Data Compression using Discrete Cosine Transform in TelosB Mote/Cooja

Assignment 2, WSN 2021

1 Introduction

In this assignment, data compression using discrete cosine transform (DCT) will be studied for the time-series signals. For testing electrocardiogram (ECG) signals will be used. DCT-II transform of a length N signal $\mathbf{x}$ is given as,

$$y_k = \sum_{n=0}^{N-1} x_n \cos(\frac{\pi}{N}(n+\frac{1}{2})k), \tag{1}$$

where y_k is the k^{th} DCT coefficient of the signal, $\mathbf{x}$, for a particular k. It can be also be rewritten in the matrix form as,

$$\mathbf{y} = \mathbf{A}\mathbf{x}, \tag{2}$$

where $\mathbf{A} \in \mathcal{R}^{N \times N}$ is a DCT matrix. ECG signal for $N = 512$ is shown in Fig. 1.

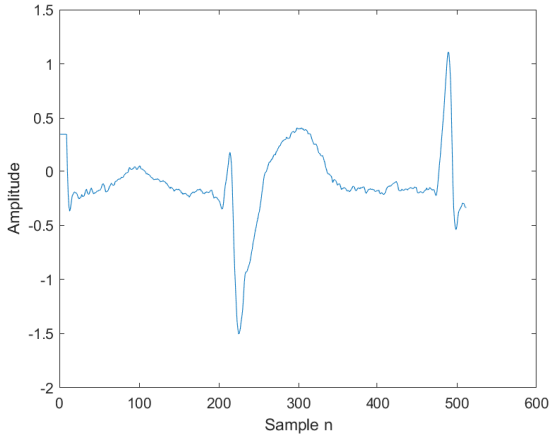

Figure 1: Original ECG signal

The DCT transform of this signal is shown in Fig. 2. From this figure it can be observed that the most significant information is in the first few DCT coefficients The

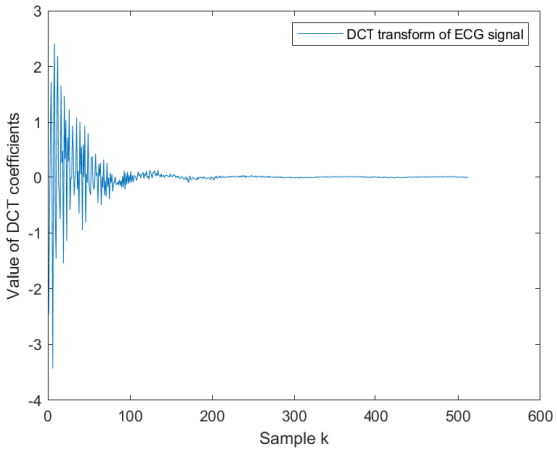

Figure 2: DCT coefficients

compression is achieved by selecting only some DCT coefficient and making the rest as zero. The signal is reconstructed from the selected coefficients only. It can be observed that first $M = 120$ DCT coefficients contain the significant information. The reconstructed signal from the $M = 120$ DCT coefficients is shown in Fig. 3.

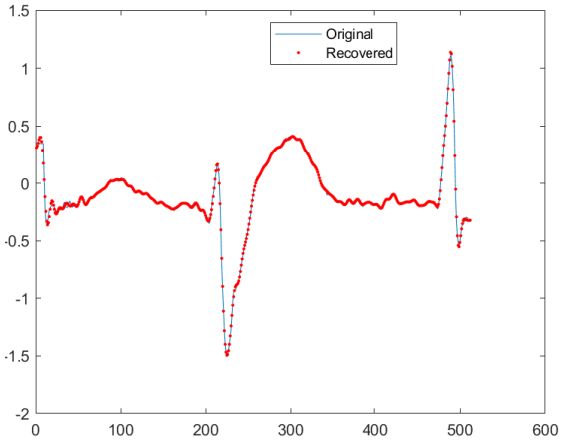

Figure 3: DCT coefficients

It can be observed that the reconstructed signal is almost identical to the original signal. The compression ratio achieved here is $\frac{120}{512}$. The DCT coefficients can be further compressed by applying entropy coding such as, Huffman coding.

2 Goal

The goal of this assignment is to implement DCT transform in the TelosB mote in an energy-efficient way for time series data.

Tasks to be completed are:

- Implement and measure the execution time and energy consumption for the following cases:
  - $N = 256$ and $M = 75$.
  - $N = 512$ and $M = 120$.
- Measure the mean square error between the original signal and reconstructed signal for both the cases.

Here N is the signal length and M is the number of selected DCT coefficients.

NOTE: Signal compression should be performed in TelosB mote/Cooja and then the compressed signal can be transferred to the connected workstation/laptop for reconstruction.

3 Possible Approaches and Tips

DCT is materialized using only *cos()* function. One possible naive approach is to use the *cos()* from the **math** library and implement Eq. 1 in the TelosB mote. Another approach is to use the symmetry of the *cos()* function and implement Eq. 1 in a energy-efficient way. One can follow the given references [1,2] for designing the DCT in energy-efficient way.

3.1 Tips

Suggestions for TelosB mote implementation:

- Use of fixed-point arithmetic can help in reducing execution time and memory usage.
- **Execution time measurement in TelosB mote:** The execution time can be measured by placing the function *clock_time()* before and after the function. The difference in reading provides the execution time in ticks because the function *clock_time()* gives result in ticks. There are 128 ticks per second. It can be observed that the resolution of *clock_time()* is not that good. One can also use *RTIMER_NOW()* for better resolution but it resets in every two seconds. There are 32768 ticks per second for RTimer.

MATLAB functions which might be useful:

- *dctmtx()* gives the matrix representation of the DCT. This can be used for studying the symmetric structure present in the matrix.
- *fi()* for converting a real number to fixed point number.
- *dct()* for DCT transform of a signal.
- *idct()* for inverse DCT transform to get back signal.

4 References

1. Discrete cosine transform http://dsp-book.narod.ru/BYPRDCT.pdf

2. Fast algorithms for the discrete cosine transform
   http://homepages.cae.wisc.edu/ ece734/references/feig92.pdf

Matlab code for experimentation

```
1
2  close all; clear all; clc;
3  N=512; % signal length
4
5  M=120; % select the number of DCT coefficients
6
7  %% Load the test signal
8
9  load mit200
10 x=ecgsig(1:N,1);
11
12 %% DCT transform of signal
13 y=dct(x);
14
15 %% Select first M coefficients
16 y(M+1:end)=0;
17
18 %% reconstructed signal
19 x_rec=idct(y);
20
21 plot(x);
22 hold on;
23 plot((x_rec), 'r.');
24 legend('Original', 'Recovered');
```