Assignment 1: Implement Data Compression in TelosB Mote or using Cooja

Data compression is a typical method used in WSNs to reduce the sensor data size and transmission cost, however, data compression will cost some processing time and energy. To achieve the goal of energy saving, the implementation of efficient data compression is of great importance. In this assignment, student will learn how to implement a data compression method and what can impact the compression time.

1 Data Compression based on Discrete Cosine Transform

Discrete Cosine Transform (DCT) is a widely used transformation technique that can transform time-series data or an image to its frequency domain and have a compact representation of the signal. After DCT, a small fraction of the data points contain most of the signal information and the rest of the data points in the frequency domain have very small values which can be removed without causing significant distortion in the reconstructed signal. Therefore, a sensor node only needs to send the data points with large values and achieve the goal of data compression.

DCT-II transform of a length L signal $\mathbf{x}$ is given as,

$$y_k = \sqrt{\frac{2-\delta(k)}{L}} \sum_{n=0}^{L-1} x_n \cos\left(\frac{\pi}{L}(n+\frac{1}{2})k\right), \tag{1}$$

where y_k is the k^{th} DCT coefficient of the signal, $\mathbf{x}$, for a particular k. The function $\delta(k)$ is given as $\delta(k) = \begin{cases} 1 & k = 0, \\ 0 & elsewhere. \end{cases}$

DCT transformation can be also be rewritten in the matrix form as,

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

where $\mathbf{H} \in \mathcal{R}^{L\times L}$ is a DCT matrix.

The signal reconstruction process is as follows

$$\mathbf{x} = \mathbf{H^{-1}}\mathbf{y}, \tag{3}$$

where $\mathbf{H^{-1}} \in \mathcal{R}^{L\times L}$ is the inverse DCT matrix.

1.1 Compression Encoding

- Take an input signal $\mathbf{x}$ of length of signal N, use discrete Cosine transform (DCT) of length L, the number of DCT retained coefficients is equal to M. Note that $L \leq N$, it is usually to let N be a multiple of L, for example, $N = 256$ and $L = 8$, 16 or 32. It is clear that $M < L$, the compression ratio is L/M (Note that here compression ratio is defined as the original data size divided by the compressed data size.).

- Take an example of an 8×8 DCT matrix which is represented by $\mathbf{H}$. (In this case, $L = 8$.)

- To compress a signal of length N, you need to divide the signal, $\mathbf{x}$ into small blocks $\mathbf{x}_i$ and each of length L, and then apply DCT transform on each signal block $\mathbf{x}_i$. The transformed signal of each block is represented by $\mathbf{y}_i$.

- Select the first M coefficients of the transformed signal $\mathbf{y}_i$.

- The M coefficients of each signal block are the data that need to be transmitted.

1.2 Compression Decoding

- For each signal block, construct the received signal by using the M DCT coefficients and padding with $L - M$ zeros.

- Reconstruct the signal using the inverse DCT matrix $\mathbf{H}$.

1.3 Task

To implement the Compression Encoding in TelosB Mote. In the implementation, the DCT matrix $\mathbf{H}$ can be pre-stored in the TelosB mote or is generated every time performing data compression. Try to see what the difference is, which implementation is better and why. For a signal of size, e.g. $N = 512$, please try different values for L and M, then study the impact of L and M, in terms of processing time, compression ratio, and reconstructed signal quality.

Note that reconstruction can be done in laptop.

It is worth mentioning that DCT can be implemented in many different ways. Besides the basic implementation, there are also more efficient implementations, for instance, a fast DCT implementation in [1].

An example of matlab implementation of the basic compression encoding and decoding is given, in which an ECG signal is used as the input signal. Fig. 1 shows the original ECG signal and Fig. 2 shows the reconstructed signal and the original signal.

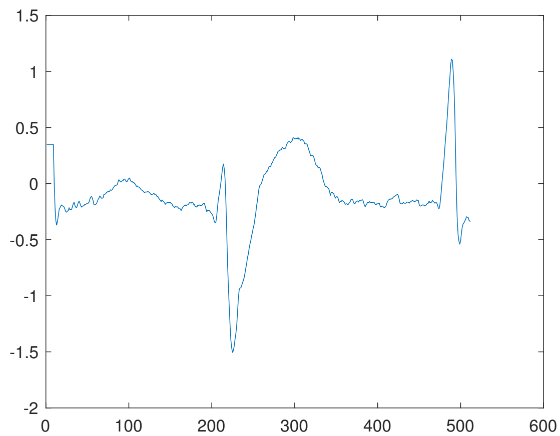

Figure 1: Original ECG signal

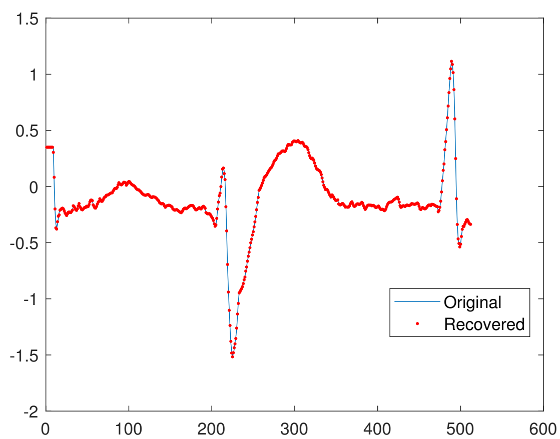

Figure 2: Original and reconstructed signal for DCT

References

[1] Fast DCT 8x8 algorithm C code: https://www.nayuki.io/res/fast-discrete-cosine-transform-algorithms/fast-dct-8.c