Heartbeat Detection

Joseph R. Eoff

07/10/2015

1 Goal

Reliable heartbeat detector for a personal training app for Android smartphones.

Requirements:

1. Simple
2. Robust
3. Good noise rejection
4. No user intervention/adjustment required
5. Must use the microphone from a typical Android head set (wired earphone/microphone headset.)

2 Software

Development of an algorithm that can be implemented in Java, but capable of running in real time.

PureData will be used to develop the algorithm on a desktop PC. The final, functional algorithm will be translated to Java. Intermediate tests can be made using the PD patches from the desktop environment on Android with libPD.

2.1 Algorithm

The basic algorithm is a Hilbert amplitude detector, but with some pre and post processing based on observations of typical heartbeat signals. Figure 1 on page 3 is a block diagram of the method. It is also a valid PureData patch.

The preprocessing consists of a bandpass filter from 24Hz to 44Hz. This is based on the observation that a heartbeat consists of a sharp "snap" (broadband spike) and a low frequency "lub" at around 34Hz. The broadband "snap" is difficult to separate from broadband noise, but the low frequency "lub" is relatively easy to separate from noise.

The post processing begins a low pass filter to remove amplitude variations that can occur within the beat. In the next step, the logarithm of the instantaneous amplitude is calculated. The final step is a highpass filter with a cutoff of 2Hz. These three steps have the effect of only passing relatively sudden jumps in amplitude, and making the level of the jump independent of the absolute level. A highpass filter after the Hilbert transform by itself would result in pulses whose amplitude is proportional to the absolute level. Taking the log makes the amplitude of the "jump" more or less constant regardless of the true amplitude.

At this point, the amplitude of the jump is independent of the true volume. It is, however, dependent on the signal to noise ratio. A constant noise creates the equivalent of a DC offset in the instantaneous amplitude. The high pass removes that DC offset, leaving only the variations. The lower the signal to noise ratio, the closer the output of the high pass will be to zero. Amplifying the signal at this point allows the pulses to be detected at lower signal to noise ratios.

After amplification, a threshold function is applied to provide a clean indication of a detected beat.

This method is very good at picking pulses out of noise. At high noise levels, however, the beginning and end of the generated pulses will not be aligned with the peaks of the heartbeat pulses. Thus, at high noise levels you cannot calculate the heart beat rate from the time between pulses. Counting the number of pulses in a given time period (or the time for a given number of pulses) will give an accurate heart rate, however. This method can only measure heart rate variability at high signal to noise ratios.

This method has two disadvantages. One is the already mentioned random offset of the beginning and end of each pulse when the signal to noise ration is low. The other is that the amplification that allows the method to work in poor conditions also leads to false pulse detection when there is no signal. That is, if there is noise but no heart beat then this method will produce random pulses. It can therefore only be used when it is certain that there is a signal to be found in the noise.

3 Current state

1. Test program for Android that can use PD Patches. Audio input and output, minimal strip chart and heart rate display.

2. Functional PD patch that detects heart beats and counts the heart rate.

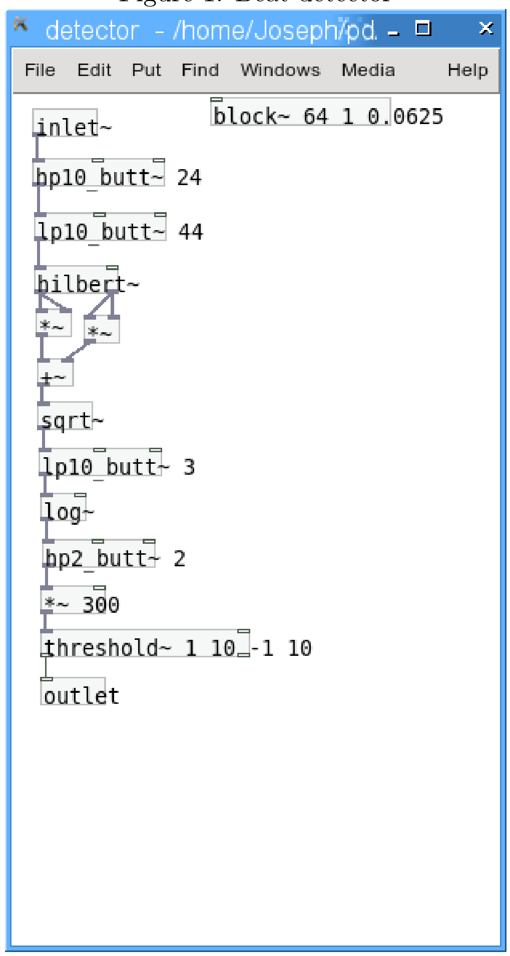
3. Requirements met:

   (a) Simple - beat detection is straight forward, runs on a slow Android phone, CPU load stays below 90%

   (b) Robust - insensitive to input level, hardware, frequency response

   (c) Good noise rejection - handles noise well. Heart beat is still well detected at 20dB signal to noise ratio.

Figure 1: Beat detector



(d) No user intervention/adjustment required - insensitivity to level and noise means that the user doesn't have to adjust anything for proper function.

(e) Hardware - works on the desktop system with a simple stethoscope as well as with a smartphone headset. No adjustment needed when changing hardware.

4 Results

4.1 Test program

All examples were created using the test program "DetectorTest.pd" with the file "CleanHeartbeat.wav"

"DetectorTest.pd" reads the input file, passes the signal through a simulator that attenuates the signal, high pass filters it, then adds in a specified level of noise. This simulates various input levels, noise levels, and the frequency response of varying hardware.

The simulated signal then goes through the same heartbeat detector as is used in the Android app. The simulated signal and the detected beats are then recombined into a single .wav file. The left channel is the simulated heartbeat, the right channel is the detected beat.

4.2 Test program examples

Figures 2 through 6.

The file "CleanHeartbeat.wav" contains a 30 second long recording of a normal heartbeat. The figures show the simulator settings used to create the example.

Input level is the attenuation applied to the original signal.

Noise level is the amount of noise relative to the attenuated microphone signal. If the input level is set to -20dB and noise level is -20 dB, then the noise is 40dB below full scale.

The highpass value gives the cutoff frequency for a highpass filter that simulates the typical frequency response of a smartphone. Some phones (such as the iPhone) have very low cutoffs while others may have a cutoff near 300Hz.

Figure 2: Input level 0dB Noise level -90dB Highpass 1Hz

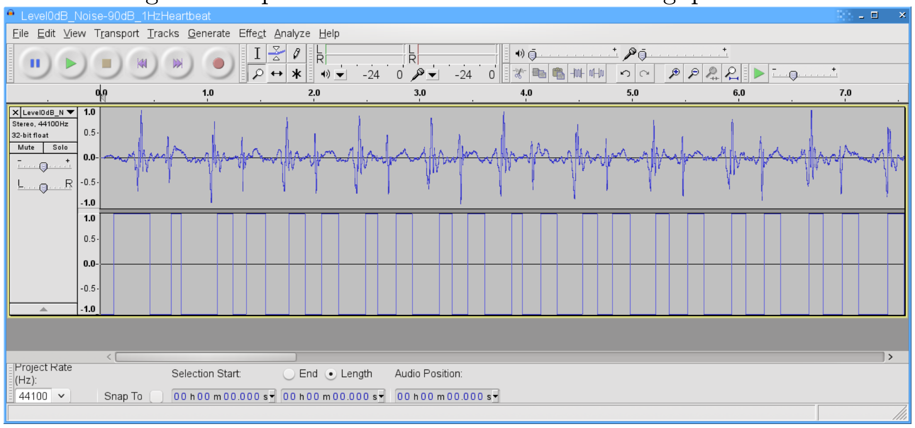


Figure 3: Input level -80dB Noise level -90dB Highpass 200Hz

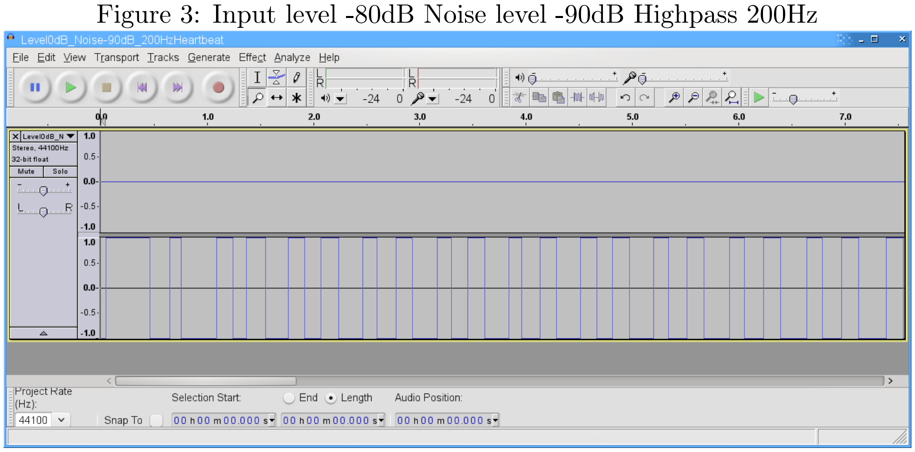


Figure 4: Input level -80dB Noise level -90dB Highpass 1Hz

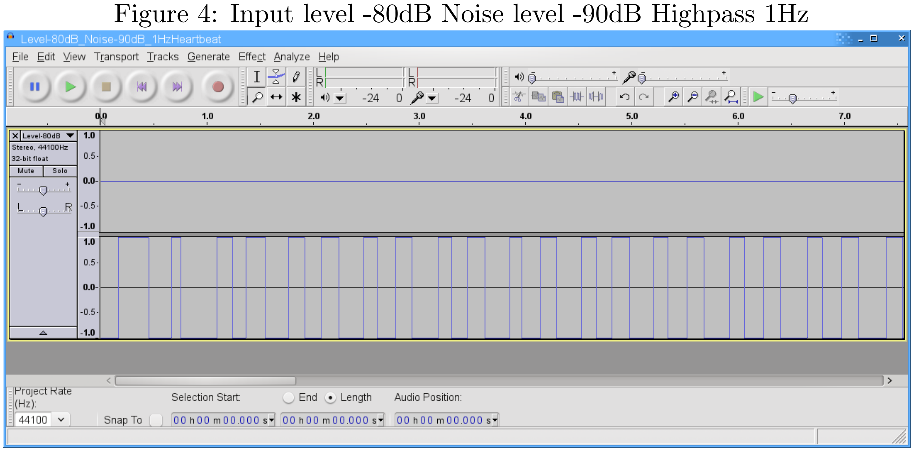


Figure 5: Input level -10dB Noise level -30dB Highpass 200Hz

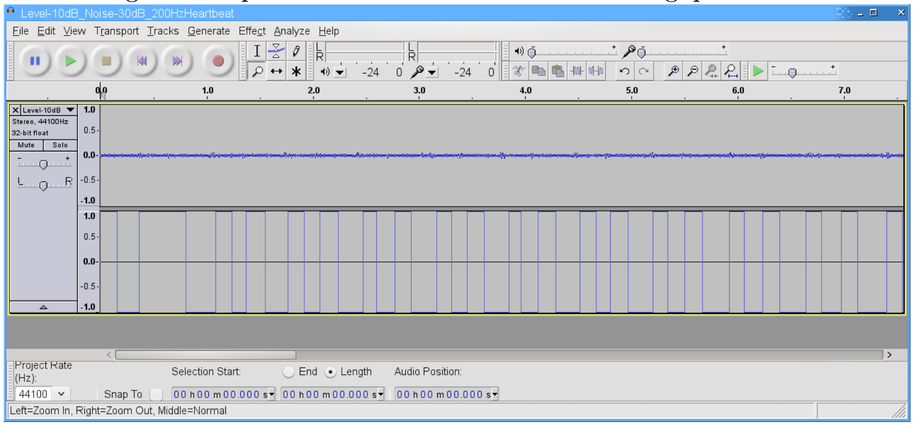


Figure 6: Input level -20dB Noise level -20dB Highpass 200Hz