Run Length Encoder

Neste caso a taxa de compressão obtida foi de 31%, com um RC de 1.448, tendo obtido um tamanho total de 899820.

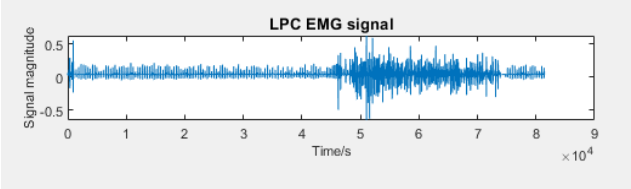


Fig.7 – Sinal EMG após o preditor LPC

e. Delta de primeira ordem + Codificador de Huffman

O uso do preditor Delta no sinal original resultou numa diminuição da entropia para 6.836 em relação 8.48 do sinal original. De seguida foi aplicado o codificador de Huffman. Este causou uma razão de compressão de 1:2,332, e uma taxa de compressão de 57.2%, tendo obtido um tamanho final de 558730 bits.

f. Delta de primeira ordem + Codificador de Golomb

A compressão do sinal juntando o delta e o codificador de Golomb resultou numa taxa de compressão de 24,95%, e uma razão de compressão de 1: 1.333, tendo o sinal que se obteve no final um tamanho de 977792 bits.

g. Delta de primeira ordem + Codificador Aritmético

A compressão do sinal juntando o delta e o codificador aritmético resultou numa taxa de compressão de 57,27%, e uma razão de compressão de 1: 2.340, tendo o sinal que se obteve no final um tamanho de 556696 bits.

h. Delta de primeira ordem + Run Length Encoder

A compressão do sinal juntando o delta e o codificador aritmético resultou numa taxa de compressão de 31,201%, e uma razão de compressão de 1: 1.456, tendo o sinal que se obteve no final um tamanho de 895103 bits.

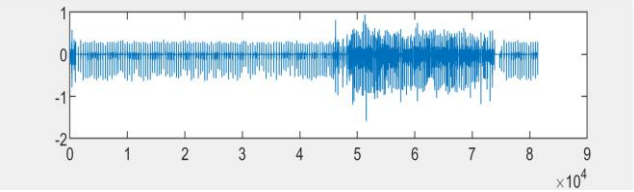


Fig.8 – Sinal EMG após o preditor Delta

IV. Análise e Conclusões

Resultados:

Original: 1302944 bits

Preditor/Codent (em bits)	Huffman	Golomb-Rice	Aritmético	Run Length
LPC	712283	984507	710077	899820
Delta	558730	977792	556696	895103

Tabela 1

CR	Huffman	Golomb-Rice	Aritmético	Run Length
Lpc	1.829	1.323	1.835	1.448
Delta	2.332	1.333	2.340	1.456

Tabela 2

Taxa Compressão	Huffman	Golomb-Rice	Aritmético	Run Length
Lpc	45%	24%	46%	31%
Delta	57%	25%	58%	31%

Tabela 3

	Original	LPC	Delta
Entropia	8.480	8.260	6.836

Tabela 4

Podemos observar que para o sinal EMG a codificação de Golomb-Rice não é tão eficiente quanto as codificações Huffman e Aritmética independente do preditor utilizado. Isso porque tal codificação deixa de ser tão eficiente quanto há símbolos com probabilidades baixas.

A compressão por Run-Length Coding também teve resultados inferiores, uma vez que só se torna viável para fontes vários símbolos consecutivos iguais ao longo do código original e o sinal de EMG constantemente muda de evento.

Também analisamos que, para todos os preditores, os bits totais necessários para a codificação aritmética e para a codificação de Huffman são próximos, com a aritmética sendo um pouco melhor. Concluímos que isso acontece por termos um alfabeto com muitos símbolos (cerca de dois mil e quinhentos símbolos) e, ao agrupá-los segundo a árvore de Huffman, temos um crescimento exponencial do número de símbolos. Isso exige muito da memória e torna-se mais viável a utilização de códigos aritméticos, que codifica uma sequência sem utilizar arranjos.

Sendo a entropia o número médio de bits para codificar uma fonte, concluímos que o melhor preditor usado foi o preditor Delta, em que $x^*prev(k) = x(k-1)$.

O CODEC que apresentou melhores resultados no nosso trabalho foi a junção do preditor delta com o codificador aritmético, tendo sido obtida uma taxa de compressao de 58%.

V. Discussão e Conclusões

Neste trabalho foi apresentado métodos para a compressão de sinais de eletromiografia de superfície utilizando algoritmos de compressão de sinais unidimensioanis com vários preditores e codificadores entrópicos.

VI. Trabalhos Futuros

Como trabalho futuro sugerimos o tema abordado no nosso estado de arte anterior de técnicas bidimensional de compressão de sinais S-EMG utilizando codificadores de imagem como o JPEG, uma vez que ao comparar com outros codificadores unidimensionais apresentaram uma maior taxa de compressão. Para além disto, a abordagem baseada em codificadores de imagem disponíveis no mercado tem potencial de implementação e disseminação rápida uma vez que muitos sistemas já trazem incorporados estes codificadores.

Referências

[1] Salomon D. Data Compression: the Complete Reference. 3. New York: Springer; 2004.

[2] Ziv J, Lempel A. A universal algorithm for sequential data compression. IEEE Trans Information Theory. 1977;23(3):337–343. doi: 10.1109/TIT.1977.1055714.

[3] Ziv J, Lempel A. A universal algorithm for sequential data compression. IEEE Trans Information Theory. 1977;23(3):337–343. doi: 10.1109/TIT.1977.1055714.

[4] Wallace GK. The JPEG still picture compression standard. Commun ACM. 1991;34:30–44

[5] Wallace GK. The JPEG still picture compression standard. Commun ACM. 1991;34:30–44.