Fotopletismograma

Francisco Pires Moreira Ribeiro
Depart. de Engenharia Informática
2017263852

Ricardo Martins
Depart. de Engenharia Informática
2017246268

Tiago Ferreira Fernandes
Depart. de Engenharia Informática
2017242428

Keywords—PPG

I. Introduction

Um fotopletismograma (photoplethysmogram em inglês) é uma técnica de medição de mudanças de volume do sangue nos vasos sanguíneos, providenciando diagnósticos essenciais às funções ao sistema cardiovascular do corpo humano. Este é feito colocando uma pinça na ponta do dedo com uma luz e, com base na variação do volume dos vasos sanguíneos, esta consegue medir estes valores (Figura 1). Através do resultado de um PPG e cálculos efetuados com a ajuda de um computador, é possível obter, para além das mudanças de volume nos vasos sanguíneos, o ritmo cardíaco, o nível de saturação de oxigénio no sangue, a pressão sanguínea, entre outros, …

Nos últimos anos, o interesse no PPG tem vindo a aumentar, graças às suas capacidades de fazer um bom diagnóstico, ser bastante portátil, simples e fácil de usar, além do seu baixo custo. Também já se tentou fazer um PPG via wireless, mas não envolveram nenhum método de compressão.

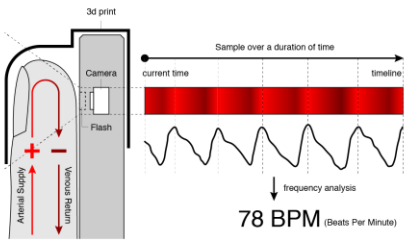


Figura 1: Funcionamento do PPG

II. Formas de codificação

A. *Delta Encoding*

Delta Encoding é um algoritmo que pode ser usado para a compressão de dados e consiste em guardar, não valores mas sim a diferença entre valores adjacentes. Este método tem limitações, nomeadamente quando a distância entre valores adjacentes varia bastante, no entanto, serve perfeitamente para o tipo de dados gerado por um sinal, que é o caso do PPG. Para fins de compressão, após ou uso do delta encoding, são geralmente utilizados, os códigos de huffman (esta opção irá ser explorada mais à frente) ou run length encoding [5]. Este tipo de codificação é uma forma muito simplificada de Linear Predictive Coding (LPC), no entanto, enquanto o LPC tem em consideração todas as amostras prévias o delta encoding apenas considera a amostra codificada anteriormente para fazer a sua previsão. Apesar disso ambos os métodos utilizam um preditor que apenas varia na complexidade da função associada, sendo ambos algoritmos preditivos sem perdas.

B. *Diferencial Pulse-Code Modulation*

Este método usa um preditor para poder prever o valor atual, baseando-se nos valores previamente calculados, trata-se de um preditor linear com perda de dados, uma vez que na sua implementação usa um quantizador para o erro, tornando assim impossível recriar os valores originalmente transmitidos pela fonte, ou seja, usando esta implementação é pouco provável que os valores resultantes sejam iguais, no entanto vamos obter uma boa aproximação. O uso deste tipo de algoritmos vai aumentar a redundância dos dados e permitir uma maior taxa de compressão. Especificamente para o PPG é usada uma variação deste algoritmo onde apenas é comparado o valor atual, a ser codificado com o que foi codificado anteriormente (Delta Modulation, irá ser explorado mais à frente).

C. *Linear predictive coding*

Linear predictive coding (LPC abreviado) é uma ferramenta bastante precisa e eficiente para modelar sinais de áudio. Este método consiste em criar um modelo para os dados apresentados, que é posteriormente representado por um conjunto de coeficientes, o número de coeficientes depende da ordem que é usada. No nosso caso determinamos a ordem -n a utilizar comparando, para vários valores n, os diferentes valores de entropia e do erro quadrático da predição para chegar ao melhor valor para a ordem do LPC. Os coeficientes resultantes são utilizados para fazer uma previsão dos valores originais, seguidamente é calculado o erro entre a previsão e a amostra original, e esse erro vai ser quantizado para ser posteriormente comprimido. No entanto para este processo, para além do erro quantizado, é necessário guardar também todos os coeficientes e também os primeiros n valores da fonte onde n é a ordem de predição usada para criar o modelo.

D. *Move-To-Front*

O algoritmo move-to-front é um algoritmo de reordenamento que é utilizado para diminuir a entropia de uma fonte. O funcionamento deste algoritmo está descrito em [6]. O vetor gerado por este algoritmo é composto por valores inteiros não negativos. Apesar deste algoritmo poder ser utilizado em qualquer parte do processo de compressão são obtidos melhores valores se este algoritmo for utilizado antes de um codificador entrópico [6].

A maior desvantagem do uso deste algoritmo na compressão é o facto do alfabeto da fonte ter de ser compresso juntamente com os valores produzidos pelo

Move-to-front, de outra forma seria impossível reconstruir o sinal original.

III. TIPOS DE COMPRESSÃO

Existem vários métodos de compressão de um sinal PPG, mas apenas nos vamos focar em 5 deles.

A. *Delta encoding e Huffman*

Para usar a codificação delta e Huffman [1], os valores que vamos usar para contruir a árvore de Huffman são as diferenças absolutas entre as várias amostras consecutivas que são obtidas com recurso a um preditor. Um preditor é uma maneira de prever os valores do sinal e com base no seu acerto ou não, envia essa diferença entre o valor que previu e o valor real para um array [6] (Figura 2). De seguida, são atribuídos códigos em função das probabilidades de ocorrência das diferenças absolutas (Tabela 1). Por fim, é criada a árvore de Huffman (Figura 3) em simultâneo com o descodificador. Por exemplo, observando a sequência 543026, fica codificada como 01011011101111011111.

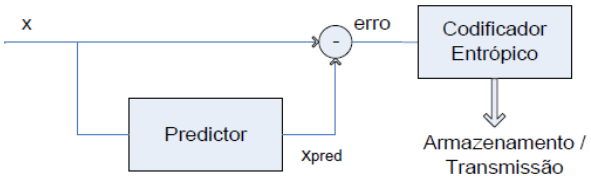


Figura 2: Preditor e o envio do erro

$b\Delta_2$ elements (symbols)	Number of occurrences	Probability	Codes
5	18	0.375	0
4	15	0.312	10
3	12	0.250	110
0	1	0.021	1110
2	1	0.021	11110
6	1	0.021	11111
	$\Sigma = 48(*)$		

(*: sample block size considered)

Tabela 1: Tabela de Huffman e códigos

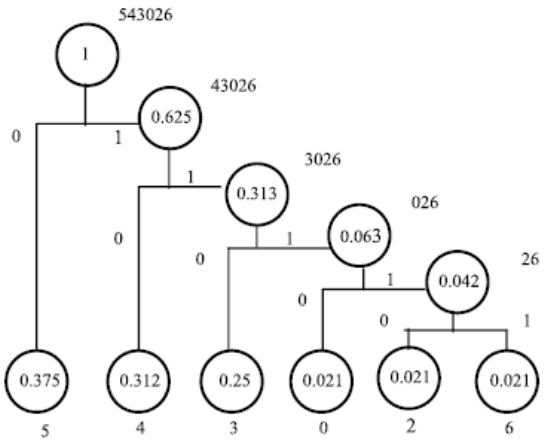


Figura 3: Árvore de Huffman criada com base na tabela 1

B. *Grouping*

O método de grouping [2] é um algoritmo de compressão que não se baseia em nenhuma transformação matemática complexa como por exemplo a transformação Wavelet. Simplesmente trata-se de um agrupamento de valores seguindo regras definidas de modo a conseguir comprimir ao máximo o sinal já previamente tratado. Define-se um tamanho em que se divide o sinal e aplicam-se essas regras. Por exemplo, define-se tamanho de 8 números inteiros e uma regra pode ser se todos esses números forem iguais, representamos esse grupo por esse único inteiro.

3	4	7	12	14	6	11	13
b[1]	b[2]	b[3]	d[4]	b[5]	b[6]	b[7]	b[8]

⇩

34=(3×10+4)	107=(7+100)	112=(12+100)	114=(14+100)	106=(6+100)	189=(11×16+13)
c[1]	c[2]	c[3]	c[4]	c[5]	c[6]

Antes de podermos utillizar esta técnica temos vários passos de preparação como o *Denoising* do sinal, que passa por eliminar grande parte do sinal que sabemos que não tem informação útil para o caso. Neste caso sabemos que as frequências acima de 15Hz podem ser eliminadas num sinal PPG. Também temos o *Downsampling* que consiste em baixar a frequência do sinal para diminuir o tamanho. No caso do PPG o sinal é adquirido a 500Hz sampling rate com 16-bit de resolução. Ao baixarmos a frequência para 100Hz o que vai ajudar também no tempo de computação.

Por fim, temos o tratamento dos samples do sinal antes de passarem pelas regras de Grouping. O objetivo é termos só inteiros positivos em cada sample logo vamos multiplicar por 100 e ignorar qualquer casa decimal. E por fim vamos gerar um ***Sign-Byte*** antes de cada grupo de 8 samples que determina quando um sample é negativo ou positivo conseguindo assim na descompressão obter o sinal correto.

0.03	0.04	-0.07	0.12	0.14	0.06	-0.11	0.13
d[1]	d[2]	d[3]	d[4]	d[5]	d[6]	d[7]	d[8]

⇩

3	4	-7	12	14	6	11	13
a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]

C. *Delta Modulation*

O Delta Modulation (DM) é um algoritmo usado para converter sinal analógico para digital de uma forma rápida e simples, no entanto, a utilização deste método permite a redução do tamanho de conjunto de dados.

Este processo de compressão usa um algoritmo preditivo com perdas que gera um output baseado na predição anterior e no erro correspondente, considerando o primeiro valor o valor real da fonte uma vez que não pode ser previsto por não existirem valores anteriores. [3] Este algoritmo consiste, portanto, em reduzir o tamanho de todos os valores, com exceção do primeiro, para um bit. O passo é um valor constante, dado como parâmetro, ajustado previamente para se adequar aos valores obtidos da fonte, pode ser calculado sabendo a voltagem de referência do aparelho usado para fazer o exame [6], através da equação (1). O erro é calculado através da equação (2). Quando o erro é positivo é transmitido o valor 1 para a nova fonte (fonte codificada), caso contrário é 0 [4]. A descodificação é feita através do processo inverso, partindo do primeiro valor, relembrar que este não foi codificado, da cadeia de bits codificados e da constante delta. Quando aparece 1 na cadeia codificada soma-se delta ao valor anterior, caso contrario subtrai-se delta. Este processo pode então resumir-se às equações:

Cálculo do erro:

(1) Cálculo do passo sabendo a voltagem de referência do aparelho (Esta fórmula aplica-se apenas ao PPG).

$$Passo\ (\mu V) = \frac{Vmax - Vmin}{2^{\alpha-1}}$$

(2) E(n)=X(n)-X'(n), onde X(n) é o valor real e X'(n) é o valor obtido pela codificação do valor anterior.

(3) Y(n) = 1 ou Y(n) = 0, consoante o erro,

1 se o erro é positivo;
0 se o erro é negativo.

Sendo que Y(n) é a fonte comprimida, no entanto Y(n=1) tem o valor da fonte e não um valor codificado.

D. *Codificação Aritmética*

A codificação aritmética é um método de codificação entrópica usado para comprimir dados, quando a distribuição estatística dos dados a comprimir é conhecida [6]. Este método consiste em representar um conjunto de dados por valores reais maiores ou iguais a 0 e menores que 1[8]. Este método em vez de ser fixo relativamente ao número de símbolos a codificar depende da frequência estatística com que a fonte produz cada símbolo do seu alfabeto [9]. Inicialmente a distribuição é feita pela probabilidade de cada elemento do alfabeto da fonte. Á medida que vai codificando novos símbolos os intervalos vão sendo subdivididos para codificar um símbolo novo. Este processo é repetido até que seja possível representar vários símbolos por um valor [6].

Dependendo da distribuição das probabilidades este método pode obter melhores resultados para a taxa de compressão que o huffman encoding. A sua vantagem provém do facto de ser possível agrupar vários símbolos da fonte e representa-los com apenas um símbolo na fonte comprimida.

E. *Run Length Encoding (RLE)*

Este algoritmo de compressão consiste em agrupar símbolos consecutivos iguais, agrupa-los e representa-los com o menor número de bits possível [10]. Este método é um dos tipos de compressão mais populares e mais intuitivos. No entanto, este tipo de codificação e bastante limitado pela fonte, uma vez que, é tão mais eficiente quanto maior for a redundância dos dados na fonte a comprimir, ou seja não obtêm bons resultados onde a repetição de símbolos é pouco provável ou em fontes onde as repetições tenham um comprimento reduzido, por exemplo, se apenas se cada símbolo aparecer no máximo duas vezes seguidas.

AAAABBBBBCCCDDDDDD

poderia mais eficientemente ser codificada como:

(4,*A*), (5, *B*), (3, *C*), (6,*D*)

conseguindo assim um ganho significativo em espaço de representação.

Funcionamento do RLE, agrupamento de símbolos

IV. Modulação de Dados

A redundância dos dados é um fator importante para a compressão, visto que o objetivo deste processo é reduzir ou eliminar a redundância dos dados, a compressão vai ser tão mais eficaz quanto maior for a redundância da fonte a comprimir. São usados diferentes métodos para o tratamento de dados dessas fontes de forma a aumentar a eficácia da compressão corrompendo ao mínimo os dados originais.

São usados preditores com e sem perda de informação em delta modulation e em delta encoding, respetivamente. Estes algoritmos, como se baseiam no valor previamente codificado para codificar o valor atual aumenta a probabilidade de que valores consecutivos sejam semelhantes ou iguais.

O método de compressão grouping usa outra metodologia para aumentar a redundância da fonte. Primeiramente é usado o downsampling para reduzir o tamanho e a complexidade do tratamento de dados. Seguidamente é usada a truncação, uma vez que os dados de PPG são gerados com pelo menos 3 casas decimais usamos a truncação para deixar todos os valores com apenas duas casas decimais aumentando assim o numero de valores semelhantes, visto que por exemplo 0.313.0,312 e 0,3176 vão passar a ser números iguais(0.31), este passo aumenta significativamente a taxa de compressão, no entanto é também neste passo que é gerado o erro que se verifica entre os dados originais e os dados depois de serem codificados e descodificados.

V. ANÁLISE DOS RESULTADOS PARA A COMPRESSÃO

Para analisar os vários resultados, existe uma fórmula que compara a eficiência dos compressores

$$CR = \frac{file\ \ size\ \ of\ \ original\ \ PPG\ \ data}{file\ \ size\ \ of\ \ compressed\ \ PPG\ \ data}$$

$$PRD\% = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \overline{y_i})^2}{\sum_{i=1}^{n} y_i^2}} \times 100\% \qquad QS = \frac{CR}{PRD}$$

Onde o CR é a taxa de compressão, o numerador é o tamanho da fonte original antes de ser tratada e o denominador é o tamanho da fonte após ter sido comprimida. São ainda utilizados outros parâmetros para avaliar a performance dos algoritmos de compressão, como por exemplo o Percent Root mean square Difference (PRD) e o Quality Score (QS). QS é um valor numérico usado quantificar o desempenho de um algoritmo de compressão [2]. O PRD é usado para medir a diferença entre os dados das duas fontes, a fonte original e a fonte resultante da compressão e descompressão.

Usando a codificação de Huffman, o método de grouping e o Delta Modulation, o valor médio de CR é de 2.223, 122.24 e 16, respetivamente.

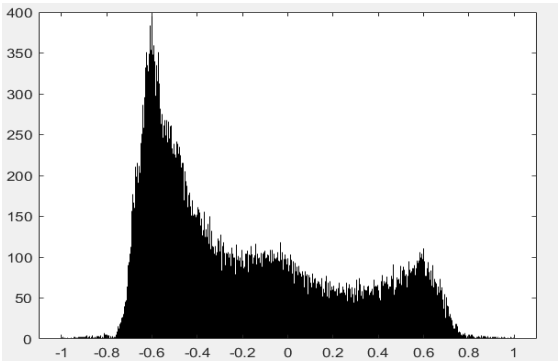
De todos os métodos analisados no trabalho e até à data, o mais eficiente é o método de grouping (Tabela 2).

Methods	PRD (%)	PRDN (%)	CC	CR	QS
Delta modulation	0.000392	–	–	16	40816.32
Fourier series analysis	–	–	–	12	–
Delta and Huffman encoding	0.127	0.187	–	2.223	17.50
Delta encoding	5.82	7.57	–	3.84	0.6597
Proposed	0.02	0.03	0.998845	122.24	7228.41

Tabela 2: Valores de CR entre os vários analisados

VI. TESTES E RESULTADOS

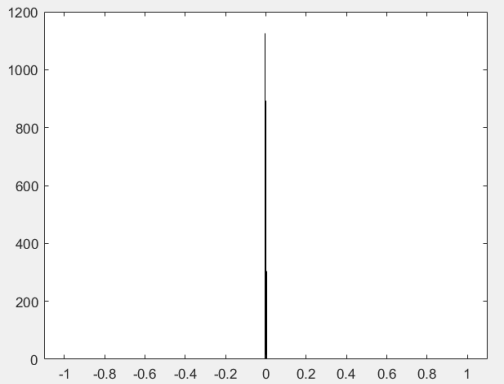
Foram realizados vários testes, tanto com preditores como com métodos de compressão. A fonte está quantizada a 16 bits com uma frequência de amostragem de 100Hz. As quantizações dos erros vão também ser feitas a 16 bits. Calculando sabemos que o tamanho de bits da fonte original quantizada é de 60001*16 bits, ou seja, 960016 bits no total.



Valores da fonte e a sua frequência

Durante a fase de testes experimentamos dois métodos para diminuir a entropia. O delta encoding que consiste em guardar a diferença entre um valor e o valor imediatamente anterior. E testamos também um preditor linear (LPC).

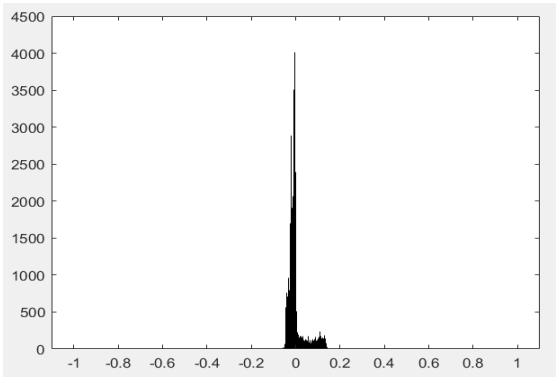
Relativamente ao LPC, para calcular a ordem de predição a usar para criar um modelo para os dados originais, para cada valor da ordem calculou-se a entropia e o erro quadrático entre os valores previstos e os valores reais. Para tentar chegar ao melhor valor possível fizemos testes com valores para a ordem de predição entre 1 e 1000. O melhor valor obtido tanto para a entropia, como para o erro quadrático foi com ordem igual a 1. Usando ordens superiores vamos obter uma entropia maior, o que provavelmente se deve ao facto de a predição usar os coeficientes.



Valores do erro quantizado usando o LPC e a sua frequência

Apesar de o gráfico da distribuição dos valores do erro da predição do LPC parecer pouco disperso e próximo de zero, se observarmos o eixo dos ordenadas que mostra a quantidade de valores em cada intervalo da quantização podemos ver que faltam representar vários valores uma vez que existem 60001 valores no vetor. Após uma análise mais cuidadosa concluímos que existem valores no alfabeto que apesar de não serem visíveis no gráfico também surgem nos valores do erro a representar, no entanto a probabilidade da sua ocorrência é bastante baixa. Usando o LPC de ordem 1 conseguimos uma entropia de 7.2668.

Ao usar o delta encoding conseguimos alcançar, para o erro quantizado, uma entropia de 6.0668. Reduzimos o número de valores diferentes a representar de 689 para 267, o que nos vai permitir aumentar a taxa de compressão.



Valores do erro quantizado usando o Delta Modulation e a sua frequência

Ao analisar o gráfico que mostra os valores do erro resultantes do uso do delta encoding (eixo das abcissas) e a sua frequência (eixo das ordenadas), observamos que os valores aparentam estar muito mais dispersos que os valores do erro quantizado resultante do LPC, no entanto a entropia destes valores é mais baixa.

Após obtermos os resultados dos preditores passámos á fase de compressão. Começamos por tentar comprimir a fonte sem recorrer aos preditores para usar como valor de referência. Com o RLE, aumentamos o tamanho do ficheiro, com os códigos de Huffman obtivemos uma taxa de compressão de 1.8002. Testamos ainda vários algoritmos de compressão, para cada tipo de preditor usado. Os algoritmos usados foram o RLE, os códigos de Huffman e a codificação aritmética.

Começando pelos códigos de Huffman, começamos por gerar o dicionário com o auxílio da função *huffmandict* do MATLAB, seguidamente recorremos a outra função do MATLAB, *huffmanenco*, esta função retorna um bit stream (0's e 1's), que representam os valores codificados.

Começando por comprimir os erros quantizados do LPC usando os códigos de Huffman, no entanto é necessário ter em conta que é necessário guardar os coeficientes gerados pelo LPC e também os primeiros n valores do array original, sendo n a ordem de predição usada. Após o uso das funções acima referidas obtivemos um vetor de valores binários com um tamanho de 437757 valores que podem ser representados por um bit cada. Recorrendo á formula para calcular a taxa de compressão, chegamos ao resultado, obtendo uma taxa de compressão de 2.1928.

Testamos também a compressão através dos códigos de Huffman sobre o erro quantizado do delta encoding, como tinha sido descrito em [1]. Conseguimos um valor para a taxa de compressão de 2.6270, que é ligeiramente superior valor obtido pelo autor, o que pode ser influenciado pela qualidade dos dados da fonte.

Seguidamente tentámos comprimir os erros quantizados utilizando o *Run Lengh Encoding*. Para calcular a compressão utilizamos uma função disponibilizada para o MATLAB, não criada por nós, mas que se encontra em [11].

No entanto, devido ao facto de a repetição ser reduzida, o tamanho do ficheiro comprimido acabou por aumentar uma vez que em vez de guardar apenas um vetor com os valores do erro quantizado passou a guardar dois vetores com quase o mesmo tamanho do original em que o primeiro. Este tipo de resultados verificou-se tanto na compressão do erro gerado pelo LPC, no erro gerado pelo Delta Encoding, nos valores codificados pelo Move-to-Front e também quando tentamos aplicar o RLE diretamente á fonte.

Usando o algoritmo do move-to-front conseguimos usar os códigos aritméticos, uma vez que os valores codificados pelo algoritmo são inteiros positivos. O que nos permite utilizar o algoritmo da compressão aritmética. Contando com o alfabeto da fonte a quantizar que é necessário guardar obtemos uma taxa de compressão de 1.8624. Testamos também comprimir os valores codificados pelo MTF com o RLE e com os códigos e Huffman, no entanto os resultados não foram muito bons.

VII. Conclusoes

Após vários testes concluímos que a melhor forma de compressão de dados sem perdas para dados do tipo PPG é utilizando o delta modulation para diminuir a entropia, e seguidamente utilizar um codificador entrópico, sendo o que teve melhores resultados dos testados os códigos de Huffman, tal como tinha sido referido em [1]. Ao usar o LPC temos um erro menor no geral, no entanto as taxas de compressão com os diferentes algoritmos são mais baixas.

Podemos também concluir que os algoritmos de compressão que se baseiam na codificação de cadeias de símbolos não obtêm bons resultados para a compressão deste tipo de dados, deve-se principalmente ao facto de não existirem sequências de grandes dimensões com o mesmo valor, visto que se trata de uma onda.

VIII. Trabalho Futuro

Para um trabalho futuro, o método de compressão melhor seria o grouping porque como observarmos na nossa pesquisa inicial, é o algoritmo que obtém melhor taxa de compressão, no entanto é um método que implica perda de dados relativamente ao sinal original. Sendo possível transformar este algoritmo num método de compressão sem perdas de dados, e usando um preditor que consiga reduzir ao máximo o número de valores diferentes, ou seja, que diminua o seu alfabeto de ocorrências, seria possível atingir valores de compressão mais elevados. Para melhorar ao máximo o desempenho é necessário encontrar o melhor modelo possível do grouping, visto que, o que se encontra acima descrito é funcional quando assumimos a possibilidade da perda de dados.

Outra opção a explorar futuramente seria testar a eficácia do uso as transformadas de Fourier para a compressão do sinal em causa. Como podemos ver na tabela 2, o autor conseguiu atingir valores de compressão mais elevados, mantendo o processo lossless ou near-lossless.

Referências

[1] R. Gupta, Lossless compression technique for real-time photoplethysmographic measurements, IEEE Trans. Instrum. Meas. 64 (4) (2015) 975–983.

[2] Dhar, S., Mukhopadhyay, S. K., Pal, S., and Mitra, M. (2018). An efficient data compression and encryption technique for PPG signal. *Meas. J. Int. Meas. Confed.* 116, 533–542. doi:10.1016/j.measurement.2017.11.006.

[3] K.S. Chong, K.B. Gan, E. Zahedi, M.A.M. Ali, Data compression technique for high resolution wireless photoplethysmograph recording system, in: Proc. of the 2013

[4] K. Prashant, R. Kumar, Study on performance analysis of Delta Modulation (DM), July 2017.

[5] S. W. Smith, The Scientist and Engineer's Guide to Digital Signal Processing, 1999

[6] M. Brito, J. Henriques, P. Carvalho, B. Ribeiro, M. Antunes, An ECG compression approach based on a segment dictionary and Bezier approximations, 5th European Signal Processing Conference (EUSIPCO 2007), Poznan, Poland, September 3-7, 2007

[7] K. S. Chong, E Zahedi, K. B. Gan, M. A. M. Ali, Evaluation of the Effect of Step Size on Delta Modulation for Photoplethysmogram Compression, December 2013

[8] I. H. Witten, R. M. Neal, J. G. Cleary. Arithmetic coding for data compression, June 1987

[9] Mathworks.com. (2018). Arithmetic Coding – MATLAB & Simulink. [online] Available at: https://www.mathworks.com/help/comm/ug/arithmetic-coding-1.html [Accessed 16 Dec. 2018]

[10] Filestore.aga.org.uk. (2018). [online] Available at: https://filestore.aga.org.uk/resources/computing/AQA-8520-TG-RLE.PDF [Accessed 16 Dec. 2018]

[11] Matworks.com. (2018). RLE de/encoding – File Exchange – MATLAB Central. [online] Available at: https://www.mathworks.com/matlabcentral/fileexchange/4955-rle-de-encoding [Accessed 16 Dec. 2018].