Fixed Point Implementation of Pan-Tompkins

Hooman Sedghamiz

January 27, 2020

Abstract

This is an efficient and fixed-point implementation of the well-known Pan and Tompkins beat detector [1]. This code implements an updated version of the published article where the authors published corrections to a few filters employed in the paper. This implementation is implemented to be multi-platform and efficient for MCUs not designed for signal processing. Many optimizations have been performed to improve the implementation that are detailed in this brief.

1 Background

Pan and Tompkins proposed an accurate real-time ECG beat detector in 1985 that has been successfully used in many commercial devices [1]. The main stages of the algorithm are detailed below

2 Method

2.1 Preprocessing:

Steps:

1. Bandpass filter(5-15 Hz).
2. Derivating filter to high light the QRS complex.
3. Signal is squared.
4. Signal is averaged (MWI) to remove high frequency noise (0.150 seconds length).
5. Depending on the sampling frequency of your signal the filtering, options above are changed to best match the characteristics of the ECG signal

In preprocessing stage, the signal is passed through 5 filters namely low pass ($\approx< 15Hz$), high pass ($\approx> 5Hz$), 5 point derivative, squaring and moving average window respectively.

The implementation of these filters is very cruicial both in terms of end results and computational load. Therefore, the following optimizations are performed:

- No use of convolution: Filters are implemented based on their difference equation making them recursive and minimizing the number of operations.
- Additional care has been given to overflow control, while there is no guarantee that an overflow would never happen but several gain control checkpoints are implemented. Note that still the range and resolution of ADC plays a crucial point in filter outputs.
- Almost no division nor multiplication is employed and all the operations are performed with bit shifting, making the filters extremely efficient.
- Filters are implemented in both Direct I and Direct II form which are easily selectable in the header file.

2.2 Signal analysis and peak detection:

After signal enhancement, both integrated signal (output of moving average filter) and band-passed signal (output of HP filter) are simultaneously analyzed for beat detection. One of the advantages of this code is the implementation of search-back and T-wave discrimination without using any buffers as described below:

- Search-back: Once no beat is detected for 166back needs to be initiated to find the potentially missed beat. In order to avoid storing a buffer, the code only stores the tallest peak which was previously classified as noise, once there is a need for search-back the stored peak is compared against thresholds and therefore no buffer is required.
- T-wave discrimination: If a beat is close than 360msec to the previous detected beat, a slope test to the previous beat is employed to check whether it is a T wave. In order to avoid storing a buffer, the code saves the largest peak in the differentiated signal every time a beat is detected. Once a new beat is suspicious of being T wave, that stored slope is employed.

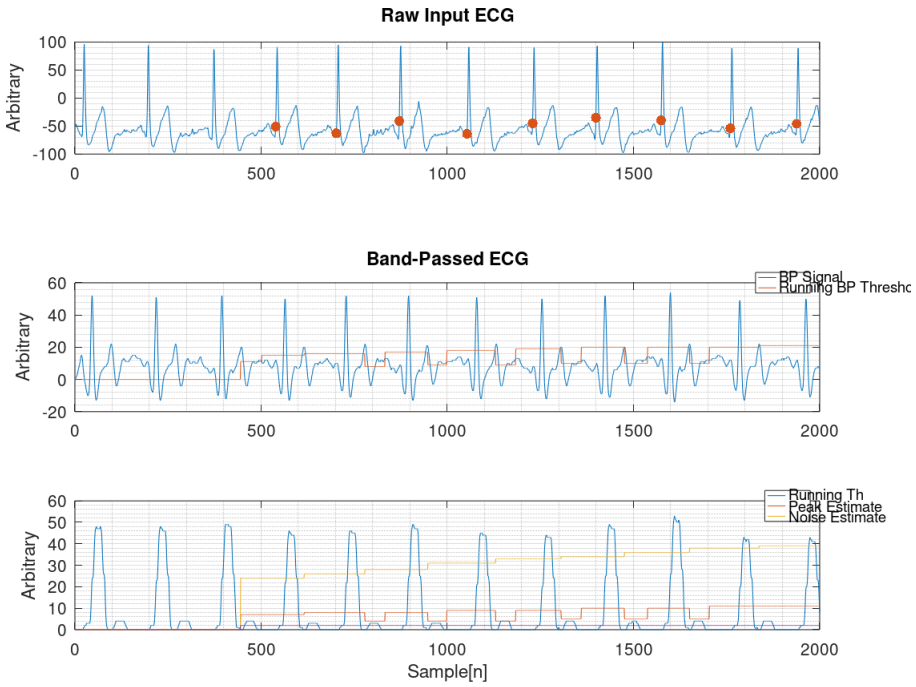
2.3 Decision making:

The algorithms stores 2 separate thresholds namely one for band-passed signal and one for the integrated. Each of these thresholds is adaptively updated based on detecting noise and signal peaks. Once a peak is above threshold in integrated signal and also the corresponding peak in BP signal is also above threshold that peak is classified as a beat.

3 Implementation

The whole beat detector is implemented in a single C and header file namely *PanTompkins.c* and *PanTompkins.h*. Additionally, for testing and debugging purposes, a compiled and executable (only for 64bit windows OS) C file is provided as well (*PanTompkinsCMD.c*).

The main process of the code is implemented as a struct making it easy to access and debug the program on the go. There are various so called *get functions* are also provided that



let the user query any parameter in real-time from the main algorithm. This is illustrated in (*PanTompkinsCMD.c*).

The code is written in a sequential and organized manner where each single step is implemented as sub-module for easy access and understanding. The use of the code in any application is very simple. The user only needs to include the header file in their main application and then first call *PT_init()* to initialize and then recursively call *PT_StateMachine((int16_t) sample)*, where *sample* is the latest ECG sample acquired from ADC.

PT_StateMachine routine implements the state flow of the Pan Tompkins algorithm and calls the other subroutines such as filtering and decision making units. Please refer *PanTompkins.c* and *PanTompkinsCMD.c* for detailed documentation and example use.

An example output of the code is illustrated below.

References

[1] J. Pan and W. J. Tompkins, "A Real-Time QRS Detection Algorithm," *IEEE Transactions on Biomedical Engineering*, vol. BME-32, no. 3, pp. 230–236, 1985.

[2] P. S. Hamilton and W. J. Tompkins, "Quantitative Investigation of QRS Detection Rules Using the MIT/BIH Arrhythmia Database," *IEEE Transactions on Biomedical Engineering*, vol. BME-33, no. 12, pp. 1157–1165, 1986.