A Statistical Method for Detecting Peaks
in Electrocardiogram Signals.

ABSTRACT: In the detection of relative peaks in an ECG signal, several techniques have been used. Among these are simple threshold and slope change detection algorithms. Here I present a statistical method for relative peak detection, which is not only more efficient and more effective than previous algorithms, but it also gives insight into the information content of the signal, as humans are believed to process it.

In the computer-aided analysis of ECG patterns, the information content of unprocessed digitizations of the signal is overwhelming, on the order of 250 data points per second, (21.6 million points per 24-hour period, the usual duration of signal analyzed). In an effort to condense and distill this information, several techniques are used. These include spectral analysis (Fourier transforms, primarily), etc. One of the more useful distillations of the ECG signal, used to detect QRS waveform distortions (APD and VPD's, mostly) and abnormal rhythm patterns (bigeminy, trigeminy, skipped beats, etc.), is a triangle transformation, which results in only 3 to 5 data items (triangles) per second in the processed signal, a 50:1 reduction compared to the 250 points per second in the raw signal. This involves representing each peak of the raw waveform by a triangle of some width and height (Figure A). From this distilled information, much of ECG analysis can be performed.

There are several methods for determining when a triangle should be output to represent an input signal deflection. These include simple fixed threshold\(^1\) and slope detection algorithms. The threshold algorithm 'detects a triangle' in the input signal whenever some fixed level is crossed (Figure B). The slope algorithm detects a triangle whenever the slope of the input signal becomes "steep enough" (Figure C).

Each of these algorithms has a major flaw which limits its usefulness in ECG signal analysis programs. The simple threshold is very sensitive to the fixed, predetermined trigger level chosen, and will not detect a small peak in a "quiet" area of the signal. The slope algorithm is computationally complex, and is sensitive to input signal noise content; in fact, it works like a threshold algorithm applied to the first derivative of the input signal.

\(^1\)Almost all peak detection algorithms exhibit some form of threshold detection. Some used fixed thresholds applied to the raw signal, others use slope thresholds (fixed limit on the first derivative of the raw signal). Of course, any of these fixed thresholds may be altered dynamically.
Here I present a statistical method with is computationally simple, corrects for the noise content of the input, is self-correcting (detects small peaks in a "quiet" area), and both is developed from and gives insight into the human pattern recognition process.
BASELINES

As seen in the above examples, most of the triangle detection algorithms depend on the existence of a baseline, a "zero" of the signal. In most ECG signals, this baseline is not given in the raw data; it must be calculated from the signal. It is this calculation upon which most triangle detection algorithms depend, and which most ignore as a trivial calculation. Here we address it as a separate, non-trivial issue.

There are several standard algorithms which calculate the baseline from a signal. Most of these algorithms are based on a mathematical low-pass filter, emulating the device normally used to average signals electrically. The baseline at a particular point is some arithmetic function of the N previous points. This device exhibits an exponential decay; data nearest the point being calculated have the most effect on its expected value, points further in the past have (exponentially) less effect (Figure D).

While this algorithm has a direct electrical analogue, there are simpler and more logical alternatives which have no such analogue. The most simple of these is a rolling average, which computes the expected baseline based on the average of the N previous points of the input (Figure E). Actually, what would probably be optimal would be a rolling least squares algorithm, since that best approximates visual "averaging" (i.e. how a human would "eye" the baseline in the signal), but the rolling average approximates the least squares reasonably closely, and is clearly simpler to compute.

The size of the rolling average window is important; too small a window would cause the calculated baseline to fluctuate wildly, and too wide a window would not follow shifting baselines (which DO occur in ECG signals) closely enough. Figure I shows comparisons of several window sizes. The width of the window is kept to a power of 2, to reduce the division in the averaging algorithm to a simple (and fast) shift operation (Figure F). Note that the size of
the rolling average algorithm is independent of the window size, which allows us to choose the window size which performs optimally, without regard to time/optimality tradeoffs. The optimal window size appears to be approximately one beat-width in duration, which again follows human analysis of the signal. A person, "eyeing" the signal, would expect the baseline of a sequence to depend mostly on the previous beat window.

**INITIALIZATION:**

\[
\begin{align*}
\text{sum} &= \text{first} \times N \\
\text{avg} &= \text{first}
\end{align*}
\]

**LOOP:**

\[
\begin{align*}
\text{sum} &= \text{sum} + \text{new} - \text{old} \\
\text{avg} &= \text{sum} \gg K \quad \{ \text{shift by } K = \log_2 N \}
\end{align*}
\]

Figure F: Rolling average algorithm

**THE "BAND"**

Now that we have a baseline to work from, we need to know when a triangle occurs. Our method of determining triangles should have the following properties:

1. It should detect small signal fluctuations as triangles, only if they appear in a quiet area of the signal.
2. It should reject signal fluctuations where the surrounding points vary greatly, since these most likely represent some sort of noise.
3. It should be expressed as some delta in relation to the baseline calculated as above. We know the signal rides these DC "shifts", so our detection algorithm should follow such shifts.

The standard deviation, also computed in a rolling fashion, gives a "band of expectation" around the rolling average. If the next point in the sequence is within this band, I can conclude (with a 68% probability) that the point is not a peak; I can say (with 68% probability) that this point does not "stand out" from its preceding points. If the point were to fall outside this band, I know (and I know to what probability — 68%) this point is a peak, and should be counted as a triangles in the output.

Again note that this algorithm also is independent of window size (Figure G). Note that the algorithm does not compute the rolling standard deviation directly, but computes the square of the standard deviation. This can be compared to the square of the signal height (signal - current baseline) to determine if the signal is within the band of deviation. Here I used ± 1 SD to give a certainty of ≈68%. This assumes that a simple bell curve applies to the deviations from the baseline, which is not necessarily true. In this case, however, it suffices as a crude (but effective) approximation.

**INITIALIZATION:**

\[
\begin{align*}
\text{sumsq} &= \text{first}^2 \times N
\end{align*}
\]
LOOP:

\[
\text{sumsq} = \text{sumsq} + \text{new}^2 - \text{old}^2 \\
\text{sdevsq} = (\text{sumsq} \gg K) - \text{avg}^2
\]

**Figure G: Rolling deviation algorithm**

Using the standard deviation as a band around the baseline gives noise immunity (since the band grows around areas of frequent fluctuations), and enhanced detection in periods of quietude (since the band narrows when the input is stable). Another advantage of this method is that we know the information content of the output, and can vary it as desired, i.e. change to ±2 SD's, with \( \approx \)96% certainty, or any fraction of the SD required to provide a desired certainty of peak detection. We know to what probability our output is correct, and can trade sensitivity for precision, just as expected in any detection system.

Some examples of the stability and versatility of this method are shown in Figure H.

**SPEED**

This algorithm, in full implementation, requires the following operations to maintain the current baseline and standard deviation band values:

- 3 integer multiplies
- 2 integer additions
- 3 integer subtractions
- 2 7-bit shifts (where \( N = 128 \))

The final algorithm, which converts the 250 point/sec raw input signal to a file of triangle encodings, takes 274 sec / 7.9 million data points (representing 4.5 hours of signal). At this rate (57x real-time), a 24-hour signal could be processed on a SUN-2 in 25 minutes, close to the targeted speed for this phase of the algorithm on a SUN-3 (a faster machine).

**Notes:**

1. width at 'a' reduces spurious peak detection, only 2 large peaks are detected (b,c).
2. small peak at 'a' ignored due to large recent beat, peaks near 'b' seen since later.
3. larger peaks at 'a','b' are detected (compare to 2 'a').
4. narrowing at 'a' allows peaks 'b'-'g' to be seen. Note 'cleanliness' of peaks seen.
Figure I: Varying window sizes

Note: widening of band at 1,3,5,7,9,11,12; narrowing of band at 2,4,6,8,10; and lack of narrowing at 13. Baselines as well as band sizes fluctuate as a function of the window size. Also note that baselines of 32 and 64-point windows fluctuate wildly, while 128 and up are relatively stable, thus the choice for the stability and narrowing of 128-point window, which is also the approximate point-width of a heart-beat.