# Report on assignement 4 of Data Capture and Processinig course

Hamza Omari[1*]

[1*]faculty of science and technology, University Paris est Créteil, Créteil, 94100, France.

Corresponding author(s). E-mail(s): hamza.omari@etu.u-pec.fr;

**Abstract**

In this report we will explore the cross correlation method used in signal processing, especially for finding heart beats in ECG signals, based on a certain heart beat template followed with thresholding technique to find peaks.

**Keywords:** Thresholding, ECG, cross correlation

## 1 Signal acquisition and display

Based on the provided signal file, with a `CSV` format, that containes the sample values of the signal we will be dealing with.

Plotting this signal will give us the plot in the figure 1

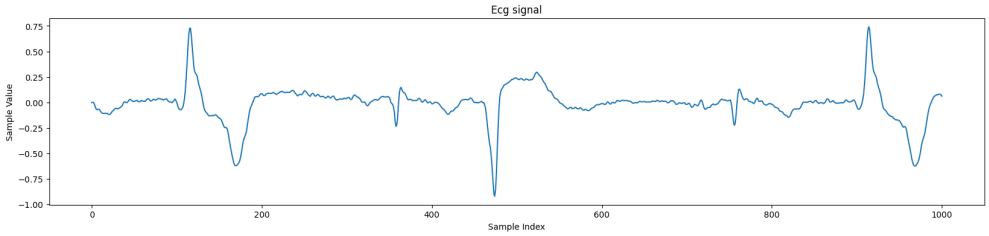


**Fig. 1**: Ecg signal

Next, for using the cross correlation further, we would like to take a template from the ecg signal. This template is the slice of the signal from 400 to 600

plotting this template will give us the following plot in the figure 2

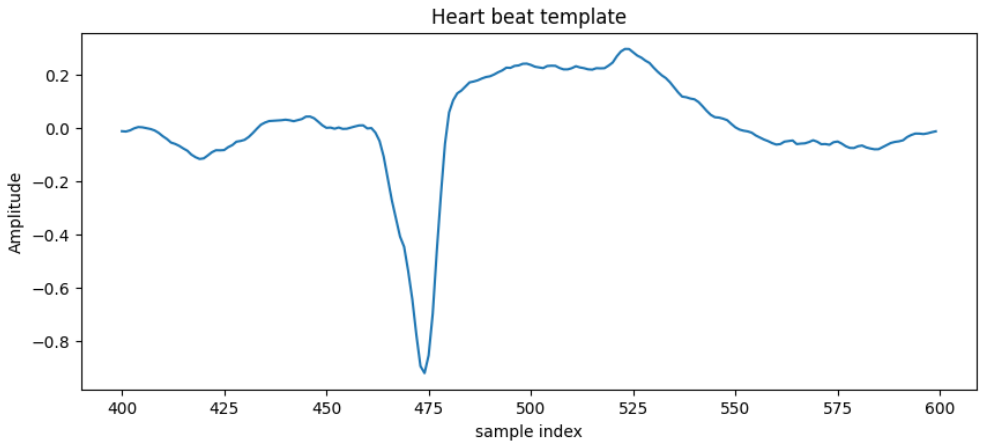


**Fig. 2**: Template

## 2 Cross correlation implementation

Unsing the `correlate` funciton from scipy library and the signal module, we would compute the correlated signal with the tamplate, the following in figure 3

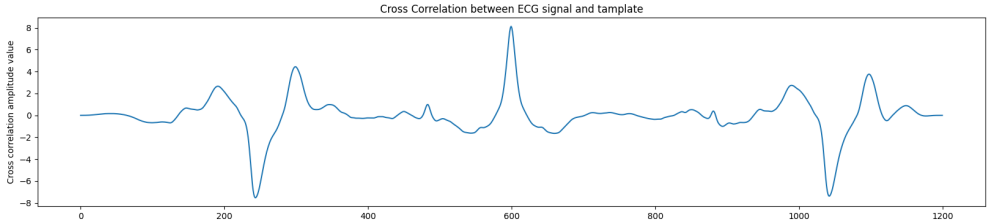


**Fig. 3**: Cross correlation of the signal with the template

## 3 Thresholding for peak detection

Using the find peaks function from signal module, we will test different threshold values to accurately find the desired peak.

the tested values are :

- $0.5 * \max(\text{signal})$ : failed

- $0.75 * \max(\text{signal})$ : passed

the peaks are found as displayed in figure 4, there is only one peak, and indeed we have only one template that matches the template used in the cross correlation. So we would have only one.

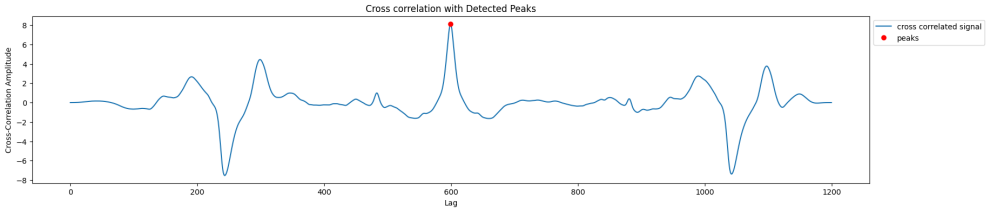


**Fig. 4**: Peak detection after cross correlation

# 4 Simulating Noise and Varying the SNR

In this section, we add white Gaussian noise to an ECG signal to simulate different levels of noise by implementing a function. first we calculates the noise power based on a specified SNR in decibel as parameter. and adds this noise to the original signal.

We generate then noisy signals at SNR levels of 10 dB, 20 dB, and 30 dB. The original and noisy signals are then plotted as shown in figure 5

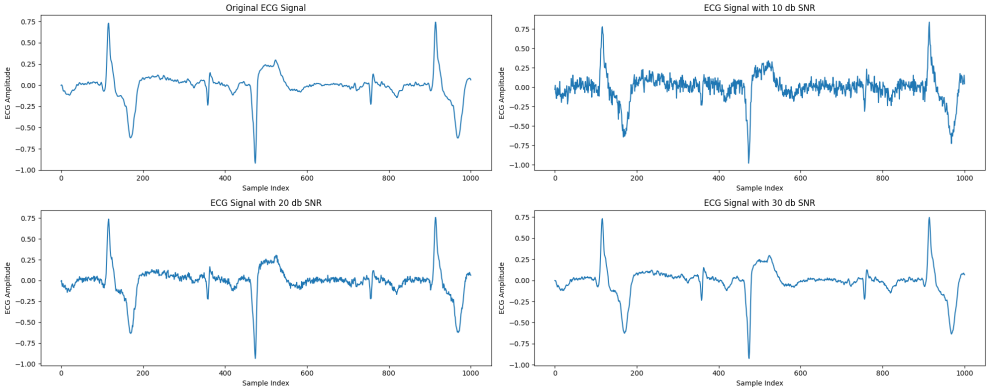


**Fig. 5**: Original and noisy signals

# 5 Detecting heartbeats with thresholding

Same as in section 3 for detecting the peaks. We will execute the peak detection algorithm along the noisy signals with different SNR levels. An array binary values over all the samples is also returned, for perfomance measurance in later section.

the output result in the following figure 6

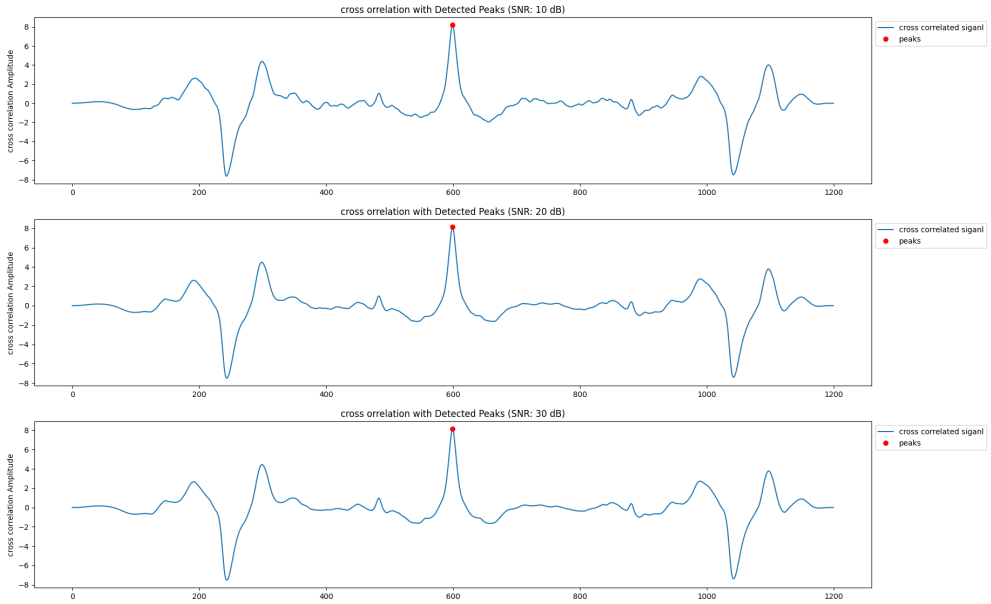


**Fig. 6**: Peak detection in the noisy signals

We can see that the peaks are correctly detected in the three noisy signals. Since that we can see in 5 the noise didn't alter completely the shape of the signals. Which means the used template in correlation and the template we are looking for in noisy signal don't differ a lot.

# 6 Analyzing with performance metrics

from the previous result, it is clear that we have accuracy = 1, True positives = 1, True negatives = number of the remaining samples -1. and we don't have any false positives or false negatives.

Computing and plotting these metrics alongside the snr levels will give us the plots of TP, FP, TN, FN in figure 7 and the accuracy plot in 8

# 7 Code and implementation

For more details, and all of the methods presented in this report, are implemented in a jupyter notebook using python and libraries like numpy, matplotlib and scipy .

**Notebook is accessible in this github repository: LINK**

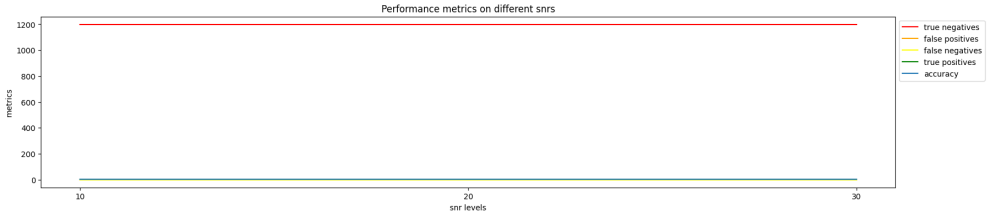


**Fig. 7**: performance metrics

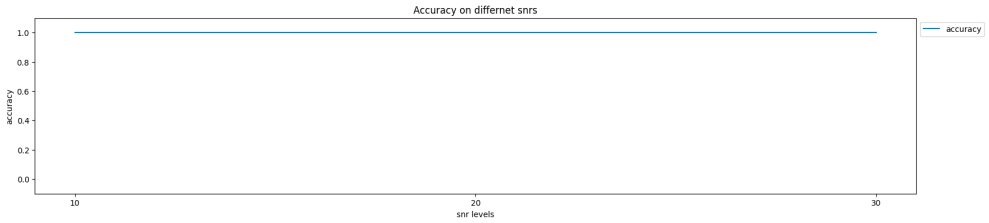


**Fig. 8**: Accuracy