This is to certify that the

thesis entitled

SIGNAL PROCESSING FOR A NON-CONTACTING HEARTBEAT DETECTOR AND ESTIMATOR

presented by

Shawn David Hunt

has been accepted towards fulfillment
of the requirements for

M.S. degree in Electrical Engineering

Maurice Bragd
Major professor

Date MAY 17, 1989

SIGNAL PROCESSING FOR A NON-CONTACTING HEARTBEAT DETECTOR AND ESTIMATOR

By

Shawn David Hunt

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Department of Electrical Engineering and System Science

1989

ABSTRACT

SIGNAL PROCESSING FOR A NON-CONTACTING HEARTBEAT DETECTOR AND ESTIMATOR

By

Shawn David Hunt

An algorithm for a non-contacting heartbeat detector is described. This algorithm is used in a system consisting of a microwave transceiver, an analog section and a digital section. The algorithm is responsible for detection and estimation of a heartbeat using the microwave return signal. It must be able to distinguish heartbeats mixed with clutter from clutter alone for detection. After detection, the signal with heartbeats has neither a known or constant shape and period. The algorithm used combines non-linear filtering with partial correlation and pattern recognition. The non-linear filter is used to produce non-zero regions where heartbeats are likely. One of the non-zero regions is chosen as being most likely to be a heartbeat and is correlated with the other non-zero regions. If there are non-zero regions that are similar in shape and almost equidistant, these regions are said to be heartbeats.

ACKNOWLEDGMENTS

I would like to thank Albert Roseiro and Dr. Marvin Siegel for their support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Introduction

This project is concerned with the remote detection of vital life signs in human subjects. The purpose is to develop a non-contacting heartbeat detector. The overall system consists of a microwave transceiver, an analog section and a digital section. The transceiver generates, transmits and receives a microwave signal. The analog section amplifies and filters the microwave return signal. The digital board then samples this signal and processes it to determine the heart rate. This thesis describes the different methods tried for solving the detection and estimation problem, and analyses one of the digital signal processing algorithms used for detection and estimation.

This research was done under a grant from the United States Navy which defined the goals of this project. They want a compact, portable instrument for the detection and estimation of heart rate. This is needed for military personnel to quickly identify in combat situations the heartbeat of battle field casualties and estimate it's value.

A microwave signal at 10 GHz is sent out, and after detection, it is band-pass filtered from four to fifteen Hz and digitized at a 512 Hz sampling rate. From the return signal, the system must decide if a heartbeat is present, and if it is, the heartrate. The algorithm works only with the sampled signal, thus when a heartbeat is referred to, it means the digitized waveform of the return signal containing a large chest movement due to the heart pulsation. To detect the heartbeat, the microwave transceiver is aimed at the chest, for details see [1]. Along with the heartbeat, breathing motion and other physiological clutter is returned. The heartbeat, if present, must be distinguished from this noise. This problem is interesting and difficult because of the number of

variables, how the variables change, and the amount they change. This problem is not new and has been attacked before [2,3].

The returned microwave signal gives information about movement of the chest wall. The motion of the heart is attenuated and distorted at the chest surface. Also, the received signal will be different because of differing physiology. Not only will heartbeats from person to person be different, but the shape of the waveform from heartbeat to heartbeat of the same person will be different. The heart is also naturally pseudo-periodic. It may remain almost constant for some time or it may change period dramatically in a very short time. The algorithm then must be able to distinguish clutter from the heartbeat, make a decision of whether a heartbeat is present, and be able to follow a heartbeat with changing shape, amplitude and period. The qualities described make the detection and estimation inherently difficult, but the heartbeat also has features that should help, and can be taken into account when developing an algorithm. A feature that has been fundamental in the algorithm implemented, uses the fact that the heart is impulsive in nature. This helps in distinguishing it from periodic and pseudo-periodic signals that are slowly varying. The problem in detection and estimation is that a clean signal with a high signal to noise ratio cannot be guaranteed. This is the main reason for the effort put forth to develop good filters and processing.

The microwave signal that returns after reflecting off the chest has information of the heart rate mixed in with ambient as well as physiological clutter. In this case everything not pertaining to the heart is referred to as clutter. Because of the clutter and variability, having an algorithm work under normal conditions without any pre-filtering is difficult. Thus methods such as peak detection or correlation cannot be used reliably without pre-filtering. The first thing to be done was to find or develop a filter that can

reduce the level of clutter while leaving enough heartbeat information in the signal to make the detection/estimation problem easier to solve.

The organization of the remaining sections is:

(a) A summary of the efforts of others involved in this research before me, and my own previous efforts.

(b) A summary of the algorithms tested for non-linear filtering.

(c) A description of the derivative filter.

(d) Justification for different parameters.

(e) Detection and estimation after the non-linear pre-filtering. Different parameter settings are justified and typical results are given.

(f) Results to compare the final algorithm with four other algorithms.

(g) Conclusions.

The appendices contain the implemented algorithms, written in C, and some sample four second data segments used for the result section.

Algorithms Previously Investigated

The first algorithm implemented was autocorrelation of a four second segment [4,5]. Because the heartbeats are not purely periodic, and vary in shape, this method does not perform as well as is needed. An example of a segment and it's autocorrelation are shown below. As is apparent, if the signal is easy to pick out visually, and is almost periodic, autocorrelation will perform well. However with a poor signal, as that shown in figure 1,the autocorrelation method does not work.

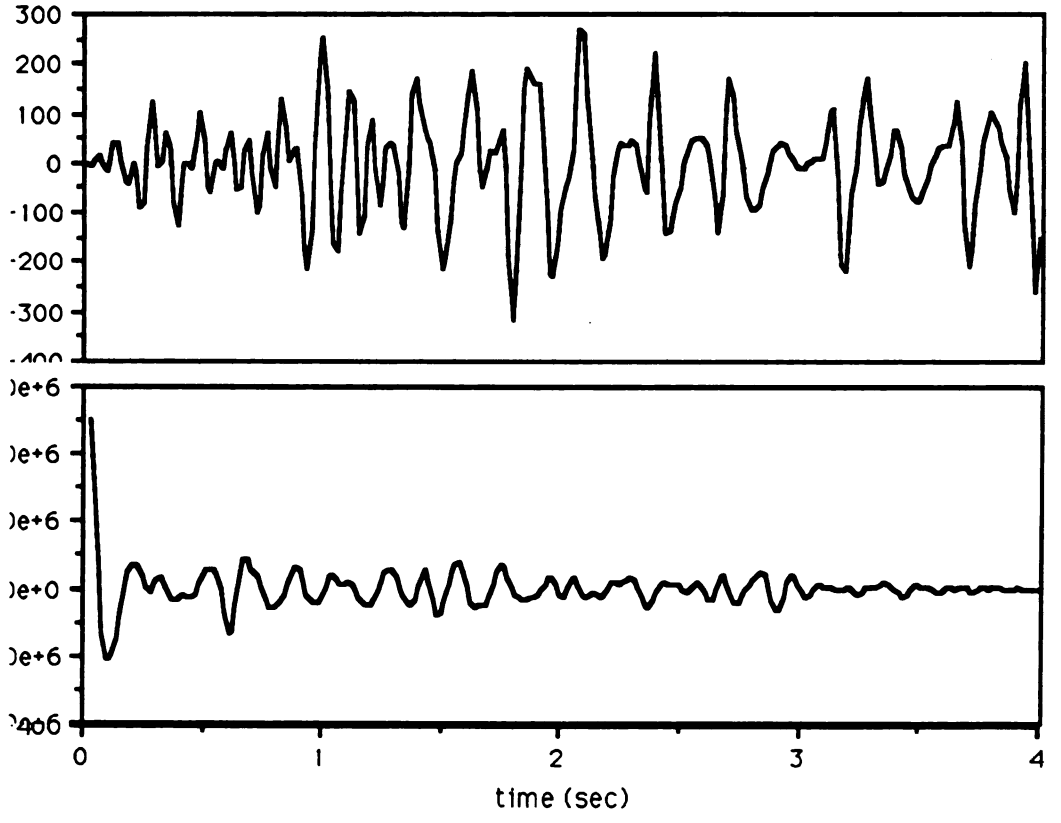


figure 1: Heart data and it's autocorrelation.

W. Byrne used an adaptive filter approach [6,7]. Pat Mahoney is presently continuing this work on an adaptive filter algorithm.

G. Hoshal implemented a number of algorithms [8]. One algorithm used a spectral approach in the frequency domain, one used a comb filter in the frequency domain [9,10], and another used features to form a multi-dimensional classification space [11].

The spectral approach uses the Fast Fourier Transform of the input signal and estimates the heartbeat frequency from that. It was found that the breathing interfered if the signal was not filtered by a low-pass filter. It was also found that the return signal due to the heart has many strong harmonics, but that due to the breathing does not. Because of this, the signal is band-passed filtered from 4 to 15 Hz. The fundamental component of the heart signal is blocked out along with the fundamental of the breathing. The heart signal is detected by it's harmonics. The breathing plays little part because of it's very small harmonics. This procedure is very effective with a strong heartbeat present, but is not good otherwise. He took the absolute value of the signal before performing the FFT.

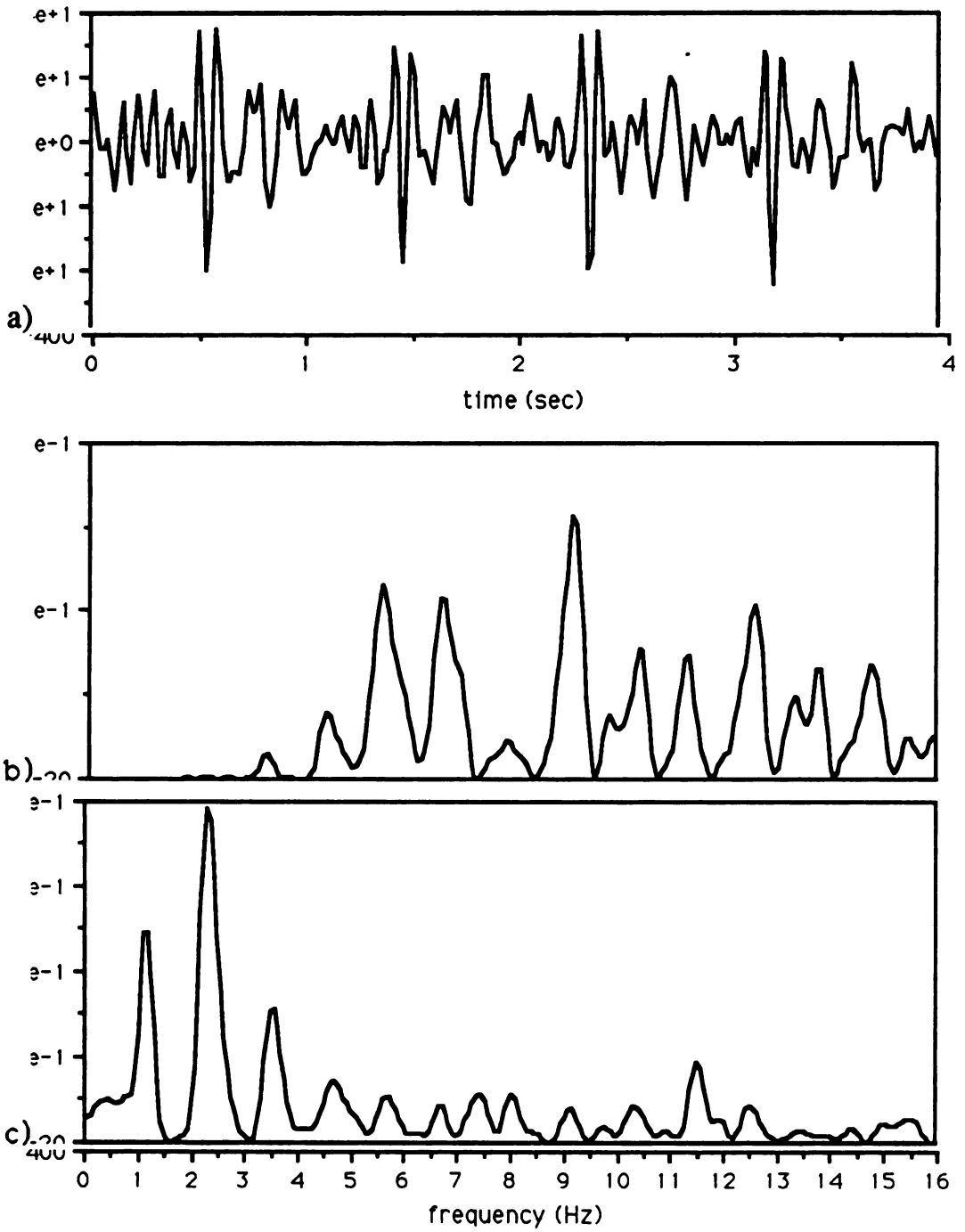


figure 2: a) Heart data, b) power spectral density, c) psd of absolute value of data.

Another approach Hoshal used was an adaptive comb filter. It used adaptive signal filtering to select the best fit and from this determine the heartbeat.

The final approach taken by Hoshal was to select several features as different dimensions of a classification space. Data from many subjects was gathered to determine the thresholds for decision regions in this space.Thus, the input signal could be processed to determine the features and from these a decision about the presence of a heartbeat is made. Many features were investigated, such as the average value of the power spectral density, and zero crossing count.

New Algorithms

Since the previous signal processing techniques were examined, a number of additional processing methods have been investigated. A statistical approach using correlation was tried first. This uses two features from the autocorrelation to form a two dimensional decision space. The two features selected were the sequential distances between the three largest peaks. The algorithm however will work with any feature from the autocorrelation. After the two features of the signal autocorrelation are selected, these features are computed from as many heartbeat files as possible. From the pool of these two sample features, a correlation coefficient and a regression line was determined. When new data is acquired, it's autocorrelation features are compared with the regression line and a decision is made about detection. Because the features selected give a heartbeat estimate, both heartbeat detection and heartbeat estimation were be determined at the same time. This works reasonably well with a good signal and windowing. The windows work as follows. If it is decided to detect a heartbeat between 60 and 120 BPM (beats per minute) we can segment, or window, a section of the autocorrelation corresponding to this frequency. The corresponding frequency for 60 to 120 BPM is 1 to 2 Hz, or a 1 Hz difference. Windows would be placed at intervals corresponding to every 1 Hz. Separation between the largest peaks in every window should correspond to the heart frequency. The drawback in estimation range is immediately apparent. For non-overlapping windows, the highest frequency estimated is twice that of the lowest frequency. For example, if 50 BPM is the lowest frequency, then 100 BPM is the highest. However the largest drawback for this algorithm is

the correlation technique itself. To extract the wanted features, an autocorrelation must be done before any other processing. Practically, for autocorrelation to work, the signal must be periodic and the noise level must be small. The heartbeat is not always periodic, and large changes in heartbeat frequency in a short time can occur. Typically, four seconds of data sampled at 64 Hz (256 samples) have been taken. If in these four seconds the heartbeat remains approximately constant, and the signal is relatively noise free, the correlation will work. These are ideal conditions however, and will not work satisfactorily in most situations. Typical heartbeat data is shown along with it's autocorrelation in figure 1.

The difficulties with autocorrelation required new algorithms to be developed.

There is a similarity between estimating pitch in speech and estimating the heart rate. The problem of estimating the fundamental frequency of speech has been widely studied and the ideas that worked well in that area can be easily tested to see if they have an application for the heartbeat estimation problem. Four main algorithms were tested.

Methods using zero crossing count and zero crossing intervals seem good choices [12,13] because the zero crossing count is lower in the segments where there is a heartbeat, and the zero crossing interval is almost constant from heartbeat to heartbeat. An algorithm to determine heartbeat frequency by zero crossing interval was developed, and the approach had no problem with aperiodicity. However, it was found that the differences of the intervals of the signal containing a heartbeat and signal containing no heartbeat did not lead to a good detection or estimation scheme, as shown in figure 3.

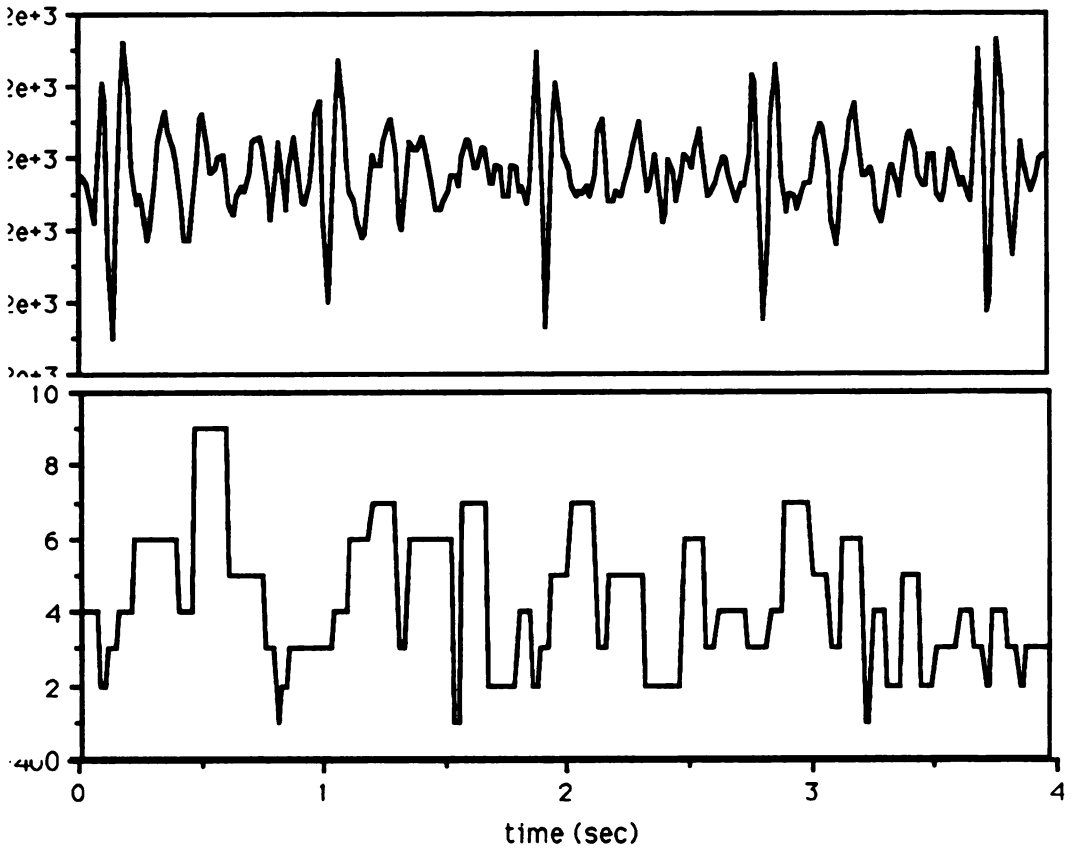


figure 3: Heart data, zero crossing intervals.

The zero crossing count did appear to lead to a reliable detection scheme. It was not used because it did not lead to a detection scheme and the algorithm used now does both detection and estimation.

The cepstrum approach was also tried [14]. The Fourier transform of the signal is taken. The square root of the power spectral density is taken and signals that are convolved can be separated if they are separated in the frequency domain. This approach is good for data where the noise is convolved with the signal. This was not the case and this approach showed no improvement over simply looking at the power spectrum.

The zero infinite clipping algorithm with autocorrelation was also investigated [14,15]. First, the mean is taken out of the signal. A threshold is

then determined, usually some percent of the average signal strength. All the elements above this threshold are assigned a one and all the elements below the threshold are assigned a zero. The signal is then autocorrelated. Although this method uses the zero crossing interval to some extent, it performed poorly because of the use of autocorrelation in a signal that is not periodic.

The last technique discussed in this section is a combination of non-linear filtering and frequency domain pattern recognition. First the 256 samples of the signal are filtered with a non-linear filter. 768 zeros are inserted after the data to make a segment of length 1024. An FFT is then taken. After the power spectral density is determined,an autocorrelation is done to detect heartbeat harmonics in the frequency domain. The process is shown in figure 4.

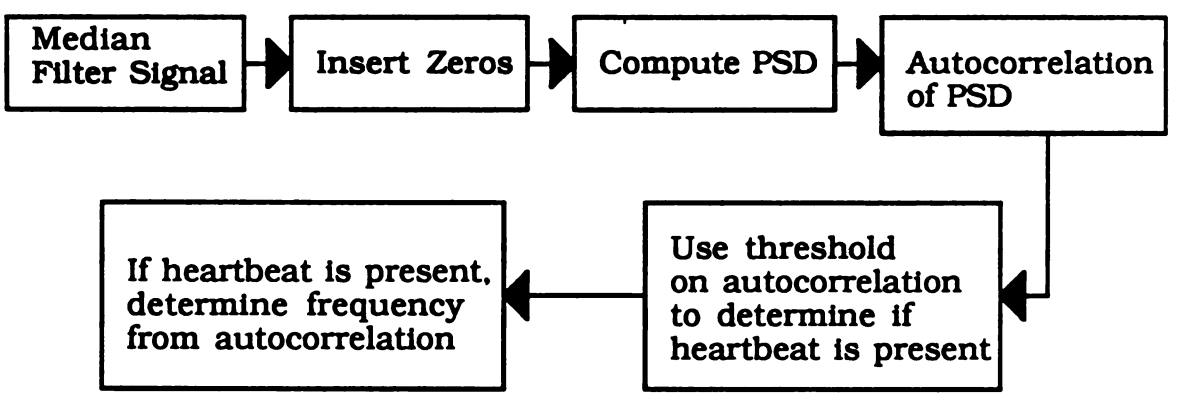


figure 4 : Block diagram of algorithm using median-PSD-autocorrelation

The non-linear filter used is the median filter. It's operation and sample input and output are shown in the next section.

After the filtering, the 256 data elements are zero stuffed to complete 1024 elements. The longer segment is used to give a precision within ±0.125

Hz. An FFT is performed and the power spectral density is determined. Because the heart is pseudo periodic and impulsive in nature, it has harmonics that are periodic in the frequency domain. The signal is band-pass filtered from four to fifteen hertz before it is sampled, so the fundamental frequency of the heart is almost entirely attenuated before the processing starts. The heartbeat harmonics remaining are used to determine the heartbeat frequency. This pre-filtering will eliminate interference with frequencies near the heartbeat fundamental, such as breathing. If they have very small or no harmonics above four Hz, they will not interfere with this detection scheme. The heartbeat harmonics are evenly spaced and the spacing is the fundamental frequency. An autocorrelation is done on the power spectral density. Looking for the highest peak in the correlation file corresponding to a fundamental frequency of .5 Hz to 2.5 Hz, a threshold is used to determine if a heartbeat is present. This method does not appear to suffer as much from an aperiodic signal as a straight correlation of the time signal. This change in period between heartbeats in the sampling time will result in wider harmonic peaks in the power spectral density. This will in turn lead to a less accurate estimate, but the estimate is still possible, something not seen in correlation.

An example of the PSD of the original signal and the filtered signal is shown in figure 5. The difference of the two large peaks in a) of figure 5 correspond to the fundamental heart frequency. Few peaks in the psd mean fewer peaks in the autocorrelation function file and choosing the correct peak is easier. If the unfiltered signal's psd and thus it's autocorrelation has more peaks, it makes the peak corresponding to the fundamental heart frequency much more difficult to detect as shown in figure 6.

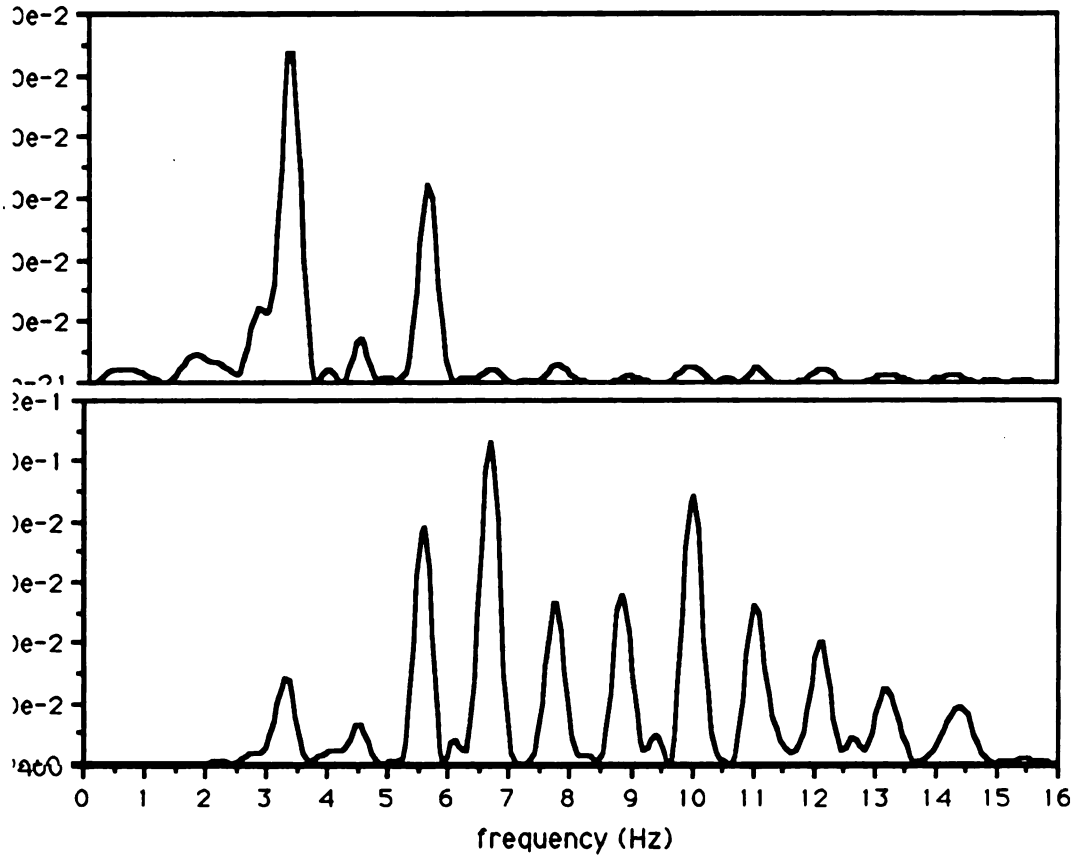


figure 5: PSD of median filtered (top) and unfiltered data.

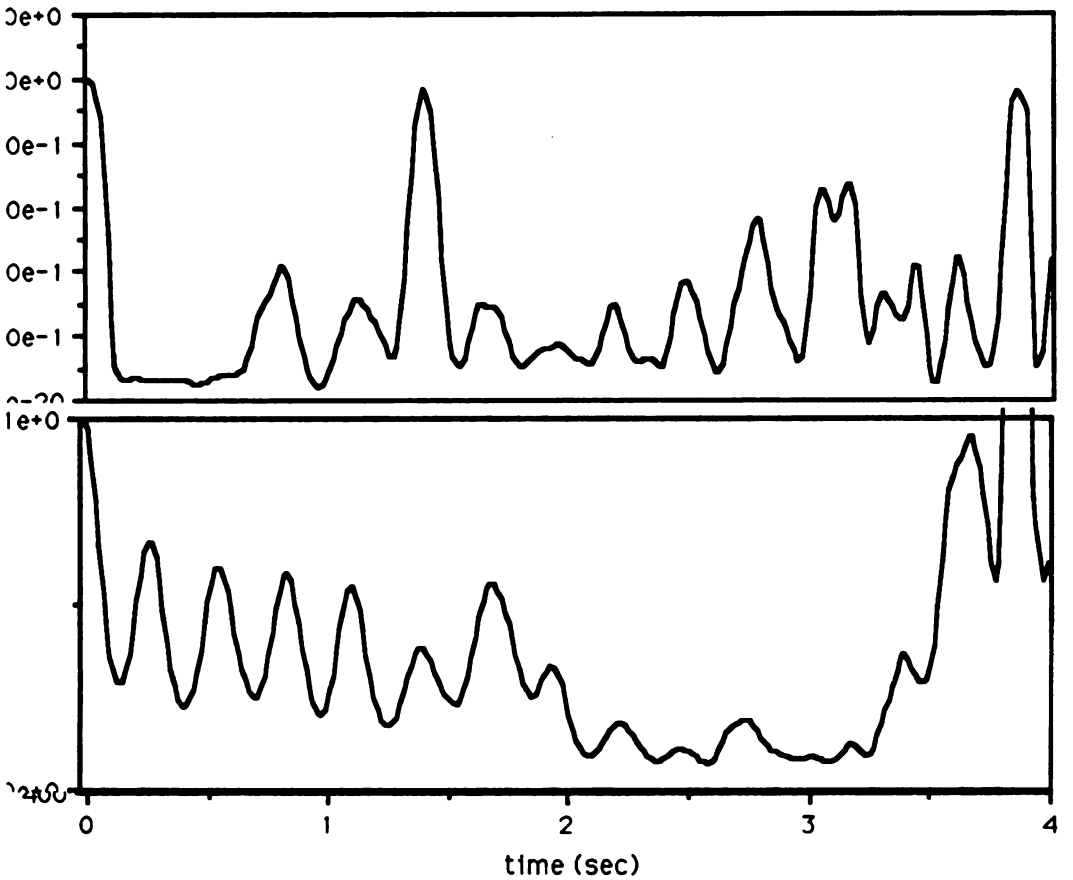


figure 6: Corresponding autocorrelation of signals in figure 5.

Pre-Filtering Algorithms

To make the heartbeats easier to detect in the clutter, pre-filtering algorithms were developed. Like the algorithms previously implemented, the filtering is done on a four second segment of data. In the analog section, a band-pass filter from 4 to 15 Hz is used, but the signal is still very cluttered and any additional linear filtering by the digital section is of little use. Of course the device cannot extract more information than is present in the signal,but we want the pre-filtering to reduce the clutter while leaving enough of the heartbeat signal so the detection and estimation can be done easier than before. A first try in using a non-linear filter was the median filter [16,17,18,19].

This filter is a special case of the L-filters. It works as shown below. In figure 7, each element is filtered using the surrounding elements.

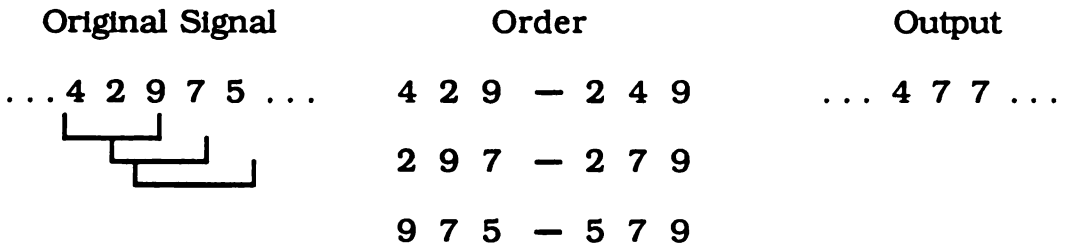


figure 7: Median filter operation.

The elements used in the determination fall into a window. This window is simply the number of elements used in the ordering. The median filter will preserve discontinuities in the signal if they are long enough, yet eliminate or greatly attenuate short discontinuities. Thus the window is chosen according

to the width of the peaks to be eliminated.The elements in the window are ordered in ascending order of magnitude. The element now in the center replaces the original element. This is done for each element in the data file. For elements on the end, the missing elements in the window can be chosen as zeros, or the value of the end element.

The effect of the filter with a window of 3 is shown in figure 8.

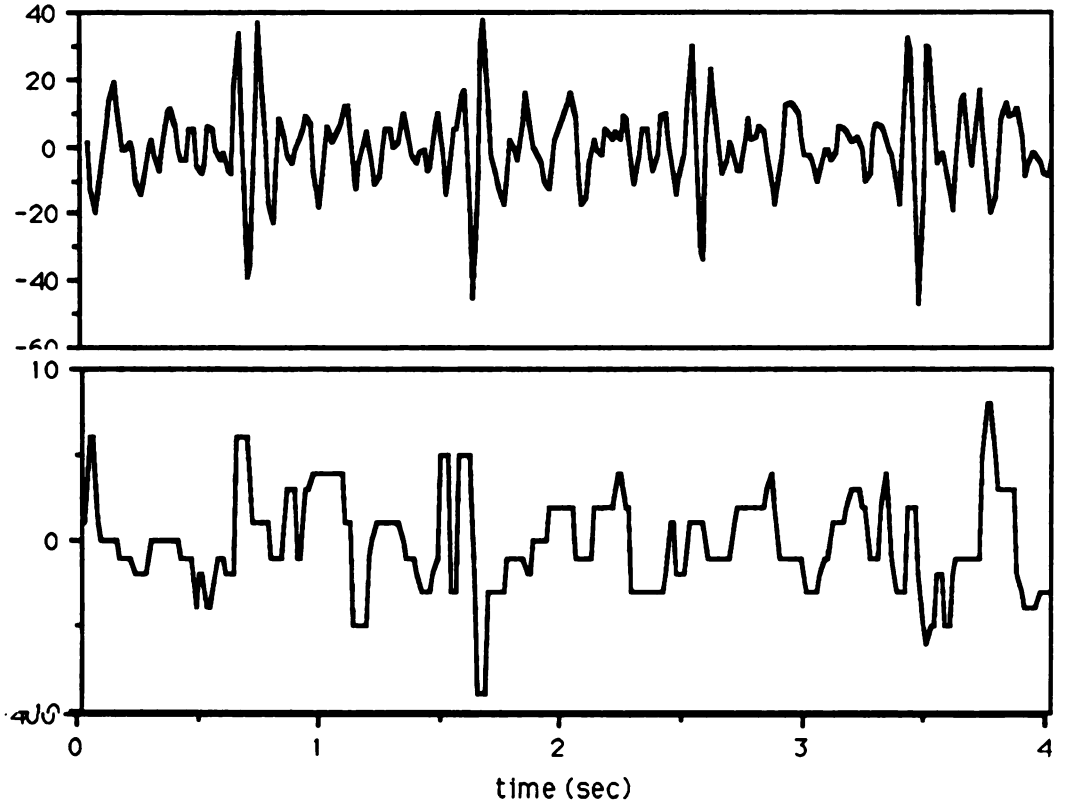


figure 8: Heart data, output of median filter.

This filter has the desirable feature that it keeps sharp edges of peaks wider than some threshold, while reducing peaks narrower than the threshold. It passes any peak wider than the threshold and it severely distorts the waveshape as shown. As previously discussed, this filter was used along

with frequency domain pattern recognition. This pre-filtering did improve detection and estimation, but the complete algorithm did not perform adequately. After this attempt, detection and estimation based on shape was investigated. Because the median filter did not generally make the signal easier to detect visually, it was not used. It did show that non-linear filters have promise, and gave reason to pursue them further.

The intent was to filter the signal so it would be easier to detect. It did not matter if the signal shape was totally changed, as long as it emphasized the heartbeat signal. Many different methods were tried. The first and simplest was simply a threshold (peak detection). The next used only the direction of change not the amplitude. An adaptive filter using the error as the output was also implemented.

The first method set all the samples smaller than a threshold to zero.The threshold was determined as a percent of the largest sample in the four second segment. This method did eliminate a lot of clutter, but it also eliminated a lot of signal information and small heartbeats. It did not emphasize the heartbeat over the clutter already present in the signal. An example of input and output is shown in part b of figure 9.

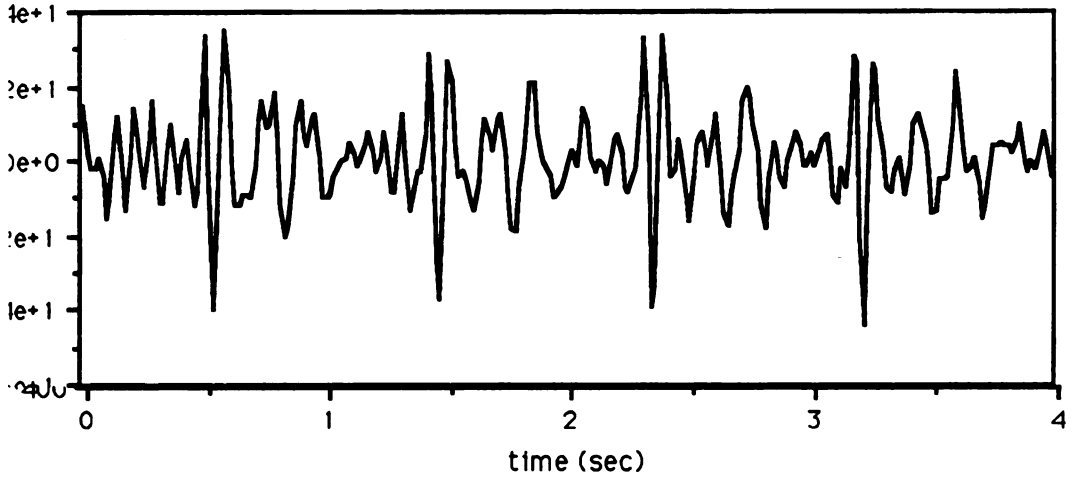
The next filter only used information of the direction of change from sample to sample. This filter starts at zero and goes up one unit if the direction of change from the first sample to the second is up, down one if the change is down and stays at zero if there is no change. Because there is only a change of one or zero each time, it removes information of amplitude change. It continues going up one, down one, or staying equal, according to the change from sample to sample. An example of this filtering is shown in part c of figure 9.

The problem is that the heartbeat can fall in the same range of frequencies, can have the same magnitude, and have no simple shape difference from the clutter. The solution found here was to go back to the basic assumptions about the heart.A feature of the heart is it's impulsive nature. It also must have enough amplitude to make it distinguishable in the clutter. We require it to be at least as large as the clutter. Since the derivative will give information about the impulsive nature, using information from derivatives seems like a good idea. Both the adaptive filter and derivative filter do this.

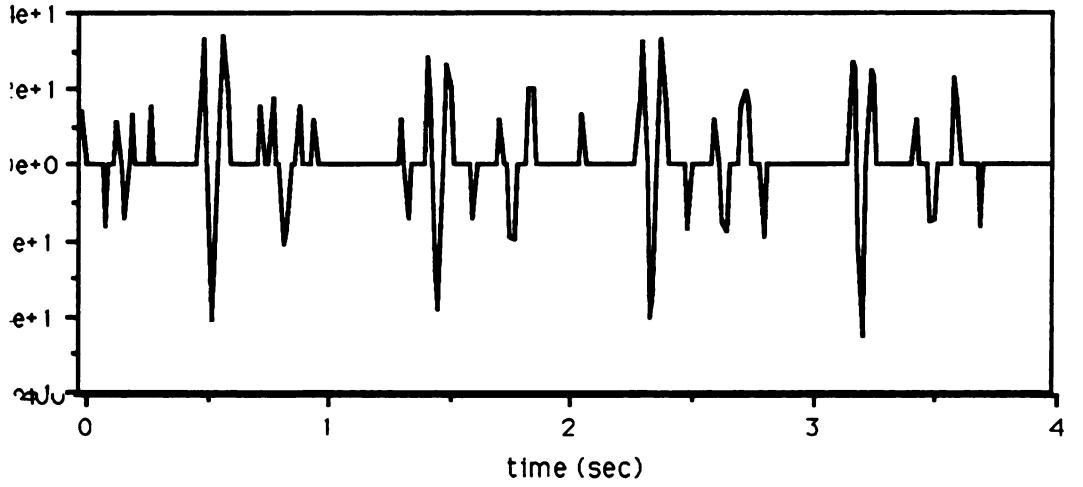
The adaptive filter is based on a difference equation where the coefficients are determined using the least square error method. Different order equations were tested, the fourth order was found to perform the best. Higher order equations did not improve the performance. An example output is shown in figure 9d.

The final test is how well it performs. The adaptive filter worked well, but because the derivative filter appeared to perform better, in this report, the derivative filter was used.

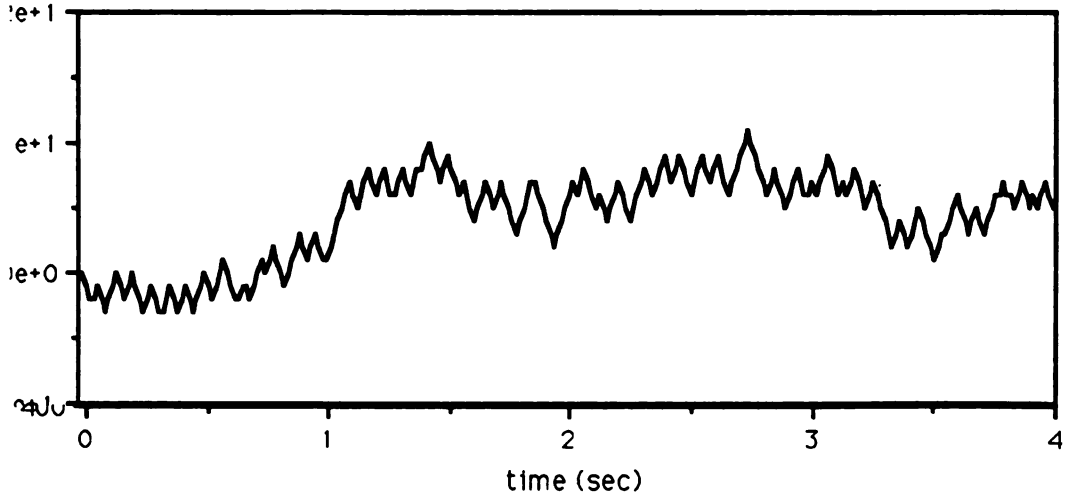
In the next section this derivative filter is discussed.



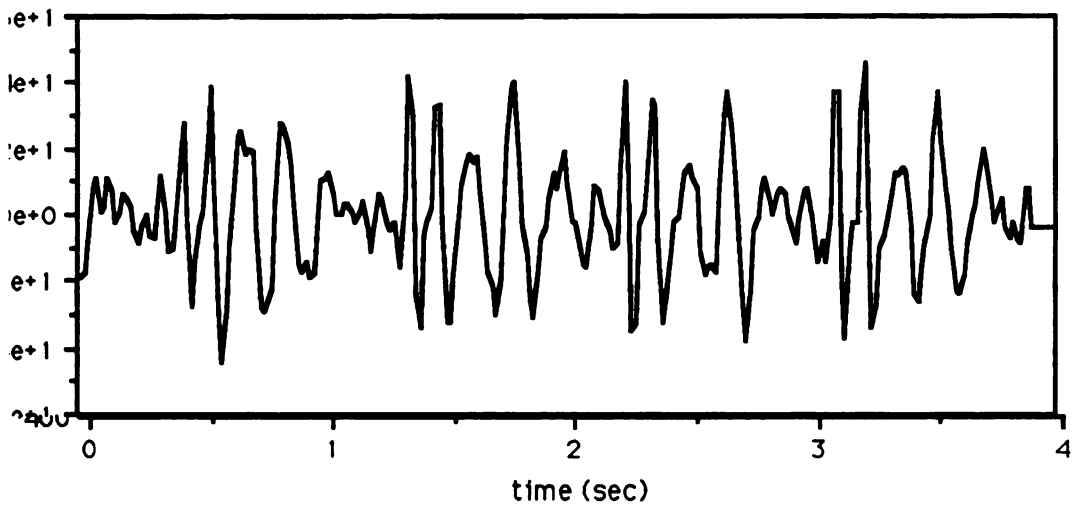
a:heart data



b:thresholded



c:discarding amplitudes



d:adaptive filter

figure 9: Example outputs of non-linear filters.

Derivatives of a discrete sequence

The derivatives, or the discrete equivalent of a derivative will be used extensively. To show how the absolute value in the derivative filter relates to the discrete derivative, the first and second derivatives of a sequence are shown.

Start with a Power series expansion of a function f(z) about a point a.

$$f(z) = f(a) + \frac{(z-a)}{1!}f'(a) + \frac{(z-a)^2}{2!}f''(a) + \frac{(z-a)^3}{3!}f'''(a) + \ldots \quad (1)$$

(1) can be put into two other forms.

first:

$z = x + h \qquad h = z - a$

$x = z - h \qquad a = z - h$

$x = a$

substituting into (1)

$$f(x+h) = f(x) + \frac{h}{1!}f'(x) + \frac{h^2}{2!}f''(x) + \frac{h^3}{3!}f'''(x) + \ldots \quad (2)$$

or:

$z = x - h \qquad z - a = -h$

again substituting into (1)

$$f(x-h) = f(x) - \frac{h}{1!}f'(x) + \frac{h^2}{2!}f''(x) - \frac{h^3}{3!}f'''(x) + \ldots \quad (3)$$

combining these we get:

(2) - (3)

$$f(x+h) - f(x-h) = 2f'(x)\,h + 2f'''(x)\frac{h^3}{3!} + \ldots$$

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - [\, f'''(x)\frac{h^2}{3!} + \ldots \,]$$

$$= \frac{f(x+h) - f(x-h)}{2h} + O(h^2)$$

(2) + (3)

$$f(x+h) + f(x-h) = 2f(x) + 2f''(x)\frac{h^2}{2!} + 2f^{iv}(x)\frac{h^4}{4!} + \ldots$$

$$f''(x) = \frac{f(x+h) + f(x-h) - 2f(x)}{h^2} - [\, f^{iv}(x)\frac{h^2}{4!} + \ldots \,]$$

$$= \frac{f(x+h) + f(x-h) - 2f(x)}{h^2} + O(h^2)$$

The error terms in our application however are actually not important since we do not want to estimate the derivative of the function, but merely use the estimate to make the heart signal easier to detect. Because the output signal will be used for detection and estimation, it is important to look at the performance of the proposed filter.

In the following sections, the derivative filter, and two variations will be introduced. How the filter and the variations reshape the signal is investigated. The conditions for the heartbeat peak to be higher than the surrounding noise is derived, and the conditions for the highest peak in the

pre-filtered signal to be the highest point after filtering (having the peaks be filter invariant) are determined.

Introduction of the Derivative Filter

Input sequence
. . . x0 x1 x2 x3 x4 x5 x6 . . .

Output sequence
. . . y0 y1 y2 y3 y4 y5 y6 . . .

The derivative filter is defined as follows:

$y_i = x_i * [(x_{i+1} - x_i) + (x_i - x_{i-1})]$

The output sequence is uniquely determined by the input sequence.

The first filter used can also be written as:

$y_i = x_i * [(x_{i+1} - x_i) + (x_i - x_{i-1})]$

$= x_i * [x_{i+1} - x_{i-1}]$ - the present value is weighted by the difference of the adjoining values

The filter in this form does not work well for 3 main reasons.
1)

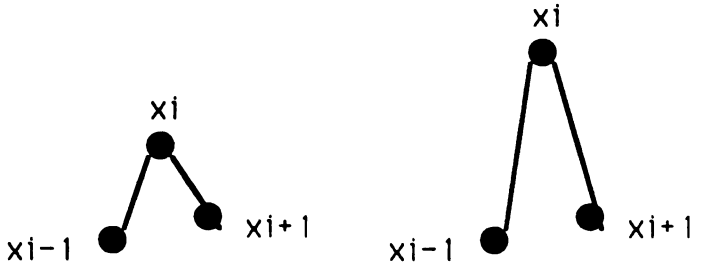


figure 10

The definition used for the discrete derivative does not take into account the middle value; the sequences shown in figure 10 have the same difference multiplier.

2)

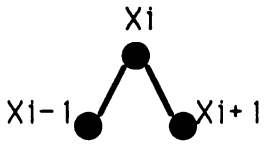


figure 11

The difference in figure 11 is zero, leading to a zero output, even though the value and the change between it and the surrounding values are not zero.

3)

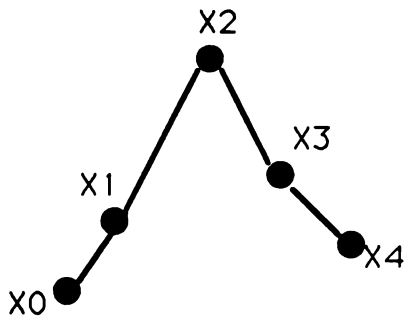


figure 12

even if x1 and x3 are both positive,as shown in figure 12, the corresponding outputs y1 and y3 will be of opposite signs.

The absolute value is introduced to help resolve these problems.

$$y_i = x_i * [|x_{i+1} - x_i| + |x_i - x_{i-1}|]$$

Here the middle value is multiplied by the absolute value of the surrounding differences or the middle value is multiplied by a factor determined by the neighboring points.

There exist five possible three point sequences. Taking only direction of change into account.

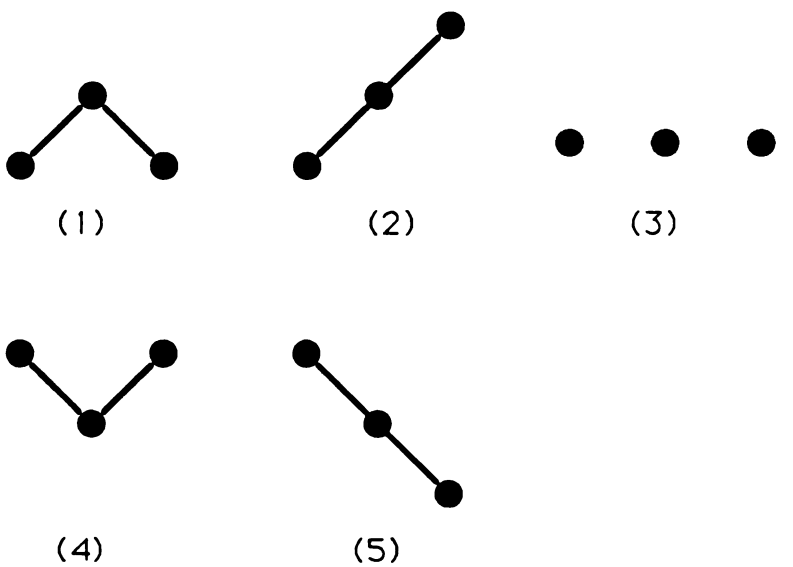


figure 13

1) if $x3 < x2$ & $x2 > x1$
$|x3-x2| + |x2-x1| = x2 - x3 + x2 - x1 = 2x2 - x3 - x1$

2) if $x3 > x2$ & $x2 > x1$
$|x3-x2| + |x2-x1| = x3 - x2 + x2 - x1 = x3 - x1$

3) if $x3 = x2$ & $x2 = x1$
$|x3-x2| + |x2-x1| = x2-x3+x1-x2 = 0$

4) if $x3 > x2$ & $x2 < x1$
$|x3-x2| + |x2-x1| = x3 - x2 + x1 - x2 = x3 + x1 - 2x2$

5) if $x3 < x2$ & $x2 < x1$
$|x3-x2| + |x2-x1| = x2-x3+x1-x2 = x1-x3$

Case 1

A relative maximum, is weighted by the absolute value of the second derivative.

Case 2

Strictly increasing.
This is the same as the original derivative formula.

$$y_i = x^*x' = x_i[x_{i+1} - x_{i-1}]$$

Case 3

All values are equal, so the derivative, and output of filter is zero.

Case 4

A relative minimum point.

$$y_i = x^*x'' = x_i[(x_{i+1} - x_i)+(x_{i-1} - x_i)] = x_i[(x_{i+1} + x_{i-1} - 2x_i)]$$

Using half step points (denoted $x_{i\pm1/2}$), the second derivative can be calculated as

$$x_i' = [(x_{i+1/2} - x_i) + (x_i - x_{i-1/2})] = x_{i+1/2} - x_{i-1/2}$$
$$x_{i+1/2}' = [(x_{i+1} - x_{i+1/2}) + (x_{i+1/2} - x_i)] = x_{i+1} - x_i$$
$$x_{i-1/2}' = [(x_i - x_{i-1/2}) + (x_{i-1/2} - x_{i-1})] = x_i - x_{i-1}$$

$$\Rightarrow \quad x_i'' = [(x'_{i+1/2} - x'_i) + (x'_i - x'_{i-1/2})]$$
$$= [(x_{i+1} - x_i) - (x_{i+1/2} - x_{i-1/2})] +$$
$$[(x_{i+1/2} - x_{i-1/2}) - (x_i - x_{i-1})]$$
$$= x_{i+1} + x_{i-1} - 2x_i$$

Neglecting the h^2 factor, this is the same as the second derivative shown in the previous section. X_i is weighted by the second derivative.

Case 5

When strictly decreasing, we have a negative derivative, this is the negative of the original formula.

$$y_i = -x^*x' = -x_i[x_{i+1} - x_{i-1}] = x_i[x_{i-1} - x_{i+1}]$$

From these five cases it is seen that when the samples are strictly increasing or strictly decreasing, the output is x_i multiplied by the absolute value of the normal discrete derivative. When it is a relative minimum or maximum, where the first derivative becomes very small, or goes to zero, the value is multiplied by a factor of the second derivative. The sample will not be multiplied by zero even when the first or the second derivative goes to zero. The only case of a zero will be when three consecutive points are equal, when both derivatives are zero.

How the Modified Derivative Filter Reshapes Signals

Two of the five possible three point sequences are the inverse of the other, so only three need to be investigated. Because three consecutive equal points result in a zero output, this case will not be investigated. To be determined is what kind of signal, in terms of relative amplitudes and relative derivatives, is needed for the heartbeat detection and estimation scheme to work. This criterion will also lead to choosing the best variation of the filter. Because the intervals between samples is constant, the relative amplitude is directly related to the derivative.

To get a intuitive feel for the filter, here are some sample of how it reshapes three common signals.

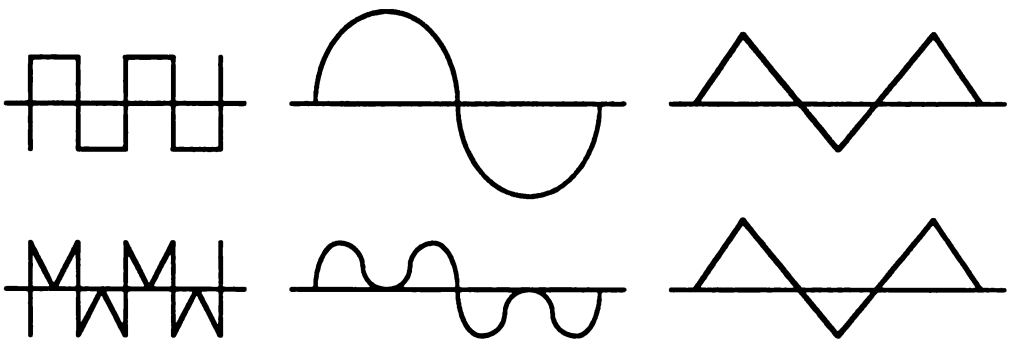
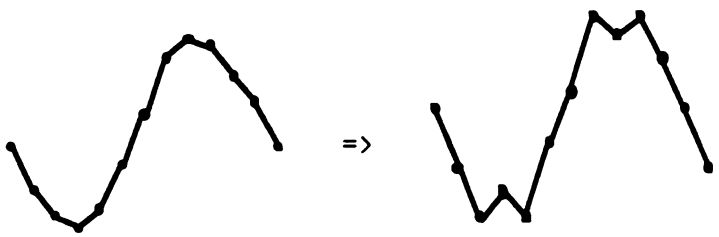


figure 14

Smooth shapes and constants will not be passed, those with constant change will not be reshaped.

Improvements and Modifications of the Filter

We want to emphasize the heartbeat signal. In our signals, the largest derivatives are where the signal is the smallest. Where the signal is large the derivative is small. This leads to output signals that look like:



before filter after filter

figure 15

This leads to a trade off. To get a good approximation of the signal, a high sampling rate must be used. Thus the points with the largest amplitude will have small derivatives. If a low sampling rate is used, the derivative will not be small, but the peak might be missed as shown in figure 16.

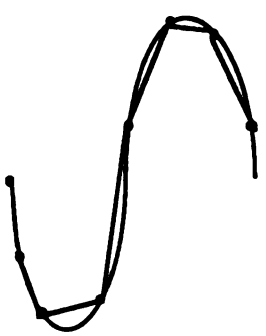


figure 16

It is possible to improve the situation in two ways. One uses a clever data reduction scheme, the other uses a new method to calculate the derivative.

The data reduction scheme could be as illustrated below:

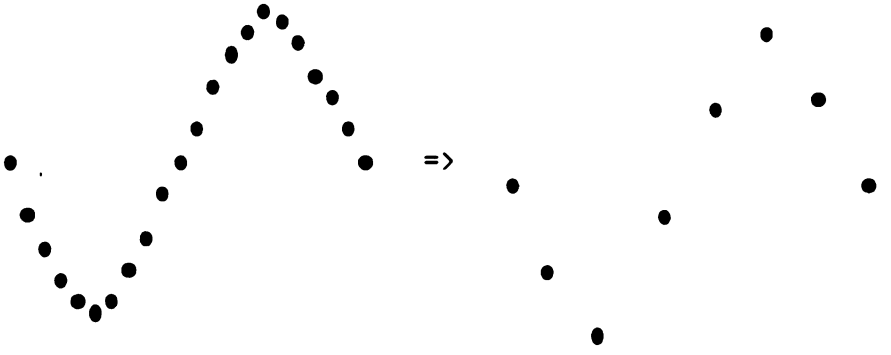


figure 17

This method attempts to approximate the signal with a piecewise linear funtion. It would help in the problem of peak reduction, but it is unattractive because the spacing between the samples may not be equal.

A new method to calculate the derivative is as follows: Use only the extreme points to calculate the derivative, then multiply all the samples between these extreme points by the same derivative.

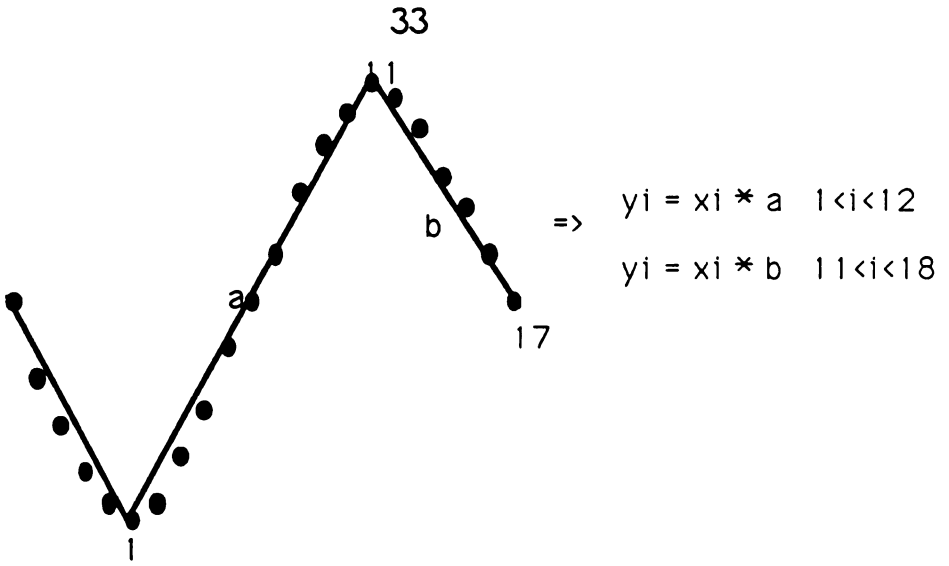


figure 18

This method retains much of the waveshape, keeps the spacing equal and eliminates the dips encountered before. Keeping the spacing of the samples uniform is necessary for later processing, so only the derivative filter using the absolute value of the differences of the 2 surrounding points, and the derivative filter using extreme points to calculate the derivative will be compared. These will be designated the first and second variation, respectively. An example of both filtering methods is shown in figure 19.

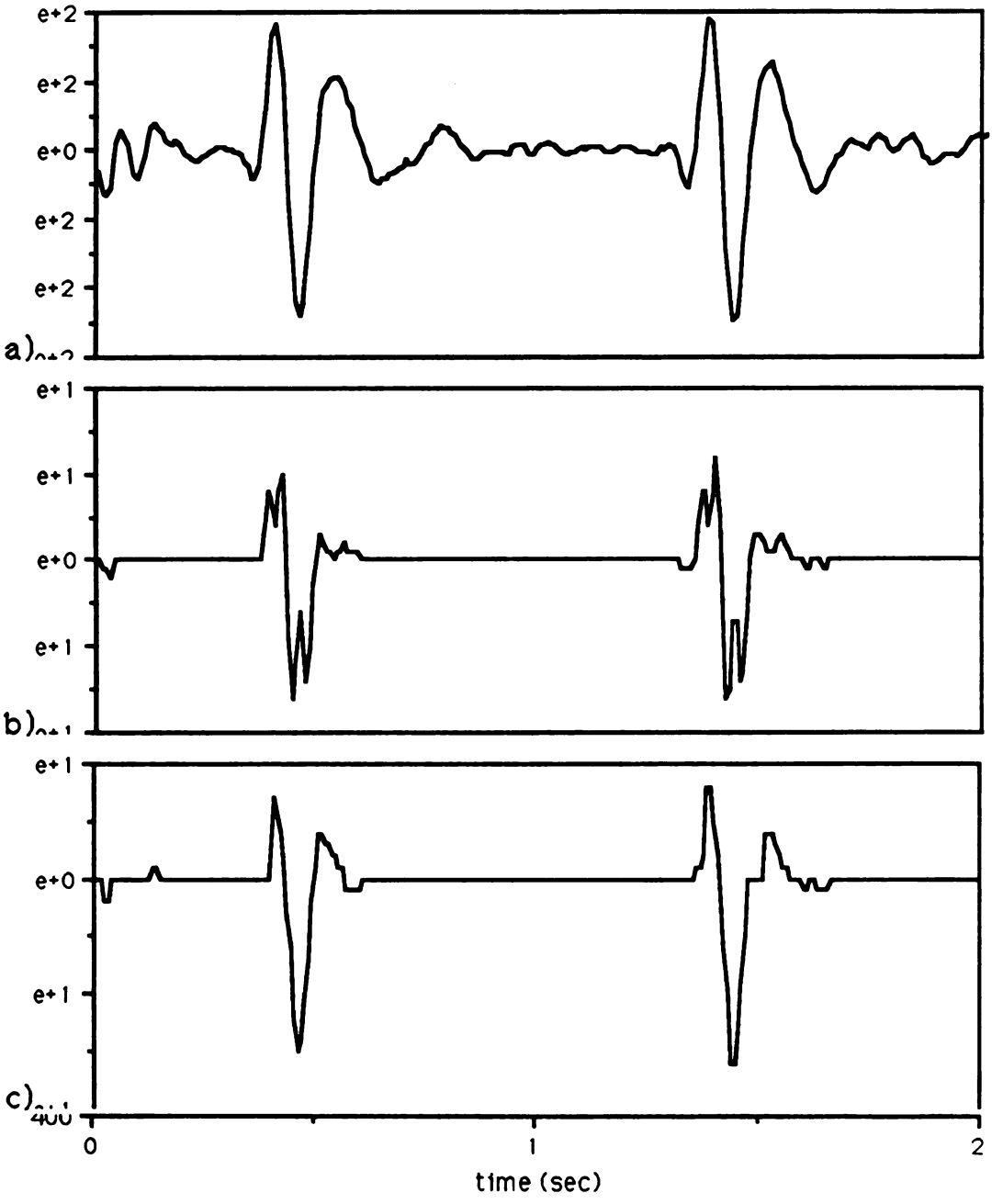


figure 19: a) Heart data, b) first variation, c) second variation filter outputs.

Comparison of two variations of the Derivative Filter

This filter is used to point to the heartbeats in the original unfiltered segment, so we would like the peaks to be filter invariant.

The first variation: using the absolute value of the surrounding points

Let x_{i-2} and x_i be fixed and let x_{i-1} move on a vertical line located at -h. This constraint applies in this system because the interval between samples is uniform.

Let x_i be bounded below and above by x_{i-2} and x_i . This section will determine the amplitude constraint on x_{i-1} so that x_i will be remain higher after the filter.

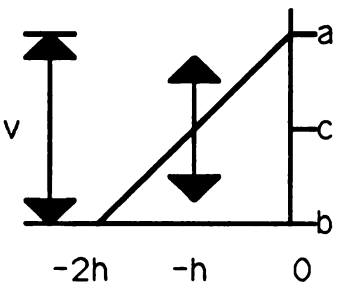


figure 20

If x_{i-2} (at -2h), x_i (at 0), and x_{i-1} (at -h),all fall on a line, a triangle is formed, and it has been shown this will pass with only a scaling factor. If x_{i-1} does not fall on the line formed by x_{i-2} and x_i then

$|x_i| = a$ $\qquad$ $a-b = v$

$|x_{i-1}| = c$ $\qquad$ $y = \frac{v}{2h} t + a$

$|x_{i-2}| = b$

at -h $\quad y = \frac{v}{2h}(-h) + a$

$$= \frac{a-b}{2h}(-h) + a = \frac{b-a}{2} + \frac{2a}{2} = \frac{a+b}{2}$$

$$c < \frac{a+b}{2}$$

For the first variation:

$y_i = x_i * [|x_{i+1} - x_i| + |x_i - x_{i-1}|]$ - to find the positive peaks, find the zeros of the first derivative and of the second derivative.

negatives

$$y_i' = \frac{(y_{i+1} - y_i) + (y_i - y_{i-1})}{2h}$$

because this derivative never, or almost never, goes to zero, will define critical points as points where $(y_{i+1} - y_i)$ and $(y_i - y_{i-1})$ have opposite signs.

$$y_i'' = \frac{(y_{i+1} + y_{i-1} - 2y_i)}{h^2}$$

when y_i'' is negative there is a local maximum and when y_i'' is positive there is a local minimum. We would like these local maximums and minimums to be the same samples as before, only the relative heights of the samples changed.

$$x_i' = \frac{(x_{i+1} - x_i) + (x_i - x_{i-1})}{2h} \qquad x_i'' = \frac{(x_{i+1} + x_{i-1} - 2x_i)}{h^2}$$

$$y_i' = \frac{(y_{i+1} - y_i) + (y_i - y_{i-1})}{2h} \qquad y_i'' = \frac{(y_{i+1} + y_{i-1} - 2y_i)}{h^2}$$

considering only positive peak points,

$$y_i' = \backslash f([x_{i+1} * (|x_{i+2} - x_{i+1}| + |x_{i+1} - x_{il})] - [x_{i-1} * (|x_i - x_{i-1}| + |x_{i-1} - x_{i-2}|)], 2h)$$

case 1
$x_i > x_{i-1} > x_{i-2}$
$x_i > x_{i+1} > x_{i+2}$

this extreme point in the x_i series will remain the extreme point in the y_i series, if a and b are of different signs.

$$y_{i+1} - y_i = [x_{i+1}(|x_{i+2} - x_{i+1}| + |x_{i+1} - x_i|)] - [x_i(|x_{i+1} - x_i| + |x_i - x_{i-1}|)]$$

$$= [x_{i+1}((x_{i+1} - x_{i+2}) + (x_i - x_{i+1}))] - [x_i((x_i - x_{i+1}) + (x_i - x_{i-1}))]$$

$$= x_{i+1}(x_i - x_{i+2}) - x_i(2x_i - x_{i+1} - x_{i-1})$$

$$= 2x_{i+1}x_i - 2x_ix_i - x_{i+1}x_{i+2} + x_ix_{i-1}$$

$$y_i - y_{i-1} = [x_i(|x_{i+1} - x_i| + |x_i - x_{i-1}|)] - [x_{i-1}(|x_i - x_{i-1}| + |x_{i-1} - x_{i-2}|)]$$

$$= [x_i((x_i - x_{i+1}) + (x_i - x_{i-1}))] - [x_{i-1}((x_i - x_{i-1}) + (x_{i-1} - x_{i-2}))]$$

$$= x_i(2x_i - x_{i+1} - x_{i-1}) - x_{i-1}(x_i - x_{i-2})$$

$$= 2x_ix_i - 2x_ix_{i-1} - x_{i+1}x_{i+1} + x_{i-1}x_{i-2}$$

to get:
$(y_{i+1} - y_i) < 0$
$(y_i - y_{i-1}) > 0$

will need:
$2x_{i+1}x_i - 2x_ix_i - x_{i+1}x_{i+2} + x_ix_{i-1} < 0$
$2x_ix_i - 2x_ix_{i-1} - x_ix_{i+1} + x_{i-1}x_{i-2} > 0$

$$2x_{i+1}x_i - 2x_ix_i - x_{i+1}x_{i+2} + x_ix_{i-1} < 2x_ix_i - 2x_ix_{i-1} - x_{i+1}x_{i+1} + x_{i-1}x_{i-2}$$

$$3x_{i+1}x_i + 3x_ix_{i-1} - x_{i+1}x_{i+2} - x_{i-1}x_{i-2} < 4x_ix_i \qquad (5)$$

assumed: $x_i > x_{i+1} > x_{i+2}$
$x_i > x_{i-1} > x_{i-2}$
introducing two more constraints,
$x_{i+1} = x_{i-1}$
$x_{i+2} = x_{i-2}$

(5) becomes,

$$6x_{i+1}x_i - 2x_{i+1}x_{i+2} < 4x_i^2$$
$$3x_{i+1}x_i - x_{i+1}x_{i+2} < 2x_i^2$$

$3ca - cb - 2a^2 < 0$ or $3x_{i-1}x_i - x_{i-1}x_{i-2} < 2x_i^2$

$c(3a - b) - 2a^2 < 0$ $x_{i-1}(3x_i - x_{i-2}) < 2x_i^2$

$c < \dfrac{2a^2}{3a - b}$ $x_{i-1} < \dfrac{2x_i^2}{3x_i - x_{i-2}}$

So x_{i-1} must be less than $\dfrac{2x_i^2}{3x_i - x_{i-2}}$ for the peak to remain in the same place. Of course, the same can be done for x_{i+1} the first sample after the peak. On the negative peaks the inequality is reversed.

It becomes apparent from this equation how nonlinear the process is. Suppose $x_i = 10$ and $x_{i-2} = 5$. x_{i-1} must be less than 8. If $x_i = 5$ and $x_{i-2} = 0$, the same relative difference, x_{i-1} must be less than 3.33. Thus, the same difference is changed in amplitude, the relative difference of x_i and x_{i-1} does not stay the same.

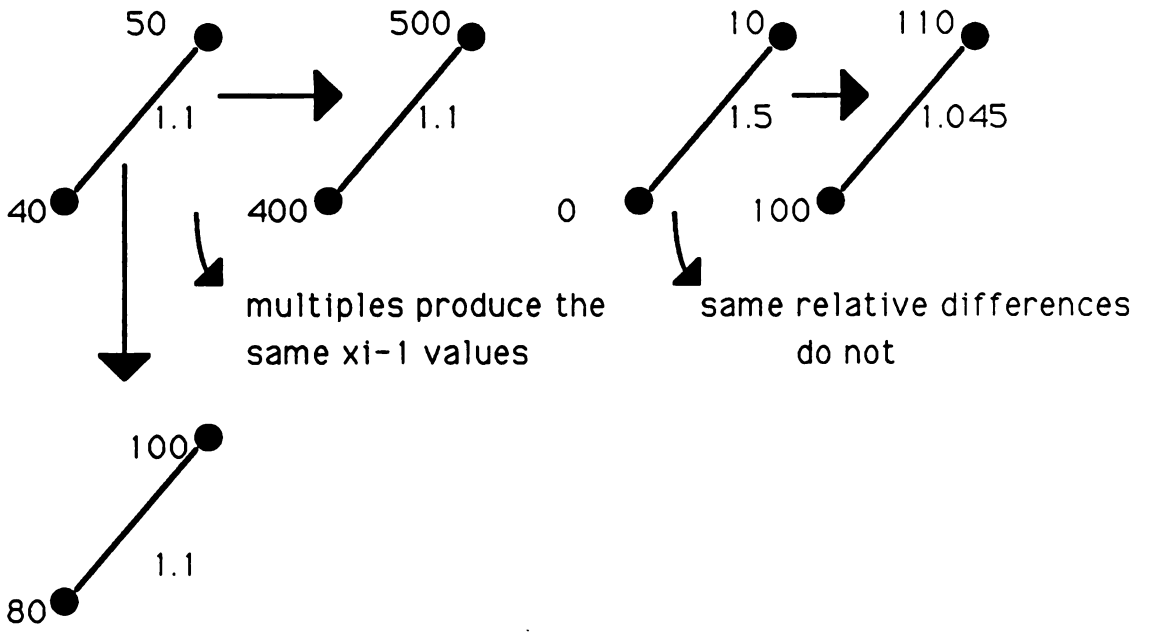


figure 21

In the leftmost figure above, if $x_{i-2} = 40$ and $x_i = 50$, x_{i-1} must be less than 1.1 of x_i. Any multiple of these two numbers will have the same requirement for x_{i-1}. The same relative differences, such as 0 to 10 and 100 to 110, do not have the same requirements for x_{i-1}. In general, x_{i-1} must be about on the line formed by x_i and x_{i-2} or below to get good results. This is very important. With a sampling rate to assure the peak is recorded, this will not be the case in general.Looking at 41 segments of typical heart data both filters were used and the results are as follows. Using only three points to determine the derivative, 78 heartbeat peaks did not change position, and 41 did. Using the extreme points to calculate the derivative, 93 heartbeat peaks did not change position and 26 did.

The second variation of the filter: using extreme points to calculate derivative.

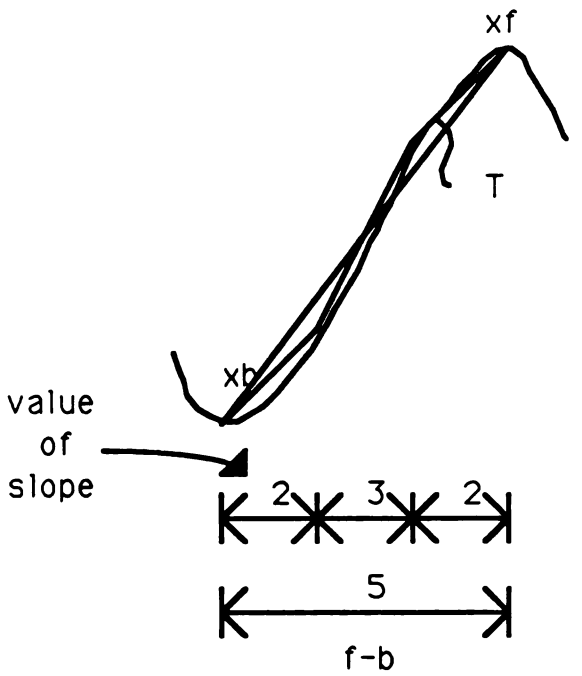


figure 22

$$\Delta_T = \frac{x_f - x_b}{f - b}$$ the slope from x_b to x_f

All horizontal distances between the points are the same. Between two extreme points, all the slopes are either all positive or all negative, and ΔT is just the average of all the slopes between these extreme points.

$$\frac{x_i - x_{i-1}}{h} = \frac{\Delta x_i}{h}$$

$$\frac{\Delta x_f + \Delta x_{f-1} + \ldots + \Delta x_{b+1}}{f - b} = \frac{x_f - x_b}{f - b}$$

The first variation uses $|\Delta x_{i+1}| + |\Delta x_i|$ for a multiplier, which for high sampling rates and sinusoidal signals, is smaller than ΔT giving the second filter better performance. Also,because x_f is higher by definition than the

other samples, and all the points between x_b and x_f are multiplied by the same derivative value, x_f will be the highest point after the multiplication.

The Δx_f and Δx_{b+1} for the sinusoidal signal are smaller than the other differences and using $|\Delta x_{i+1}| + |\Delta x_i|$ for the multiplier can make some inner points larger than the extreme points. This is shown in figure 23.

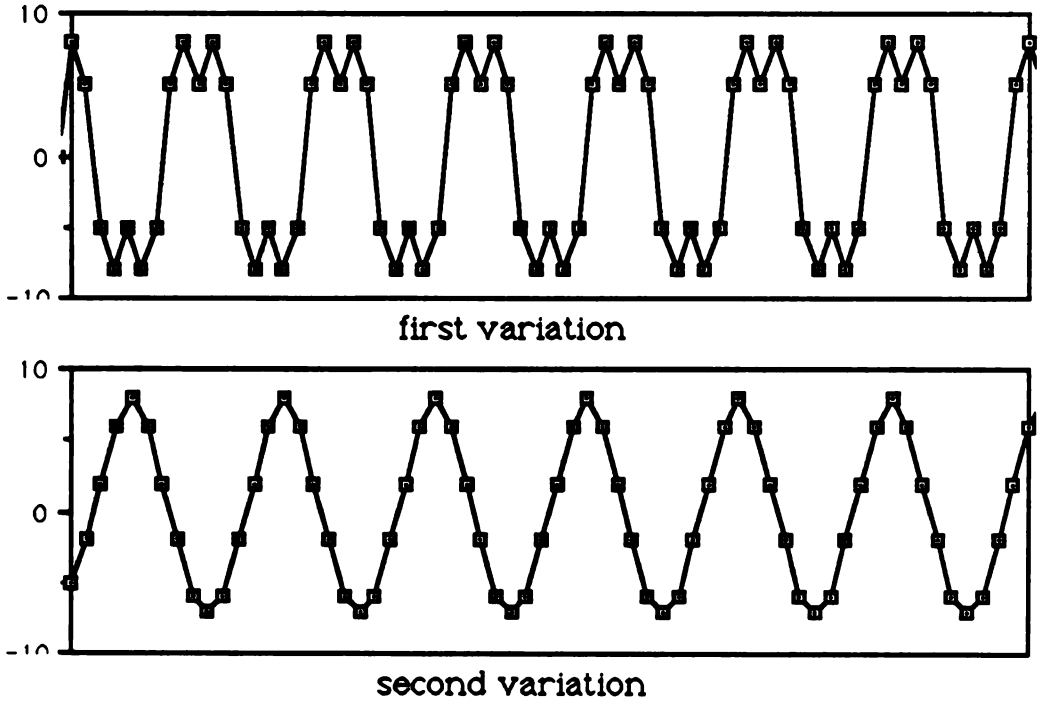


figure 23:output of derivative filter for 6.4 Hz sine wave input

Our main purpose is to increase high derivative high amplitude signals over low derivative high amplitude or high derivative low amplitude signals. Using the first variation we would need to sample at least as low as shown below to get as good results as the second variation.

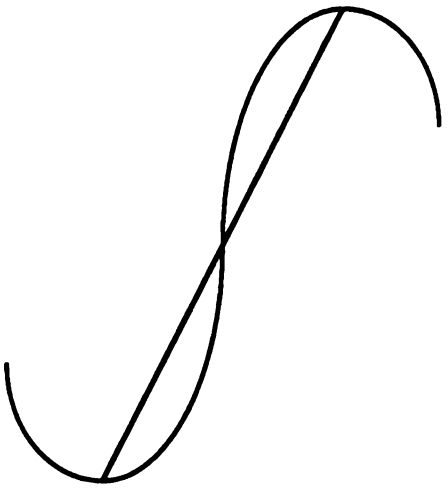


figure 24

Because we have a band-pass pre-filter, we can set it at the highest heart frequency component so no clutter is present of higher frequency, or has a higher derivative than the heart. There is a trade off in the filter: use a high sampling rate to assure getting the highest point and then have smaller derivatives; or use a smaller sampling rate to keep the large changes between samples and maybe miss the highest point. For square waves or triangle waves, both variations have the same output, so the higher sampling rate that can be used with the second variation only assures the peak is obtained. In the heart signal measurement,we deal with more sinusoidal shapes. In these waveforms the highest slope is where the signal is smallest; we would like the slope large for large amplitudes. To make the signal large, the filter is non-linear. In any sinusoidal signal, with a high enough sampling rate,

$$x_f |\Delta_T| > x_f \ [|\Delta x_{f+1}| + |\Delta x_f|]$$

as was shown in figure 23. This effect, using the first filter, becomes intolerable at high sampling rates. As seen before, there can be double peaks produced. The second filter will eliminate double peaks because, by

definition, the extreme point is higher than the others. In the detection and estimation algorithm the pre-filter is used to determine places where a heartbeat is likely to be. So double peaks are tolerable if this algorithm is better at distinguishing heartbeats from clutter. As shown, however, this is not the case and the second variation of the filter performs best.

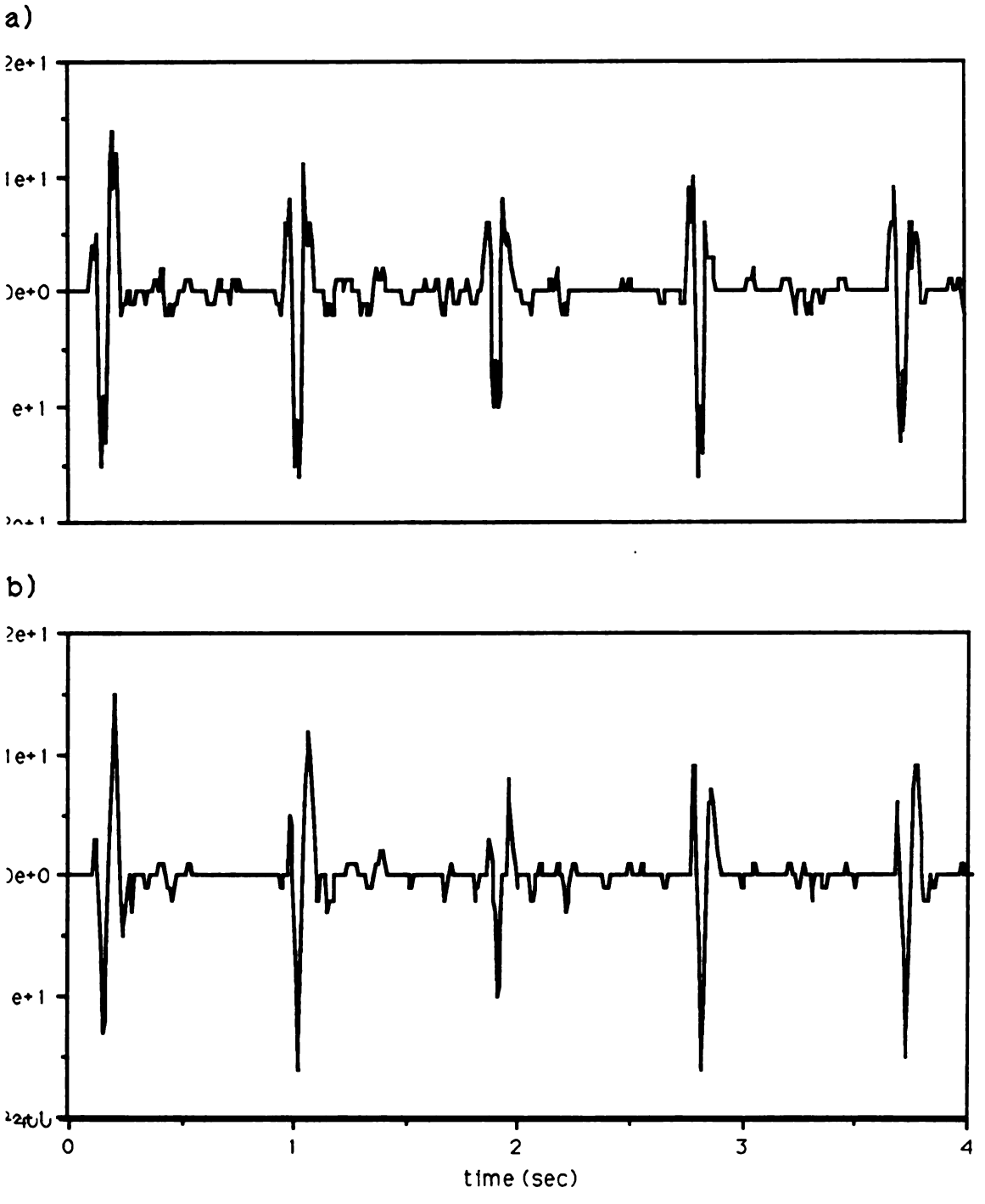


figure 25: Sample outputs of a) first and b) second variations of the derivative filter

Sampling frequency

We need to determine an adequate sampling frequency for the derivative filter. We have an 8 bit, or 256 level analog to digital converter. We use at least fifty percent, 128 levels, of these levels for the heartbeat peaks. Our analog filter cuts off at 15 Hz.

Using a 15Hz sine wave for our analysis, we will determine the sampling rate needed to discretize a sample within a pre-specified percentage of the analog peak. From our algorithm, we want to locate the heartbeat peaks. This means no matter what the phase of the signal is with respect to the sampling (0-2π) at t=0, we want to be assured of at least having one sample within 90% of the highest point in the signal.

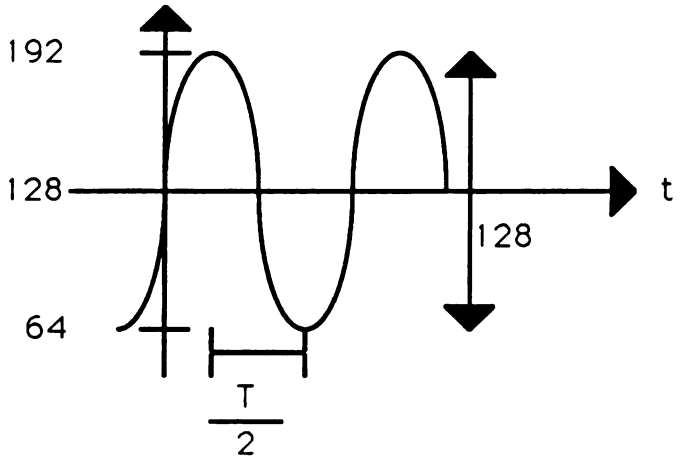


figure 26

for 15Hz signal and a peak to peak sinusoidal signal of 128,

$$\text{ø changes } 180^o / \frac{1}{30}\text{sec} \quad => \quad 5400\frac{^o}{sec}$$

$$0.9(192) = 64\cos(ø) + 128 \quad => \quad 45.57^o$$

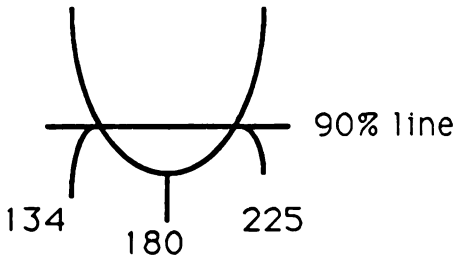


figure 27

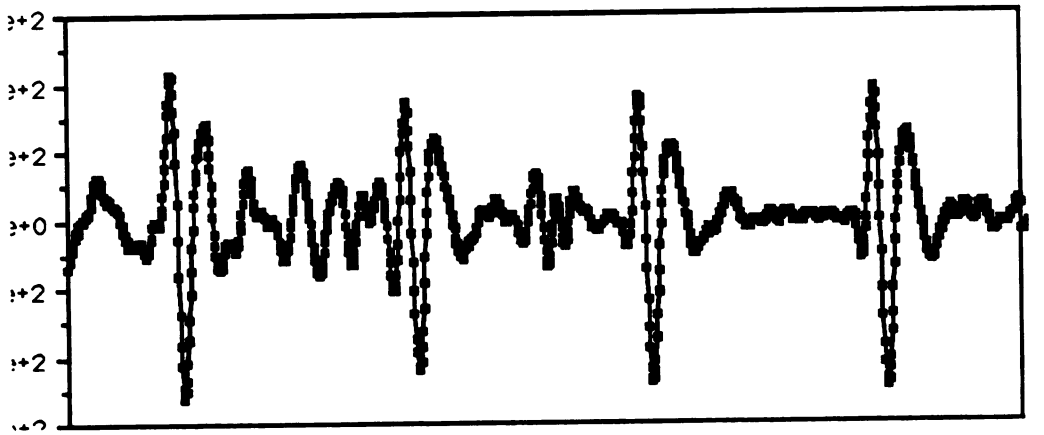
for Nyquist criterion:

SR(sampling rate) > 1/30 sec => 30 Hz SR

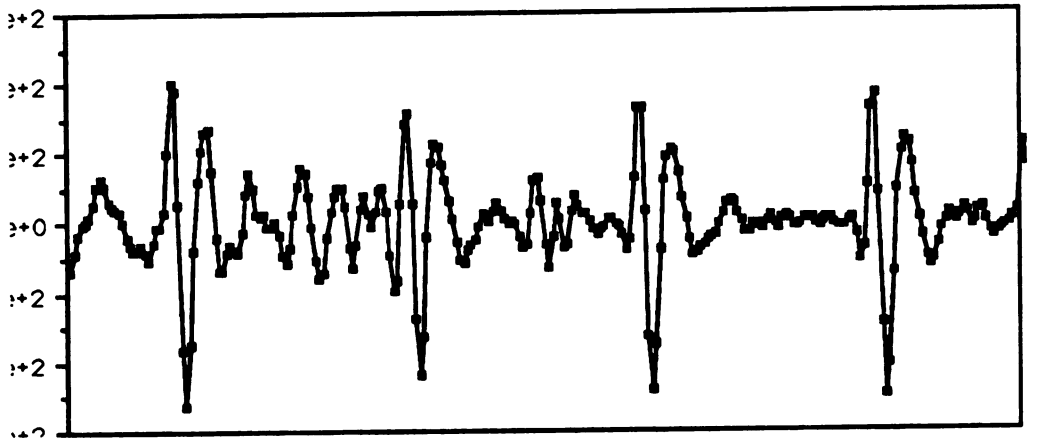
for 90% criterion:

$$\frac{45.57^o * 2}{5400\frac{^o}{sec}} = \text{sec} \quad => \quad 59.24 \text{ Hz SR}$$

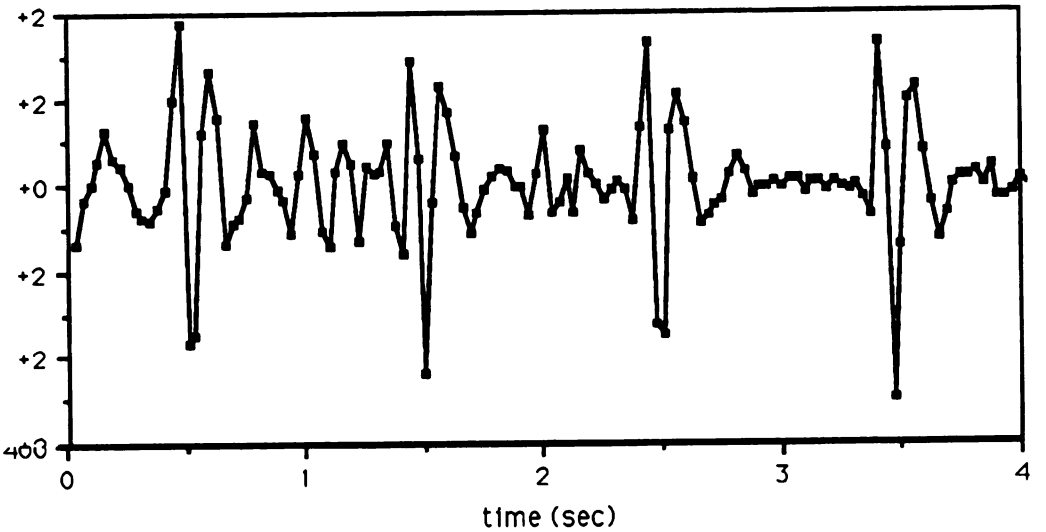
If we use all 256 levels, the sampling rate for 90% criterion is 73.23 Hz. The 64 Hz sampling rate used is adequate, as shown.



a) 256 Hz sampling rate



b) 64 Hz sampling rate



c) 32 Hz sampling rate

figure 28: different sampling rates

Determination of normalizing factor

To make the filtered data easier to process integer normalization is done. This division factor, s, is chosen so that some inter-heartbeat clutter is eliminated. Having some of the data points equal to zero, will make the detection and estimation easier, only having to look at a finite group of non-zero areas in each segment of data and determine if these are heartbeats or noise. Thus we want as many points as possible to be zero after filtering, without eliminating any heartbeat peaks. The trade off will be a greater number of zeros vs. reduction in precision. A method for determining this normalization and the normalization factor must be determined.

First, a method for calculating the normalized output must be obtained.

In the presentation that follows, s is the normalizing factor. The x's and arithmetic is all integer, which truncates all fractions.

1) The first method used is as follows.

To keep as much precision as possible, x_i is rounded off to the nearest s.

$$\frac{x_i + \mathrm{sgn}(x_i) * \frac{s}{2}}{s}$$

where $\mathrm{sgn}(x) = 1$ if $x_i \geq 0$

-1 if $x_i < 0$

With this method, all $x_i \leq |s/2| \Rightarrow$ go to zero. In other words, a factor s is chosen so that all samples less than half s will be zero and precision will be reduced by this factor also.

All x_i falling within $\pm$ s/2 of some (ks) will be rounded off to the same value.

.

$x_i = sk$ for $sk - s/2 < x_i \leq sk + s/2 \qquad k = 0,1,2,3 \ldots$

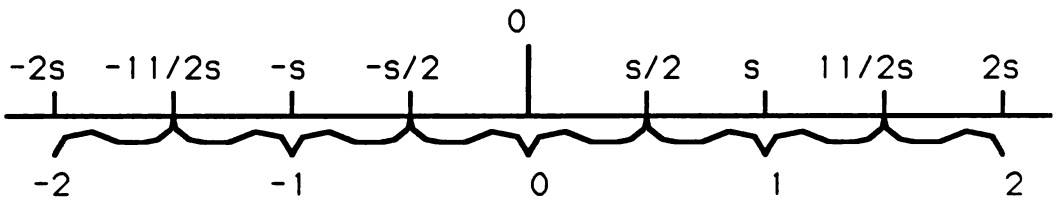


figure 29 : quantization intervals

2) The second method is simply integer division by s.

$$\frac{x_i}{s}$$

There will be reduction in resolution or precision, but if the s here is the same as the s in the first quantization method, we will have the same quantization intervals except around zero, where it will be twice as long. We need to decide how much loss of resolution can be tolerated.

All x_i falling within $\pm s/2$ of some (ks) will have the same value.

$x_i = sk$ for $sk < x_i \leq sk + s \qquad k = 0,1,2,3 \ldots$

Negative values will be the mirror image.

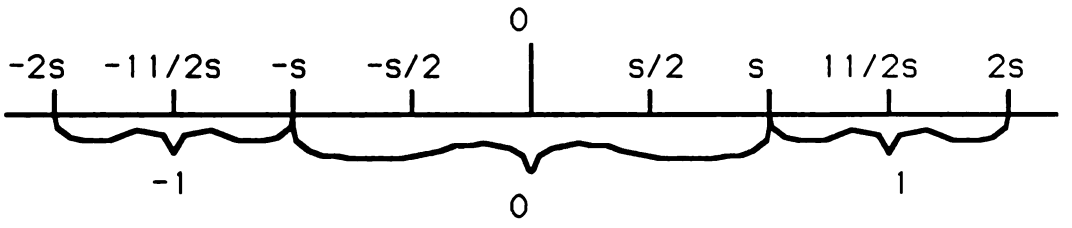


figure 30 : quantization intervals

The trade off is precision vs. eliminating clutter. Examining the precision of each one:

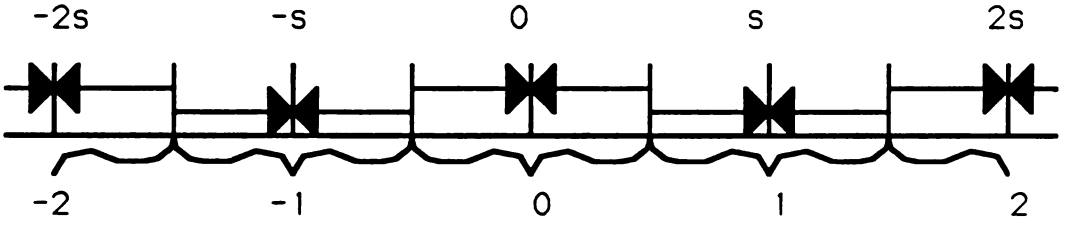


figure 31 : normalization values

the first method has a maximum quantizing error of .5s

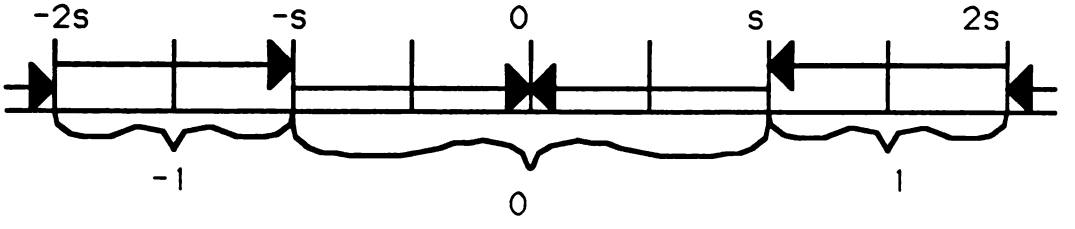


figure 32 : normalization values

the second method has a maximum quantizing error of s

The first method has a quantizing error of ±.5s and the second an error of s. The second method however, makes everything from -s to s go to zero, and the first method makes everything from -s/2 to s/2 go to zero. If the second method is used with s/2 as the normalizing factor, it will have a quantizing error of s/2 and will make everything from -s/2 to s/2 go to zero.

To get the mean square quantizing error of each [20] (MSQE)

$$MSQE = \sum_{i=1}^{L} (MSQE_i) P_i$$

L is # of levels

Pi is probability of sample in i^{th} level

$$MSQE_i = \int_{I_i} (x - \beta_i)^2 \, p(x|i) \, dx$$

$$p(x|i) = \frac{P_d(x)}{P_i}$$

$P_d(x)$ - probability density of voltage β_i quantizing value

$$MSQE = \sum_{i=1}^{L} \int_{I_i} (x - \beta_i)^2 \, P_d(x) \, dx$$

Integral evaluation for the first case:
Using 16 as the normilizing factor, the interval around zero will be from -8 to 8. All the intervals have the same $MSQE_i$.

$$MSQE_i = \int_{-8}^{8} (x - 0)^2 \, dx = \frac{x^3}{3} = \frac{1024}{3} = 341.33$$

Integral evaluation for the second case:
Using 16 as normalizing factor, the interval around zero is from -16 to 16, length 32. The other intervals will be of lentgth 16. The $MSQE_i$ shown here is the average of all the $MSQE_i$'s.

$$MSQE_i = \frac{1023}{1025}\int_{16}^{32}(x - 16)^2\,dx + \frac{2}{1025}\int_{-16}^{16}(x)^2\,dx$$

$$= \frac{1023}{1025}\left[\frac{x^3}{3} - 2x^2 + 16x\right]_{16}^{32} + \frac{2}{1025}\left[\frac{x^3}{3}\right]_{16}^{-16}$$

$$= 1367.997$$

Using 8 as the normilizing factor, to get same zero region as the first case,

$$MSQE_i = \frac{2046}{2050}\int_{-8}^{16}(x - 8)^2\,dx + \frac{2}{2050}\int_{-8}^{8}(x)^2\,dx$$

$$= 170.667$$

Thus the second method has a smaller MSQE for the same zero area. In figure 33, the signal was normalized to ±8, using $x = \frac{x+\mathrm{sgn}(x)*(\frac{s}{2})}{s}$ on top, and $x = \frac{x}{s}$ on the bottom. The second normalization method, $x = \frac{x}{s}$ has been chosen for normalization.

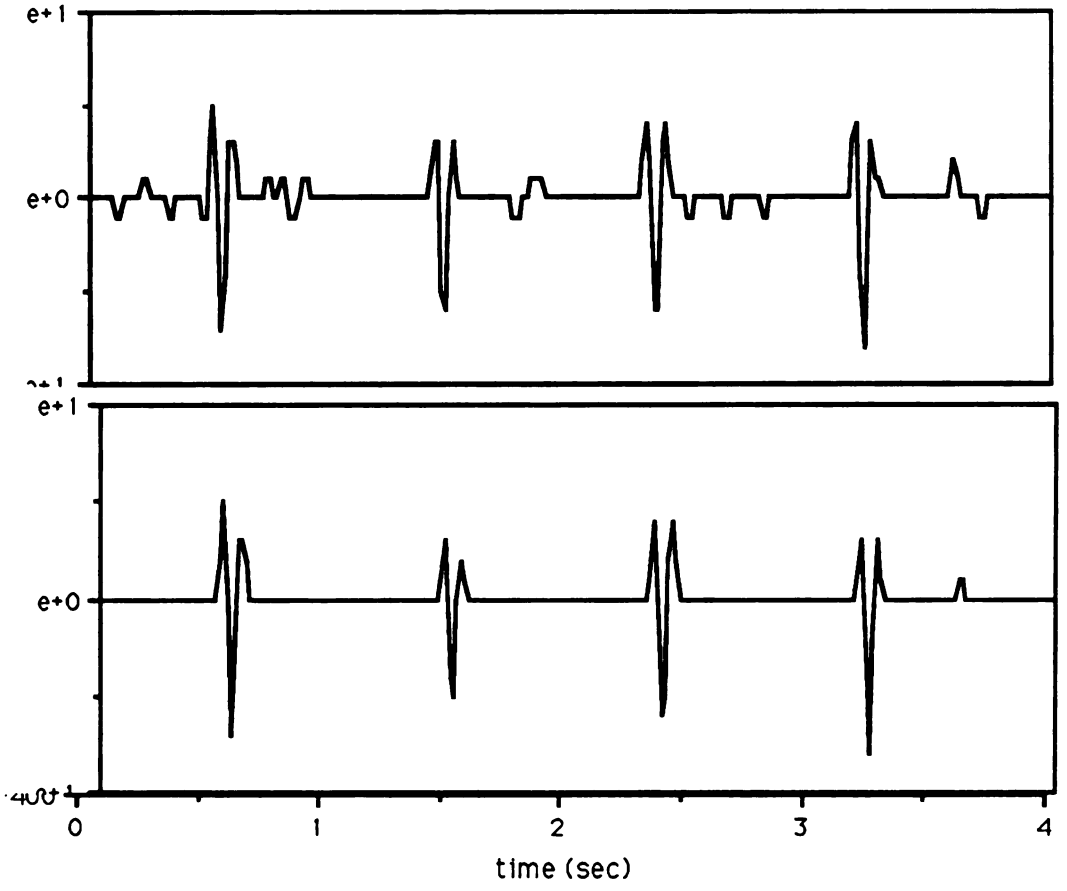


figure 33: Different normalization procedures.

Determining the normalization factor, s in the equations above, is a matter of trial and error. The method used was to use many heartbeat segments and determine the largest factor that does not eliminate any heartbeat. Figure 34 shows the output of the filter for different normalization factors. This is a very good signal, so a small normalization factor, such as 4 works very well. In general there is a large difference between heartbeat amplitudes and a larger factor is needed so none of the heartbeats is normalized to zero. The normalization factor chosen was 8.

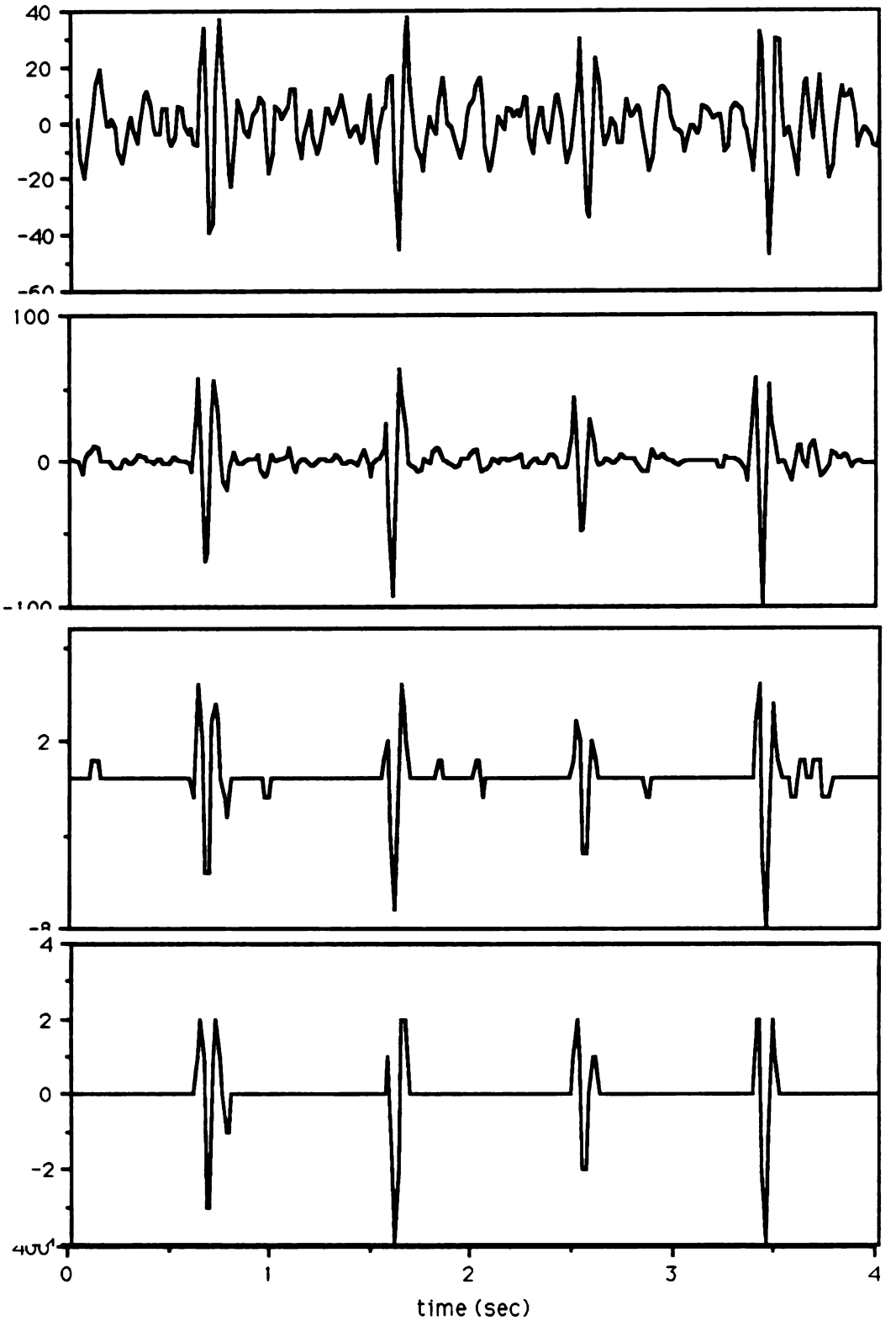


figure 34 : Original signal filtered and normalized using s of 100,8,and 4.

Determination of average number of zeros surrounding each Nonzero Section

First the signal is filtered. Then the signal is normalized using the integer division or normalization of the previous section. The normalization will produce non-zero groups surrounded by zeros. We need to determine how many zeros in a row will separate non-zero regions.What is desired is to choose a number of consecutive zeros so that a single heartbeart would not be split and consecutive heartbeats would not be joined. One or two zeros is too few, as it was found that one or two consecutive zeros can occur in the waveform of the same heartbeat. On the other hand there tend to be non-zero regions close to the heartbeat waveform and ten or twenty zeros is too many. How many zeros in a row should distinguish two non-zero groups, each possibly a heartbeat, needed to be determined. A statistical analysis of heartbeat data was used.

Using 84 segments with heartbeats and 33 with only noise, the average number of consecutive zeros for heartbeat segments was 6.7 and the average number of consecutive zeros for noise only was 4.9. Goodness of fit analysis was performed to determine an adequate model. However, this did not lead to any useful results. Detection is not performed at this level so the noise did not come into consideration. It was found that for a signal with a reasonable signal level,where the heartbeat could be detected, the number of consecutive zeros in a heartbeat did not exceed four and the number of consecutive zeros between heartbeats was larger than this. Thus, five consecutive zeros was chosen as the threshold for two non-zero segments to be considered distinct.

Detection and Estimation Algorithm

A good algorithm for determining if two waveshapes at different times are close enough in some sense to be called consecutive heartbeats is needed. When a observer looks at a collection of data, typically four seconds of data, he is able to determine if a heartbeat is present, and if it is, where each heartbeat is located, even when the signal to noise ratio is too low for computer algorithms such as peak detection or correlation to work. The observer uses pattern recognition for his detection and estimation system. Without elaborate programming it would be very difficult for a computer to take four seconds of data and do a similar procedure as the human. This is because information easily interpreted from a picture is not available to the computer. Information such as knowing if the heartbeats are there, where they are, what is their period, what is their shape, and their amplitude. It is also probable that the period, the shape and amplitude will change between heartbeats. An observer looking at a graph can easily adjust to these changes, but it is much harder for a computer working only with numbers. Following this reasoning, the following algorithm was developed. Unless a good model for the heart signal is determined, selecting an optimum algorithm cannot be done. The frame of reference for judging algorithms, is how well it performs compared to human recognition.

One basic premise is that if a human can look at a four second segment of data and pick out the heartbeats, if present, an algorithm which can do the same must exist. We may not be able to determine if a given algorithm is optimum, but since we are using human recognition as the criterion,if it can work as well or better than human recognition, it is an acceptable algorithm.

We want to have a process that can detect the repetition of similar shapes in the segment. We cannot use typical correlation of the data because the heartbeats are not periodic. We also cannot use a matched filter because the shape of the heartbeat is not known and is continually changing. Because it is a small region that is repeating, if we could take this region and correlate it with the whole segment, as shown in figure 35, we would get rid of the problem associated with non-periodicity.

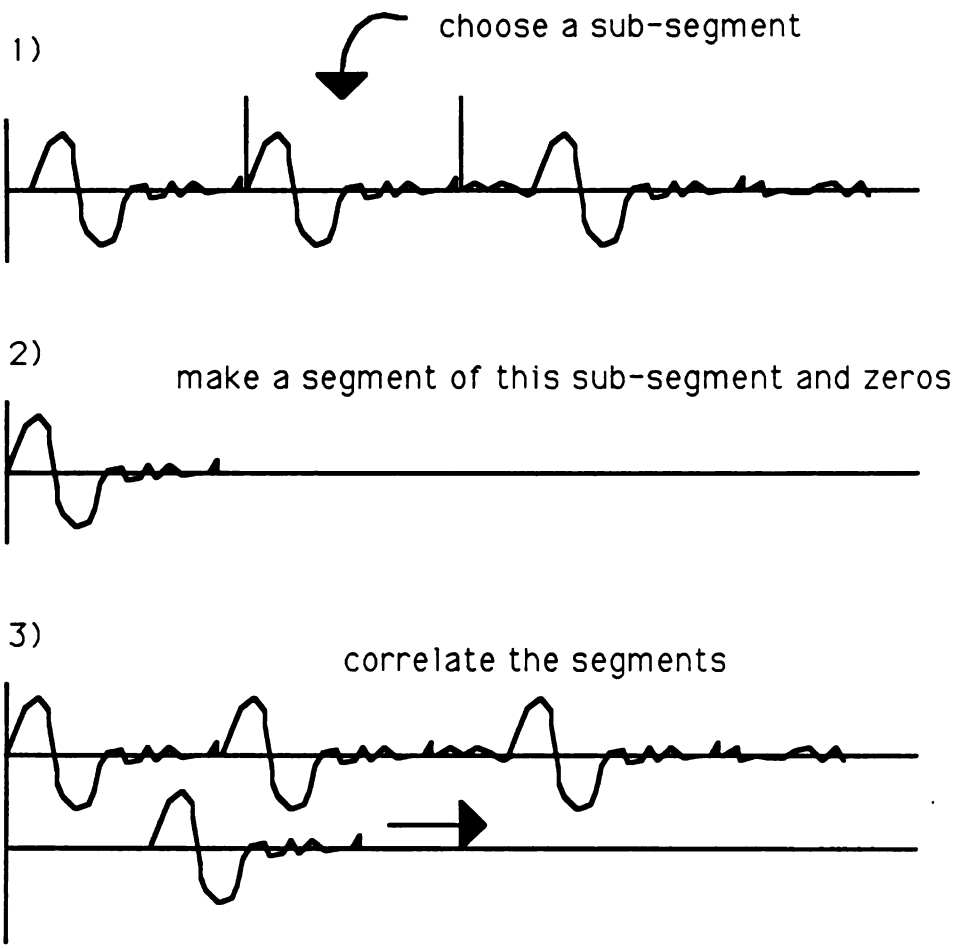


figure 35

The problem is finding a subregion to correlate with the whole segment. Sampling at 64 Hz we have 256 samples in four seconds. A typical heartbeat is 8 samples. This means there are 248 regions that can be chosen as a heartbeat. It is too time consuming to use each of the 248 sub segments and correlate with the whole segment and see which has good correlation at regular intervals in the segment. This is where the derivative filter plays a role.The normalization factors of the filters can be set so few heartbeats will be set to zero by normalization, and will produce zero regions. However, this will not set all the noise regions to zero. So the filtering and normalization by itself is not a good criterion for deciding if a heartbeat is present. However, it will point to non-zero regions that could be chosen for the correlation. Instead of having the algorithm look at the entire data segment to decide if there are repeating pseudo periodic regions that are similar in shape, it will only have to compare a smaller number of intervals produced by the non-linear filter. Thus a combined system using the derivative filter to segment the data into possible heartbeats, and a pattern recognition algorithm to get rid of false detections will be a good detection and estimation scheme.

The output array from the filter contains zero elements separating non-zero regions as shown in figure 36.

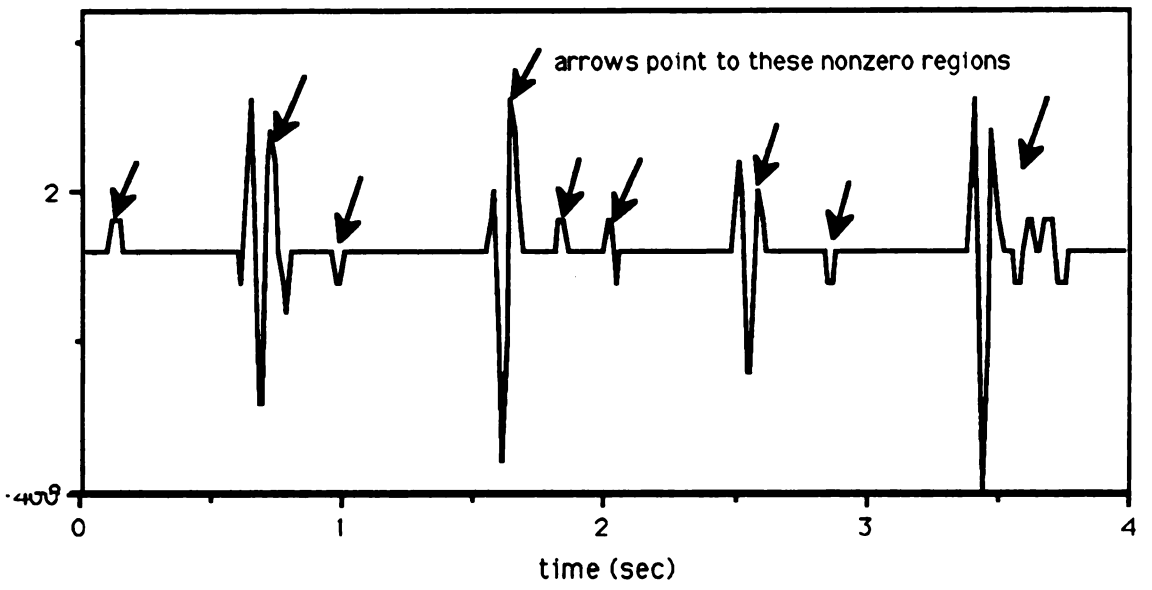


figure 36 : non-zero segments after filtering

In this algorithm there must be five consecutive zeros to consider regions distinct and different. For segments containing only heartbeat pulses, typically the number of non-zero regions in a 4 second segment varies from three or four,to fifteen or twenty. For noise signals, the number of non-zero regions typically varies from one to more than twenty. Sample outputs are shown in figure 37.

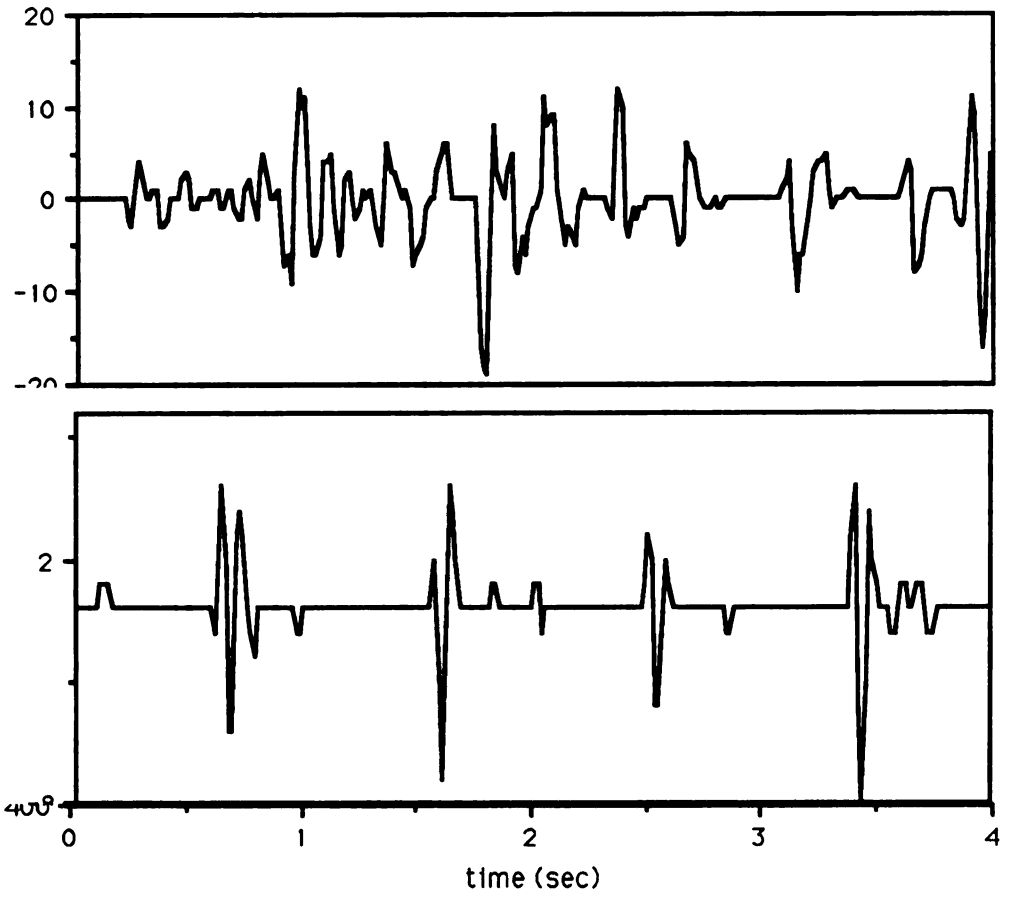


figure 37 : examples of derivative filter output

The objective is determine if these non-zero regions are close enough in shape, and are pseudo periodic in order to consider them heartbeats. One way would be to use each sub-region to make a new segment as shown in figure 35. The correlation of these regions would give an indication of the period between regions and their similarity in shape. This procedure works well, but it is also too time consuming because a new segment then correlation is performed for each sub-region. The procedure chosen selects one of the regions according to amplitude and derivative as being the most likely to be a heartbeat and uses this region for correlation. Because the derivative filter re-shapes the signal, and because the heartbeat waveform

usually extends beyond the non-zero regions determined by the filter, the correlation is done using the original unfiltered signal.

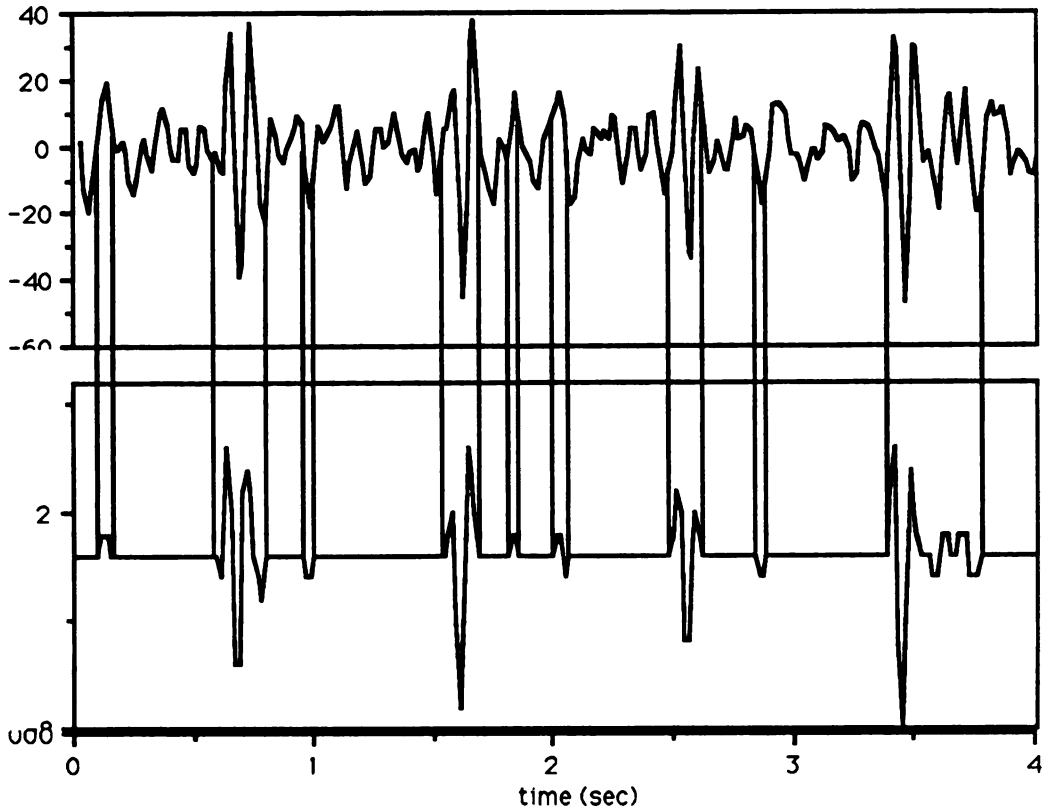


figure 38:derivative filter output pointing to possible heartbeats

A 64 Hz sampling rate is high enough to capture the peaks and high enough for the derivative filter, but is inadequate for correlation. Typically there are three of four samples per peak and this does not give enough discrimination between shapes. When doing the shape discrimination visually using a graph, the points are interpolated and 64 Hz is an adequate sampling rate. To get finer definition in the correlation, a higher sampling rate, 512 Hz, is used. Because because a 64 Hz sampling rate is high enough for good peak definition, when doing the correlation on 4 seconds of data using the 512 Hz array (2048 samples), the region is slid over eight samples instead of

one after each step in the correlation.There are then 256 steps in the partial correlation.2048/8=256.

An example of a 256 element output correlation file is shown in figure 39.

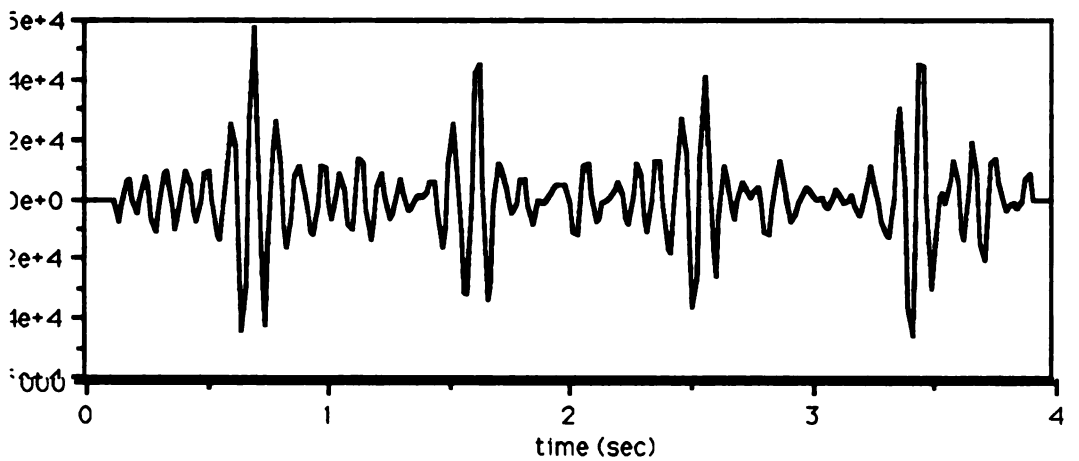


figure 39 : correlation output

Using the region boundaries found from the derivative filter, the highest peaks are found in the corresponding regions of the correlation file. If there is no value greater than zero in the region, no value is taken for this region. The final output file is all zero except at these highest points.

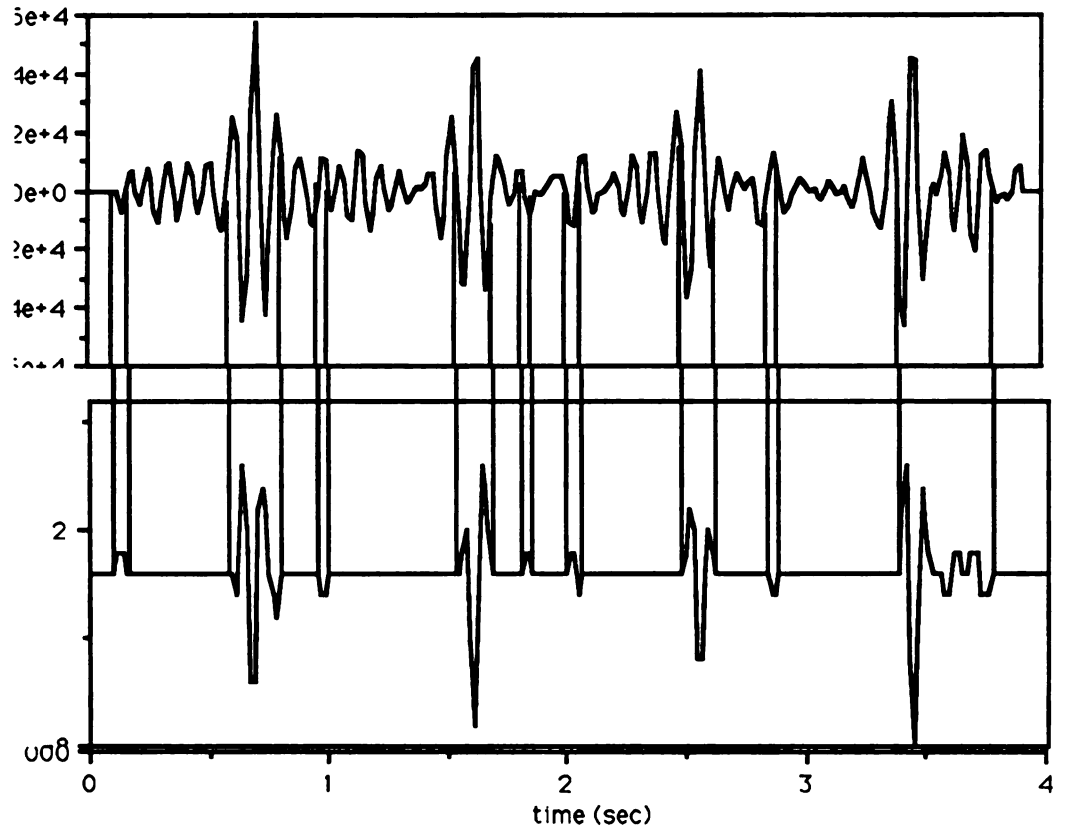


figure 40: derivative filter output determining where to examine correlation

These points are then normalized, the largest becoming 100, using integers, so that points smaller than 1/100 of the peak go to zero. An example is shown in figure 41.

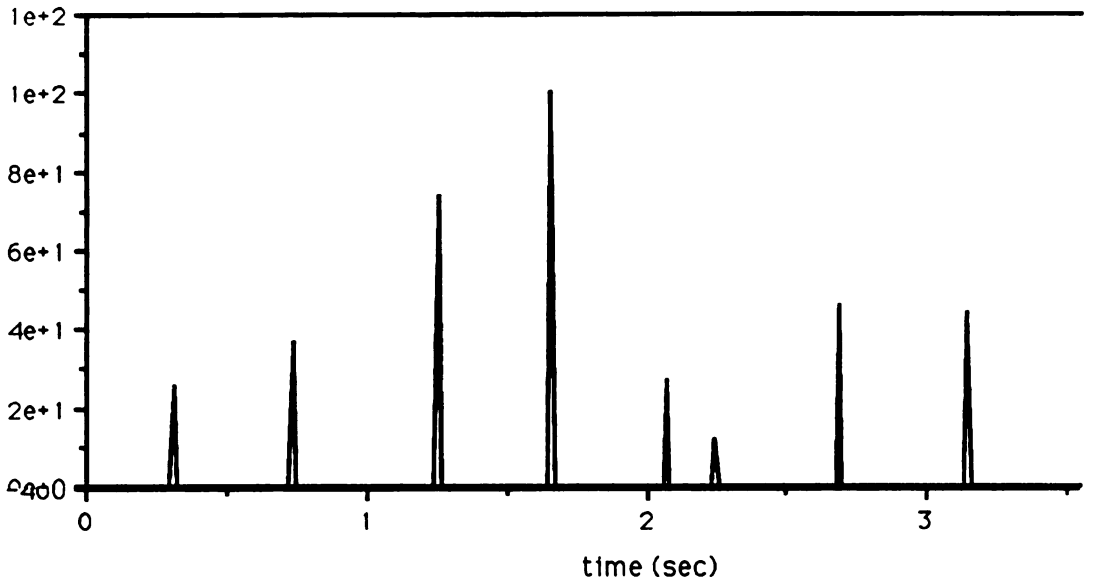


figure 41 : peaks of correlation in windows determined by derivative filter

One of the peaks in the correlation is the autocorrelation of the region selected. This value is taken as a reference for determining likeness in shape. Looking at ± some threshold about this autocorrelation value will give the sub-regions that are similar in shape to the sub-region selected. If there are at least two peaks within this threshold region, the algorithm will continue. If there is only one peak, the sub-region selected is said to be dissimilar from the other sub-regions and the algorithm proceeds to the next 4 second segment. If there are only two peaks within the threshold region, the distance or interval of these peaks is determined. If there are peaks separated by this interval through the 4 second segment, even if the peaks are outside the threshold region, the segment is said to contain heartbeats . If there are more than two peaks in the threshold region, each inter-peak interval is determined. Starting with the shortest interval, proceeding to the longer ones, if there are peaks in the 4 second segment separated by this interval, the segment is again said to contain heartbeats.

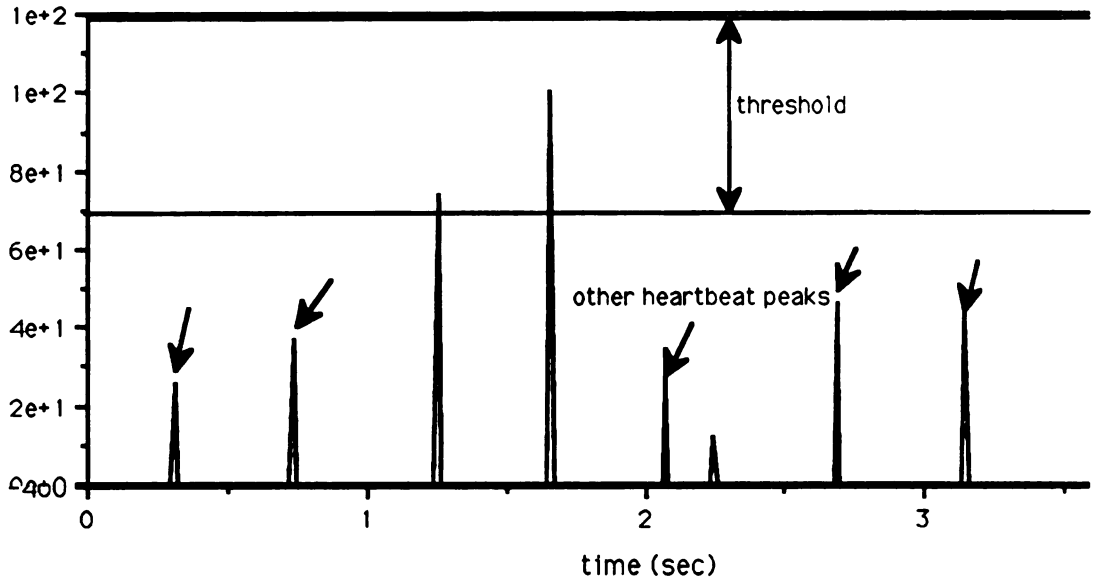


figure 42 : correlation output showing threshold and heartbeat peaks

This file of peaks is passed to the final pattern recognition scheme, and two features are used to determine if heartbeats are present, the amplitude of the output peaks of the correlation and and their periodicity. The algorithm followed is shown in figure 43.

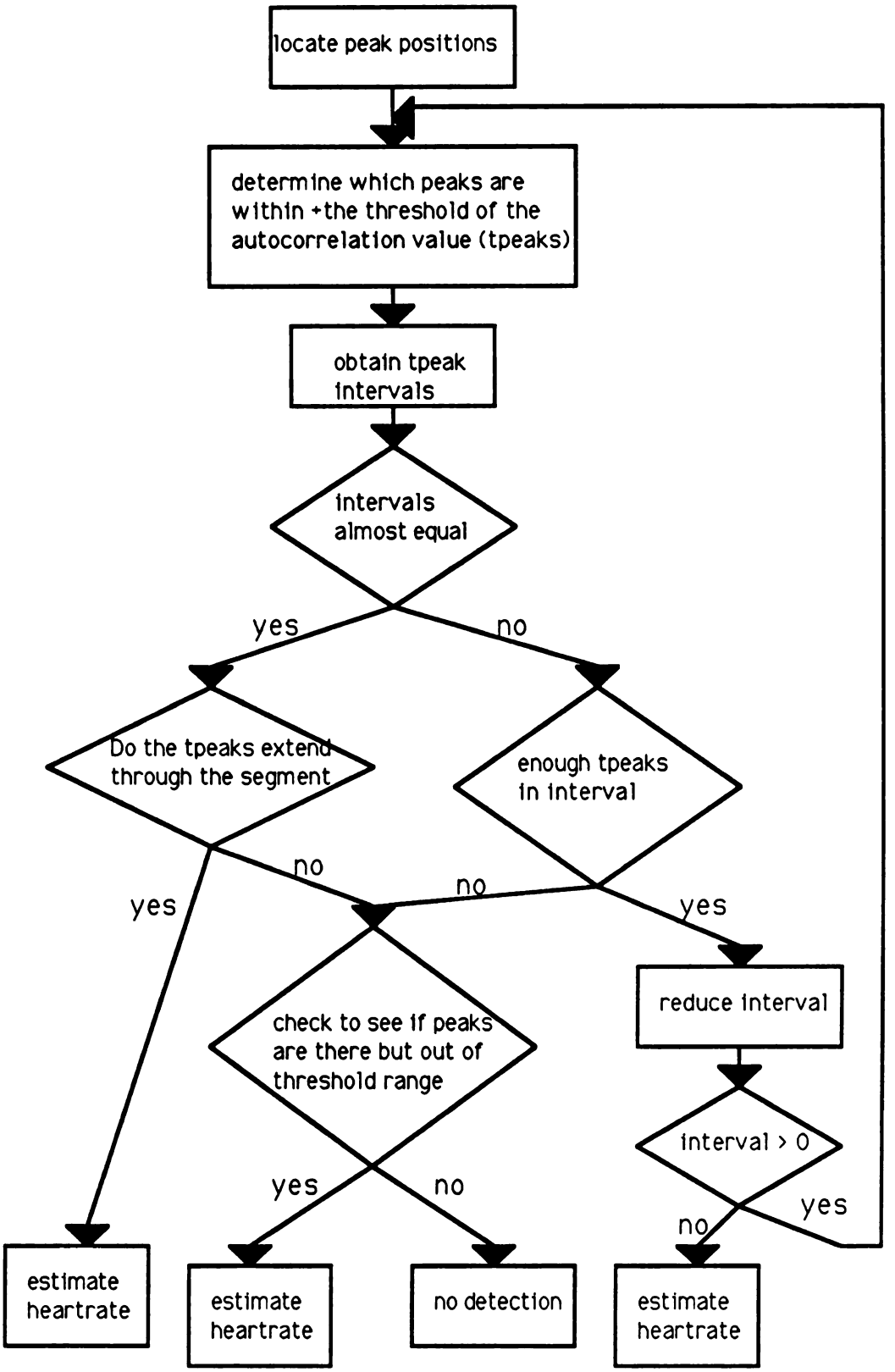


figure 43 : algorithm used for pattern recognition

A flow chart of the complete algorithm is shown in figure 44.

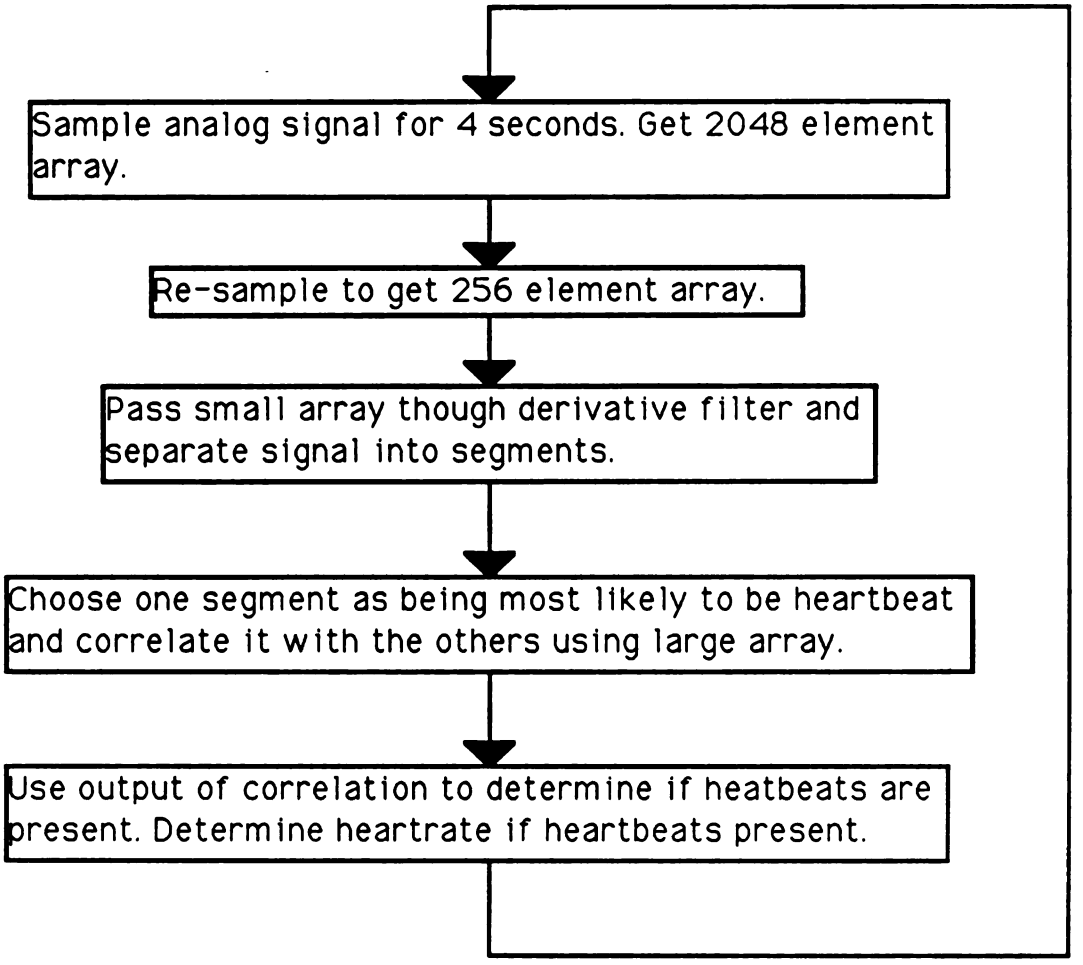


figure 44 : complete processing algorithm

Samples of complete processing are shown in figure 45 and figure 46. The top figure (a) is the original signal, (b) is the output of the derivative filter, (c) the correlation output, and (d) the normalized peaks of the correlation in the intervals determined by the derivative filter.

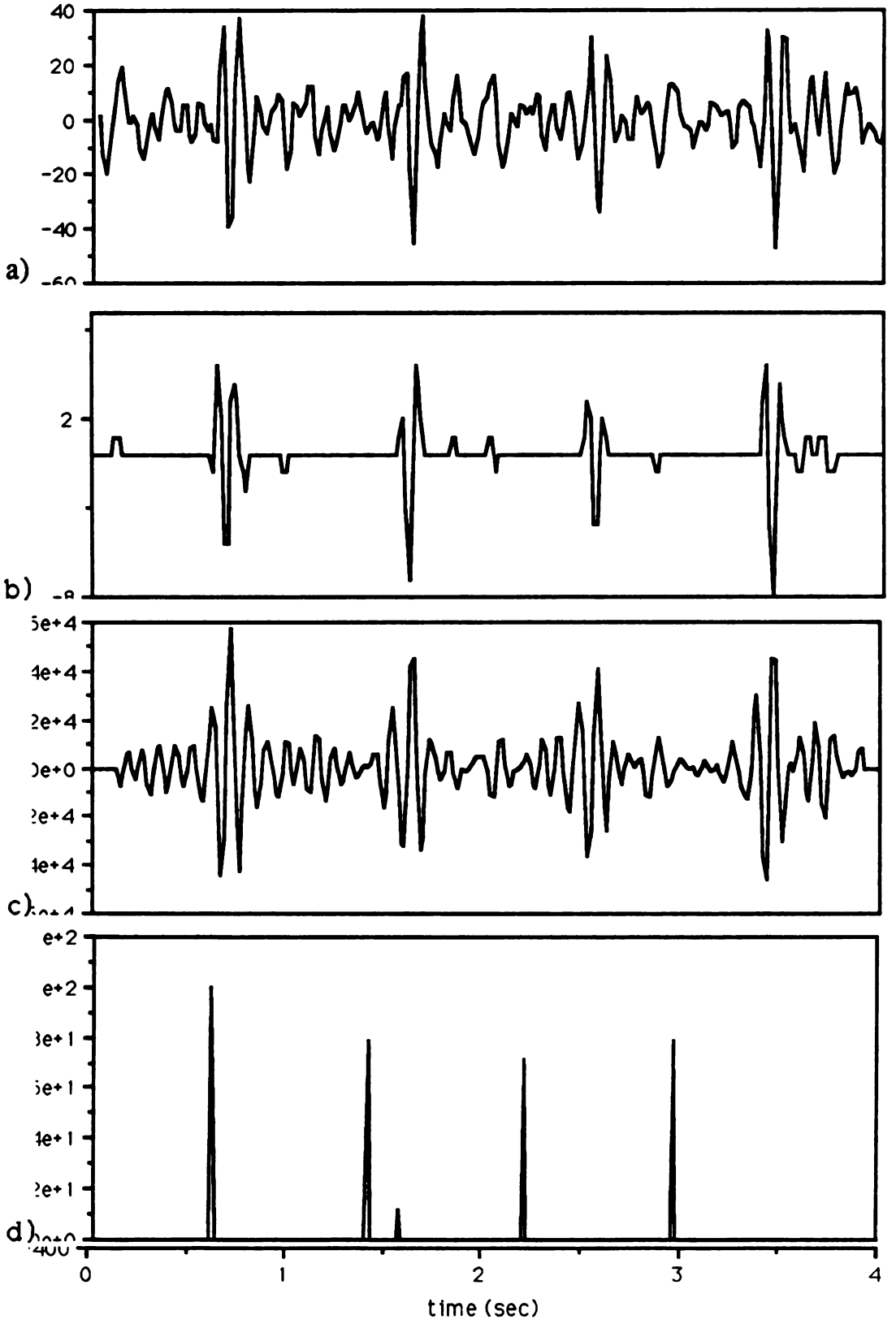


figure 45 : example of processing on human data

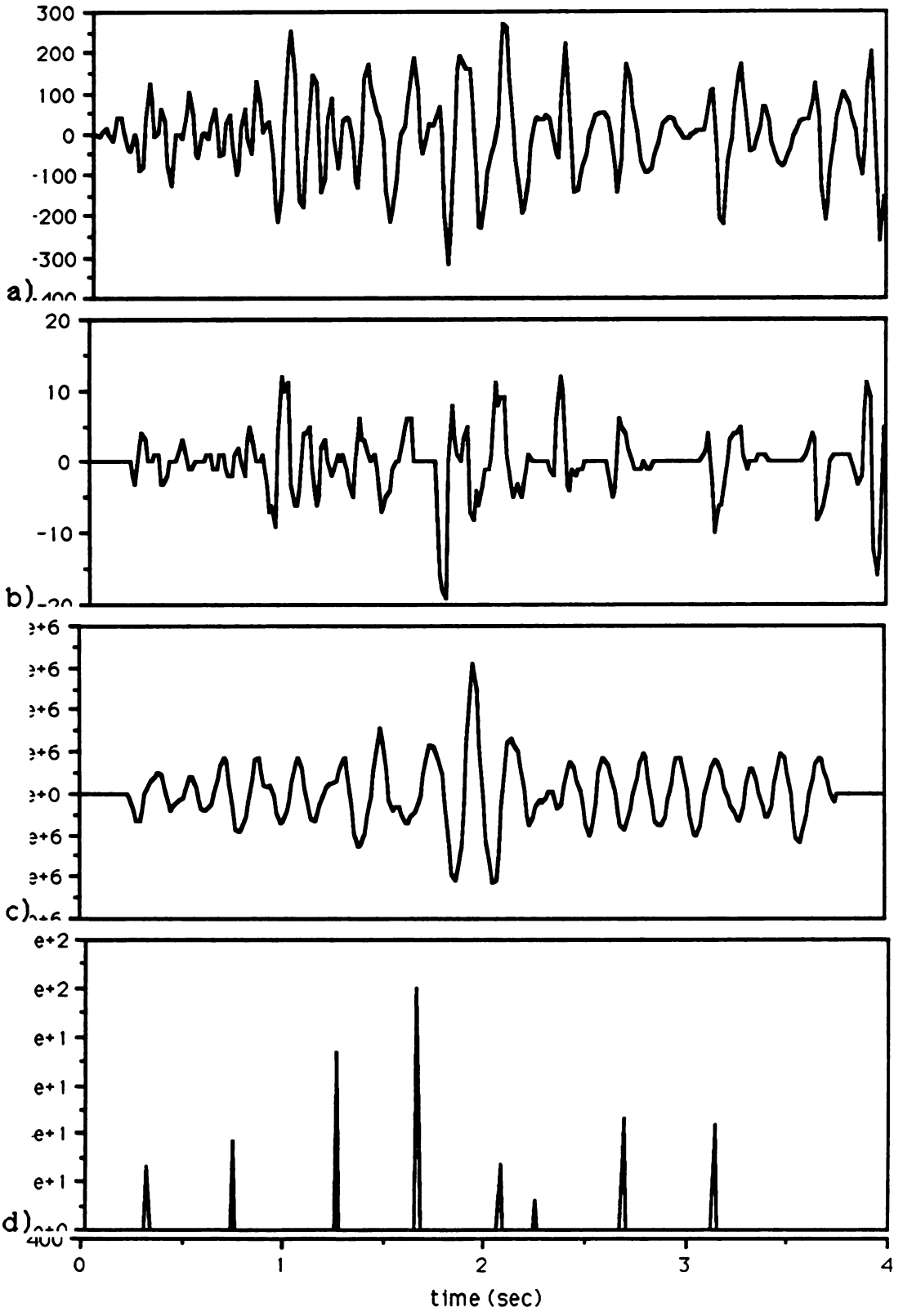


figure 46 : example of processing on human data

RESULTS

This section compares this algorithm with new thresholds (1), and old thresholds (2), with three others by having each classify heart data and noise (for enumeration and explanation of the thresholds see the programs in appendix A). One algorithm was developed by Albert Roseiro (3), the next is simply autocorrelation with a threshold (4), the last passes the data through a median filter then uses the power spectral density for detection and estimation (5). The results are shown below.

Eighty-four four second segments of heartbeat data and thirty five segments of noise were classified by the five algorithms. In the table below, correct detections means that the segments were classified as noise if noise, and as having a heartbeat if a heartbeat was present. Correct estimations means of the heartbeat segments correctly classified, the percentage of correct estimations was calculated. As can be seen, there is a trade-off between the percentage of correct detections and the percentage of correct estimations. In general, the higher the correct detections, the lower the correct estimations. This is expected because the lower number of correct detections when a heartbeat is present means the algorithm performs estimation only on relatively good segments.

algorithm	correct detections (heart)	correct estimations (of those correctly detected)	correct detections (noise)
1	43/84 (51.2%)	27/43 (62.8%)	30/35 (85.7%)
2	25/84 (29.8%)	18/25 (72%)	32/35 (91.4%)
3	20/84 (23.8%)	14/20 (70%)	32/35 (91.4%)
4	27/84 (32.1%)	11/27 (40.7%)	27/35 (77.1%)
5	65/84 (77.4%)	29/65 (44.6%)	22/35 (62.9%)

Deterministic signals were synthesized to compare the algorithms in a easily controlled manner. Four different signal structures were used. The first had one cycle of a sine wave repeated four times at equal intervals as shown in figure 47. In actual human measurements the heartbeat pulse was found to contain approximately 80 samples at 512 Hz sampling rate. The sine wave was chosen to be 6.4 Hz to give 80 points for one cycle of the sine wave. The peak of the sine wave pulse was 1000. The second structure placed the four sine wave pulses at different intervals. One was placed to start at sample 2, the next at sample 600 the third at sample 1039 and the last at sample 1551 also shown in figure 47. The third structure was the same as number two, except the sine wave pulses now had different frequencies.Gaussian noise of known variance was the added to the first three signal segments. The variance of the noise was increased until the algorithm could no longer give a correct estimate. The last segment used was actual heartbeat data with a good signal to noise ratio. The noise added was data from the microwave unit when this was pointed at inanimate objects. This noise was then scaled to different levels and added to the heartbeat data before the processing took place.

From the results in table 1 and 2, none of the four algorithms had problems determining the pulse period of the first signal before noise was added. After the noise was added, correlation and the median filter/power spectral density did very well. Algorithm 1 is close but came in third. Correlation does very well because the signal is perfectly periodic with exactly the same signal repeating. The Median filter/PSD algorithm does well because the median filter is very good at filtering gaussian noise, and because the sine wave pulses are placed at equal intervals.

To test these assertions the second segment with unequal intervals between sine pulses was used. As is seen it has no effect on the algorithm 1, but has a large effect on the Median/PSD and correlation algorithms. The correlation algorithm could not give the correct estimate even before noise was added, so the noise level given is the largest noise variance possible for the correlation algorithm to give the same estimate it gave before noise was added.

To try and put algorithm 1 at a disadvantage, the third segment was synthesized. The sine wave pulses now do not have the same period. Because algorithm 1 uses a partial correlation, it is very sensitive to zero crossing intervals. The algorithm was developed after noticing that even though the period and amplitude of the heartbeat changes dramatically, the zero crossing interval stays almost constant in actual measurements.

The last segment is an attempt to have a realistic comparison between the algorithms. Roseiro's algorithm had done poorly with the previous signal segments. The reason for it's usefulness is demonstrated here. It was, along with algorithm 1, developed for the noise and heartbeat signals from the microwave unit.

The results for the last signal segment are as expected. The results for the other signal sections were very encouraging, indicating that the algorithm presented may be used for a variety of signal shapes. This implies the algorithm could work in many different environments, even if the signal used for processing did not come from the microwave unit. The table below shows the maximum variance of the noise that could be added to the synthesized signal while the algorithm was still able to correctly estimate the heartrate. The second table shows the values in the first table normalized to one.

Table 1

algorithm	segment: periodic	non-periodic	non-periodic varying pulses	heart data
1	210,000	210,000	140,000	0.21
4	220,000	5,000	5,000	0.0
3	40,000	30,000	70,000	0.14
5	230,000	110,000	130,000	0.003

Table 2

algorithm	segment: periodic	non-periodic	non-periodic varying pulses	heart data
1	0.91	1.0	1.0	1.0
4	0.96	0.01	0.04	0
3	0.17	0.14	0.5	0.67
5	1.0	0.52	0.93	0.01

Algorithm 1: Derivative Filter-Pattern Recognition.
Algorithm 3: Roseiro's Algorithm.
Algorithm 4: Autocorrelaton.
Algorithm 5: Median Filter-Power Spectral Density.

Below the four deterministic signals before noise was added are shown.

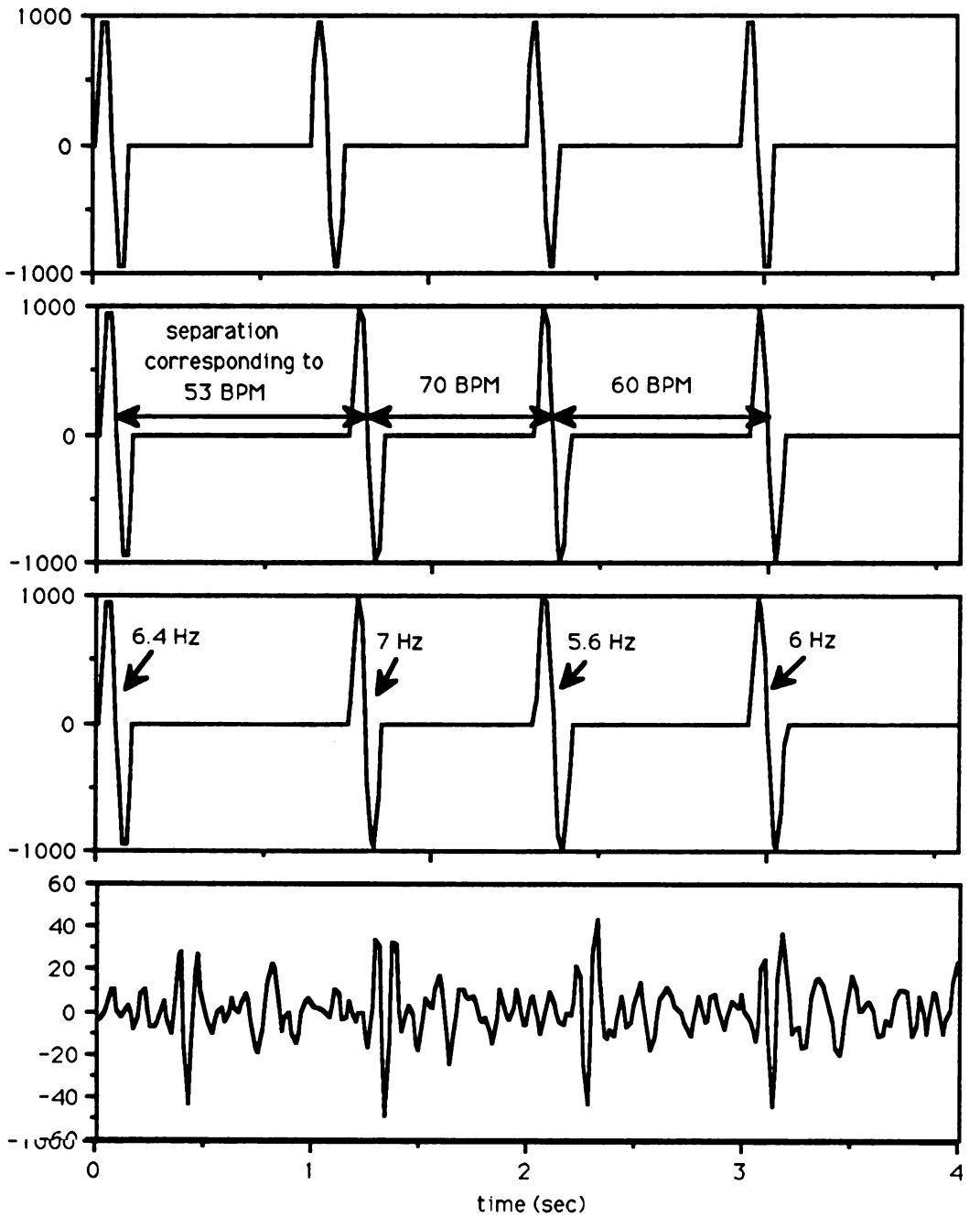


figure 47 : the four synthesized signals with no noise added

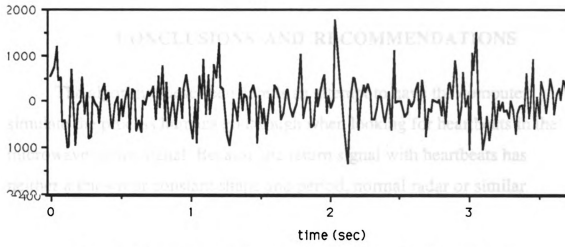


figure 48: segment 1 plus maximum noise for algorithm 1

CONCLUSIONS AND RECOMMENDATIONS

The algorithm described here is an attempt to have the computer simulate the process humans go through when looking for heartbeats in the microwave return signal. Because the return signal with heartbeats has neither a known or constant shape and period, normal radar or similar detection and estimation schemes will not work. This algorithm combines non-linear filtering with partial correlation and pattern recognition. A four second, 512 Hz sampling rate, 2048 element array is read in from the analog board and re-sampled to 256 samples. This small array is passed through the derivative filter to produce non-zero regions that give an indication of where the heartbeats are. To determine if there are heartbeats within the non-zero regions the following procedure is used. One of the regions is chosen as being the most likely to be a heartbeat, based on derivatives and amplitude. Using the large array, this region is correlated with the other regions and the output is used as an indication of similarity in shape. If there are non-zero regions that are similar in shape and are almost equidistant, these regions are said to be heartbeats.

As seen from test results, simple correlation will not work well except with a periodic heartbeat signal with a high signal to noise ratio. To help the signal to noise ratio, a pre-filtering to accentuate the hearts impulsive characteristic was developed. The idea for using integers to do the calculations and normalizations, thus producing the zero regions, came from Albert Roseiro. There is a large improvement using this pre-filter, and judging from our experience trying many other pre-filters, not much improvement can be expected in this area.

To be able to compare shapes adequately, the 512 Hz sampling rate had to be used.In this algorithm one region is chosen and correlated with the others to get an estimate of likeness of shape. The disadvantage of doing this is the possibility of choosing a non-zero segment that is not a heartbeat when heartbeats are present. Thus the major improvement in this area is a better technique for selecting the region for correlation. Another possibility is to use a lower sampling rate and having another method for measuring shape. This could be done by finding an equation that interpolates the points of the region, then compare equations of the non-zero regions. This may give a better estimate of the shape, and it also eliminates the need to select one region for use in correlation.

Only two features, the correlation output and periodicity are available for the pattern recognition following the correlation. The algorithm would be improved by finding more features. Many were tried, such as zero crossing intervals, but no additional features were found. The next improvement should be finding a better algorithm to determine if the output correlation file contains heartbeats.

Another possibility is to combine this algorithm with the one developed by Albert Roseiro.

As seen in the results presented,the algorithm developed in this report works well under many conditions, sometimes even outperforming visual inspection.

REFERENCES

1) J.C. Lin, J. Kiernicki, M. Kierniki and P.B. Wollshaeger, "Microwave apexcardiography," IEEE Trans. Microwave Theory Tech., vol. MTT-27,pp.618-620, June 1979

2) G. Hoshal, M. Siegel and R. Zapp, "A Microwave heart monitor and Life Detection System," Proceedings - Sixth annual Conference I.E.E.E Engineering in Medicine and Biology Society, Los Angeles, pp. 331-333, September 1982

3) M.A. Popovic, K. Chan and J.C. Lin, "Microprocessor-based Noncontact Heartrate/Respiration Monitor," ibid., pp. 754-755

4) A.G. Favret and A.F. Caputo, "Evaluation of autocorrelation Techniques for detection of fetal electrocardiograms," IEEE Trans. on Biomed. vol. BME-13, pp. 37-43, 1966

5) G. Hoshal, S. Ivkovich, M. Siegel, and R. Zapp, "Remote Noncontacting Heart Rate Measurement Using Microwave Signals and Digital Processing," Proceedings - Computers in Cardiology, Salt Lake City, Utah, pp. 409-412 ,September 1984

6) W. Byrne, P. Flynn, R. Zapp, and M. Siegel, "Adaptive Filter Processing in Microwave Remote Heart Monitors," IEEE Trans. on Biomed. vol. BME-33, no.7, pp. 717-722, July 1986

7) W. Byrne, M. Siegel, "Adaptive Filter Signal Processing in Microwave Heart Monitors,"

8) Personal communication with Greg Hoshal , Michigan State University

9) G. Hoshal, M. Siegel, " "Advances on Identification of Microwave Cardiograms for Remote Heart Rate Estimation", Technical Report,MSU-ENGR-86-003, February 1986.

10) A. Nehorai, and B. Porat, " Adaptive Comb filtering for Harmonic Signal Enhancement," IEEE Trans. on Acoustics, Speech, and Signal Processing, vol. ASSP-34 ,no. 5, October 1986, pp.1124-1138

11) L. Rabiner, S. Levinson, A. Rosenberg, J. Wilson, "Speaker-Independent Recognition of Isolated Words Using Clustering Techniques," IEEE Trans. on Acoustics, Speech, and Signal Processing, vol. ASSP-27,no.4, August 1975, pp. 336-349

12) R. Niederjohn, " A Mathematical Formulation and Comparison of Zero- Crossing Techniques which have been Applied to Automatic Speech Recognition," IEEE Trans. on Acoustics, Speech, and Signal Processing, vol. ASSP-23 ,no. 4, August 1975, pp. 373-379

13) S. Soyegh, Y. Kok, and J. Hong, " An Algorithm to find two-dimensional Signals with Specified Zero Crossings," IEEE Trans. on Acoustics, Speech, and Signal Processing, vol. ASSP-35 ,no. 1, January 1987, pp.107-111

14) L. Rabiner, M. Cheng, A. Rosenberg, C. McGonegal, " A Comparative Performance Study of Several Pitch Detection Algorithms," IEEE Trans. on Acoustics, Speech, and Signal Processing, vol. ASSP-24,no.5, October 1976, pp. 399-418

15) M. Sondhi, "New Methods of Pitch Extraction," IEEE Trans. Audio Electroacoust., vol. AU-16, pp. 262-266, June 1968

16) L. Rabiner, M. Sambur, C. Schmidt, " Applications of a non-linear Smoothing Algorithm to Speech Processing" IEEE Trans. on Acoustics, Speech, and Signal Processing, vol. ASSP-23,no.6, December 1975, pp. 552-557

17) G. Arce, and M. McLaughlin, " Theoretical Analysis of the Max/Median Filter," IEEE Trans. on Acoustics, Speech, and Signal Processing, vol. ASSP-35 ,no. 1, January 1987, pp. 60-69

18) J. Astola, P. Heinonen, and Y. Neuvo, " On Root Structured of Median and Med-type Filters," IEEE Trans. on Acoustics, Speech, and Signal Processing, vol. ASSP-35 ,no. 8, August, pp. 1199-1201

19) V. Rao, and K. Rao " A New Algorithm for Real-Time Median Filtering," IEEE Trans. on Acoustics, Speech, and Signal Processing, vol. ASSP-34 ,no. 6, December 1986, pp.1674-1675

20) R. Gagliardi, Introduction to Communications Engineering, Wiley-Interscience, New York, 1978

Appendix A

This is the main program. It accepts a 2048 point data segment, a 4 second segment sampled at 512 Hz. This program first removes the mean then sub-samples the 2048 point segment to 256 points, corresponding to a 64 Hz sampling rate. This segment then calls the processing subroutines for heartrate detection and estimation in order.

```
#include <math.h>
#include <stdio.h>

#define sr          512
#define seglen      2048
#define reduc       8
#define thresh1     35  /*threshold for acceptance of values
                    near autocorr*/
#define varthresh   90  /*thresh for variance*
#define thresh3     10  /*threshold for accepting small
                    correlations*/
#define thresh5     0.4 /*threshold for window in hiding
                    subroutine*/
#define thresh6     1.5 /*threshold for ends in hiding
                    subroutine*/
#define offend      20  /*threshold for missing subroutine*/


long  data[2050],temp1,der[260],w[2050],dat[260]; long  mean;
int   segment1,redseg;
int   ibeg[260],iend[260],ilarge,numseg,autocorr,segments,heartrate;
FILE  *ifp1,*fopen(),*ofp;

main()
{
float  sr2,fheartrate;
int    s,seg1,seg2,i,ns,sr1,c,bit,j,i1,k,store,sr3,heartrates;
char   fil_name[20],fils_name[20],store_fil[20],d,d1="";

printf("data file to process:");
scanf("%s",fils_name);
if((ifp1=fopen(fils_name,"r"))==NULL) exit(1);
fscanf(ifp1,"%d,%f,%d,%c%c %c,%c%c
       %c\n",&ns,&sr2,&c,&d,&d,&d,&d,&d,&d);
sr1=ns/seglen;
printf("\nnumber of samples=%d  number of data blocks=%d",ns,sr1);
printf("\nEnter starting and number of segments to be used:");
scanf("%d,%d",&seg1,&seg2);
```

```
sr3=(int)sr2;
sr3=sr3/reduc;
redseg=seglen/reduc;
segment1=seg2*(redseg);
s=seg1*seglen; for(j=0;j<s;++j){
            fscanf(ifp1,"%ld\n",&data[j]); }
for(s=0;s<seg2;++s){ for(j=0;j<seglen;++j){
            fscanf(ifp1,"%ld\n",&data[j]); } printf("\nsegment=
      %d",s+1); mean=0; for(i=0;i<seglen;++i){
            mean=mean+data[i]; } mean=mean/seglen;
      for(i=0;i<seglen;++i){
            data[i]=data[i]-mean; }
      for(j=0,k=0;j<redseg;++j,k=k+reduc){
            dat[j]=data[k]; } derivative(dat,der);
      filter(dat,der); segments=segment(dat,der);
      if(segments>49){goto out;} correlate(dat,der,data);
      heartrates=pat(der,autocorr,thresh1); if(heartrates<19 ||
      heartrates>72){
          fheartrate=0.0; } else{
          fheartrate=(3840.0)/(float)heartrates; }
      printf("\nheartrate=%d",(int)fheartrate);
out:;
}
fclose(ifp1);
}
```

This subroutine accepst a down-sampled version of the 2048 length to 256 elements. The output is the derivative filtered signal, normalized to +8.

```
filter(in,out)
long  in[260],out[260];
{
int   i,j,k;
long  peak,ten=10,hpeak,f1,f2;

       for(i=1;i<redseg-1;++i){
              f1=in[i+1]-in[i];
              f2=in[i]-in[i-1];
              w[i]=f1-f2; } w[0]=0; w[redseg-1]=0;

       for(i=0;i<redseg;++i){
              out[i]=in[i]*labs(out[i]);
              w[i]=in[i]*labs(w[i]); } for(i=0;i<redseg-2;++i){
              if(w[i] == 0 && w[i+1] == 0 && w[i+2] == 0)
                     out[i+1]=0;

              } } peak=0; for(i=0;i<redseg;++i){
              if(peak<labs(out[i])){
                     peak=labs(out[i]);
              } } peak=(peak+4)/8; if(peak==0){peak=1;}
       for(i=0;i<redseg;++i){
              out[i]=out[i]/peak; } for(i=1;i<redseg-1;++i){
              if(out[i] != 0 && out[i-1] ==0 && out[i+1] == 0){
                     out[i]=0;
              } }
}

derivative(in,out)
long in[260],out[260];
{
int   i,j,k,oldi,newi;
long  diff,oldpeak,newpeak;

newi=0;
newpeak=0;

for(i=0;i<redseg;++i){ out[i]=0;
}

for(i=1;i<redseg-1;++i){ if(in[i]>=in[i-1] && in[i]>=in[i+1]
              oldi=newi;
              newi=i;
              diff=newi-oldi;
              oldpeak=newpeak;
```

```
            newpeak=in[i];
            jeva(oldpeak,newpeak,diff,i,oldi); } else
        if(in[i]<=in[i-1] && in[i]<=in[i+1]){
            oldi=newi;
            newi=i;
            diff=newi-oldi;
            oldpeak=newpeak;
            newpeak=in[i];
            jeva(oldpeak,newpeak,diff,i,oldi); } else if(in[i]==0
        && in[i-1]==0 && in[i+1]==0){
            oldi=newi;
            newi=i;
            diff=newi-oldi;
            oldpeak=newpeak;
            newpeak=in[i];
            jeva(oldpeak,newpeak,diff,i,oldi); }
}

for(i=0;i<redseg-1;++i){ if(out[i] != out[i+1]){
            out[i+1]=(out[i]+out[i+1]+1)/2; }
}

}

jeva(oldpeak,newpeak,diff,i1,i2)
long  oldpeak,newpeak,diff;
int   i1;
{
int   i;
long  slope,magdiff,hdiff;

magdiff=labs(newpeak-oldpeak);
hdiff=(diff+1)/2;
slope=(magdiff+hdiff)/diff;
for(i=i2;i<i1;++i){ der[i]=slope;
}
}
```

This subroutine is used in the normalization process to return a 1 if the input is positive, a -1 if it is negative.

```
sign(in)
long in;
{
        if(in<0){return(-1);}
        else{return(1);}
}
```

```
        This subroutine inputs the derivative filter output and
        determines the regions of possible heartbeats and chooses the
        region most likely to be a heartbeat.

int segment(dat,der)
long dat[260],der[260];
{
int   i,j,k,ider1,ider2,ider3,corrlen,repeats;
long  peak,peak2,peak3,datpeak1,datpeak2,datpeak3;
long  diff1,diff2,diff3,avpeak1,avpeak2;

    for(i=0;i<redseg;++i){
        ibeg[i]=0;
        iend[i]=0;
    }

/* determine segment boundaries */

    peak=0;
    for(i=3,numseg=0;i<redseg;++i){
        if(der[i-3]==0 && der[i-2]==0
                        && der[i-1]==0 && der[i]!=0){
            ibeg[numseg]=i;
            for(;i<redseg-3;++i){
                if(der[i]!=0 && der[i+3]==0
                        && der[i+2]==0 && der[i+1]==0){
                    iend[numseg]=i;
                    ++numseg;
                    goto out;
                }
            }
        }
out:;
    }
    if(numseg>49){goto end;}

/* determine segment used in correlation */
    peak=0;
    peak2=0;
    peak3=0;
    for(i=0;i<numseg;++i){
        corrlen=iend[i]-ibeg[i];
        if(corrlen>5 && corrlen<17){
            for(k=ibeg[i];k<=iend[i];++k){
                if(peak<der[k]){
                    peak=der[k];
                    ider1=i;
                }
                if(peak>der[k] && peak2<der[k]){
```

```
                peak2=der[k];
                ider2=i;
            }
            if(peak>der[k] && peak2>der[k] && peak3>der[k]){
                peak3=der[k];
                ider3=i;
            }
        }
    }
}
datpeak3=0;
datpeak1=0;
datpeak2=0;
if(peak>0){
    for(k=ibeg[ider1];k<=iend[ider1];++k){
        if(datpeak1<dat[k]){
            datpeak1=dat[k];
        }
    }
}
if(peak>0){
    for(k=ibeg[ider2];k<=iend[ider2];++k){
        if(datpeak1>dat[k] && datpeak2<dat[k]){
            datpeak2=dat[k];
        }
    }
}
if(peak2>0){
    for(k=ibeg[ider3];k<=iend[ider3];++k){
        if(datpeak1>dat[k] && datpeak2>dat[k] && datpeak3<dat[k]){
            datpeak3=dat[k];
        }
    }
}

ilarge=-1;

avpeak1=(peak+peak2+peak3)/3;
avpeak2=(datpeak1+datpeak2+datpeak3)/3;
if(avpeak1 ==0 ){avpeak1=1;}
if(avpeak2 ==0 ){avpeak2=1;}

diff1=(peak/avpeak1)+(datpeak1/avpeak2);
diff2=(peak2/avpeak1)+(datpeak2/avpeak2);
diff3=(peak3/avpeak1)+(datpeak3/avpeak2);

if(diff1>=diff2 && diff1>=diff3){ilarge=ider1;}
else if(diff2>diff1 && diff2>diff3){ilarge=ider2;}
else if(diff3>diff1 && diff3>diff2){ilarge=ider3;}

if(ilarge<0){
```

```
        goto end;
    }
end: return(numseg);
}
```

This subroutine inputs the original 2048 element segment and correlates the segment choosen by the segment subroutine with all the other possible heartbeat regions. The peak of the correlation in each possible segment is the output.

```
correlate(dat,der,data)
long dat[260],der[260],data[2050];
{
int   i,j,k,s,corrlen,corrlen2;
long  peak,hpeak;
float fpeak;

   peak=0;
   corrlen=iend[ilarge]*reduc-ibeg[ilarge]*reduc+reduc;
   corrlen2=corrlen/(2*reduc);
   for(i=0;i<corrlen2;++i){
       dat[i]=0l;
   }
   for(i=0,j=ibeg[ilarge]*reduc;i<corrlen;++i,++j){
       w[i]=data[j];
   }
   for(s=0,i=0;s<seglen-corrlen;++i,s+=8){
       for(j=0,templ=0;j<corrlen;++j){
           templ+=(data[j+s]*w[j]);
       }
       dat[i+corrlen2]=templ;
   }
   for(i=redseg-corrlen2;i<redseg;++i){
       dat[i]=0l;
   }
   for(i=0;i<redseg;++i){
       der[i]=0;
   }
   for(i=0;i<numseg;++i){
       peak=-100000;
       for(k=ibeg[i];k<iend[i];++k){
           if(peak<dat[k]){
               peak=dat[k];
               j=k;
           }
       }
       if(peak>0){
           der[j]=peak;
       }
       if(i==ilarge){autocorr=j;}
   }
   fpeak=(float)der[autocorr];
   fpeak=(fpeak)/100.0;
```

```
    if(fpeak==0.0){fpeak=1.0;}
    for(i=0;i<redseg;++i){
        der[i]=(long)((float)der[i]/fpeak);
    }
end:;
}
```

This subroutine inputs the peaks of the autocorrelation and uses the peak amplitude and their separation to determine if heartbeats are present.

```
int pat(der,autocorr,peakthresh)
long  der[260];
int   autocorr,peakthresh;
{
   int    i,j,k,corr[50],diff[50],ihigh[50],numpeak,numbeats;
   int    avdiff,numdiff,smalldiff,ipeak,imedian,thresh4,largediff;
   int    oldipeak,sumbeats;
   long   x,var,peak;

   if(heartrate==0){heartrate=64;}
   for(i=0;i<redseg;++i){
       if(der[i]!=0){
           printf("\ni=%d,der[i]=%ld",i,der[i]);
       }
   }
begin:;
   for(i=0,numpeak=0;i<redseg;++i){
       if(der[i]>=(100-peakthresh) && der[i]<=(100+peakthresh)){
           ihigh[numpeak]=i;
           ++numpeak;
       }
   }
printf("\nnumpeak=%d",numpeak);
   if(numpeak>40 || numpeak<2){
       heartrate=0;
       goto end;
   }
   numdiff=numpeak-1;
   for(i=0,avdiff=0,smalldiff=1000,largediff=0;i<numdiff;++i){
       diff[i]=ihigh[i+1]-ihigh[i];
       avdiff+=diff[i];
       if(smalldiff>diff[i]){smalldiff=diff[i];}
       if(largediff<diff[i]){largediff=diff[i];}

printf("\ndiff[i]=%d ihigh[i]=%d",diff[i],ihigh[i]);

   }
   imedian=diff[numdiff/2];
   avdiff /= (numdiff);
printf("\navdiff=%d smalldiff=%d",avdiff,smalldiff);
   for(i=0,var=0;i<numdiff;++i){
       var+=((diff[i]-avdiff)*(diff[i]-avdiff));
   }
   var/=numdiff;
```

```
printf("\nvar=%ld",var);
printf("\nimedian=%d",imedian);
   if(var<varthresh){
        goto missing;
   }
   else{
        if(avdiff<30){
             peakthresh-=3;
             if(peakthresh<=0){
                  heartrate=0;
                  goto end;
             }
             goto begin;
        }
        goto hiding;
   }

missing:;

printf("\nmissing");

   if( (ihigh[0]-avdiff-offend)<0 &&
      (ihigh[numdiff]+avdiff+offend)>256 ){
        heartrate=avdiff;
        goto end;
   }
   else{
        goto hiding;
   }

hiding:;

printf("\nhiding");

   i=autocorr-heartrate;
   thresh4=heartrate*thresh5;
printf("\nautocorr=%d",autocorr);
   numbeats=0;
   sumbeats=0;
   oldipeak=autocorr;
   while(i>(thresh4*thresh6)){
printf("\ni=%d heartrate=%d thresh4=%d",i,heartrate,thresh4)
        for(j=i-thresh4,peak=0;j<i+thresh4;++j){
             if(peak<der[j]){
                  peak=der[j];
                  ipeak=j;
             }
        }
printf("\n1 peak=%d",peak);
        if(peak<thresh3){
             goto down;
```

```
        }
        i=ipeak;
        i-=heartrate;
        sumbeats+=oldipeak-ipeak;
        ++numbeats;
        oldipeak=ipeak;
    }
    i=autocorr+heartrate;
    oldipeak=autocorr;
    while(i<255-(thresh4*thresh6)){
printf("\ni=%d heartrate=%d thresh4=%d",i,heartrate,thresh4);
        for(j=i-thresh4,peak=0;j<i+thresh4;++j){
            if(peak<der[j]){
                peak=der[j];
                ipeak=j;
            }
        }
printf("\n2 peak=%d",peak);
        if(peak<thresh3){
            goto down;
        }
        i=ipeak;
        i+=heartrate;
        sumbeats+=ipeak-oldipeak;
        ++numbeats;
        oldipeak=ipeak;
    }
    if(numbeats!=0){
        heartrate=sumbeats/numbeats;
    }
end:;
    return(heartrate);

down:;
printf("\ndown");
    if(avdiff != heartrate && avdiff != 0){
        heartrate=avdiff;
        goto hiding;
    }
    if(smalldiff != heartrate && smalldiff != 0){
        heartrate=smalldiff;
        avdiff=heartrate;
        goto hiding;
    }
    if(largediff != heartrate && largediff != 0){
        heartrate=largediff;
        avdiff=heartrate;
        smalldiff=heartrate;
        goto hiding;
    }
    else{
```

```
            heartrate=0;
            goto end;
        }
    }
```

This program inputs a segment of ASCII up to 512 samples and performs a median filtering then a FFT on the data. The power spectral density is computed. The interpeak distances of the PSD fileare determined. The interpeak distance ocurring with the most frequency is determined to be the heartbeat period. The mainprogram below inputs the ASCII characters and calls the subroutines.

```
#include <stdio.h>
#include <math.h>

 /* median filtering and power spectral density */

float data[512],temp1,mean;
float w[512],YY1[2][2048],YY2[2][2048];
float tp,psdmax,fest,startf,endf;
int   segment1,sr,seglen,window;
char  fil_name[20],fils_name[20],store_fil[20],d,d1="";


main()
{
calculate_window();
median();
psd();
}
```

This subroutine performs the median filtering.

```
median()
{
int *fp,s,seg1,seg2,i,ns,c,bit,one,j,i1,k,store;
double sqrt();
float xy,xi,x2,yi,y2,dat[256],rho,EY,EX,sigmax,sigmay,a,b,max,min;
FILE *ifp,*ifp1,*fopen(),*ofp;

printf("data file to median filter:");
scanf("%s",fils_name);
if((ifp1=fopen(fils_name,"r"))==NULL)
        exit(1);

printf("\nEnter 1 to store filtered signal");
scanf("%d",store);
if(store == 1){
        printf("Name of storing file:");
        scanf("%s",store_fil);
        if((ofp=fopen(store_fil,"w"))==NULL)
                exit(1);
        }
fscanf(ifp1,"%d,%d,%d,%c%c %c,%c%c %c\n",&ns,&sr,&c,&d,&d,&d,&d,&d,&d,&d);
printf("\nEnter segment length:");
scanf("%d",&seglen);
sr=ns/seglen;
printf("\nsegment length:%d",seglen);
printf("\nnumber of samples=%d  number of data blocks=%d",ns,sr);
printf("\nEnter starting and number of segments to be used");
scanf("%d,%d",&seg1,&seg2);
printf("\nEnter window:");
scanf("%d",&window);
segment1=seg2*(seglen);
printf("\nsegment1=%d",segment1);
if(store == 1){
fprintf(ofp,"%d,64,0,%cT %c,%cR %c\n",segment1,d1,d1,d1,11);
}

s=seg1*seglen;
        for(j=0;j<s;++j){
                fscanf(ifp1,"%f\n",&data[j]);
        }
for(s=0;s<seg2;++s){
        for(j=0;j<seglen;++j){
                fscanf(ifp1,"%f\n",&data[j]);
        }
        mean=0.0;
        for(i=0;i<seglen;++i){
                mean=mean+data[i];
        }
```

```
        mean=mean/seglen;
        for(i=0;i<seglen;++i){
                data[i]=data[i]-mean;
        }
        for(i=-window;i<0;++i){
                data[i]=data[0];
        }
        for(i=seglen;i<(seglen+window);++i){
                data[i]=data[seglen-1];
        }
        for(i=0;i<seglen;++i){
                for(j=0;j<2*window+1;++j){
                        dat[j]=data[j+i-window];
                }
                for(j=0;j<2*window;++j){
                    for(k=j+1;k<2*window+1;++k){
                        if(dat[k] < dat[j]){
                                templ=dat[j];
                                dat[j]=dat[k];
                                dat[k]=templ;
                        }
                    }
                }
                YY1[1][i+1]=dat[window];
        }
}
fclose(ifp1);
if(store == 1){
fclose(ofp);
}
}
```

This subroutine calculates a hanning window used in the FFT

```
calculate_window()
{
        double cos(),zarg;
        float sk,sc,ti;
        register int i,nsoff,iend;
        sk = 0.5;
        nsoff = seglen / 2.0;
        i = 0;
        iend = seglen / 2.0;
        while(i < iend){
                zarg = tp * i / seglen;
                sc= sk + sk * cos(zarg);
                w[nsoff+ 1 + i] = sc;
                w[nsoff - i] = sc;
                i++;
        }
        w[1] = 0.0;
}
```

This suproutine caclulates the PSD

```
psd()
{
        tp = 6.2831853;
        windows();
        fft();
        magnitude();
}

windows()
{
        register int i;
        i = 0;
        while(i < seglen){
                i++;
                YY1[1][i] = YY1[1][i] * w[i];
                YY1[2][i] = 0.0;
                YY2[1][i] = 0.0;
                YY2[2][i] = 0.0;
         }
}
```

This suproutine caclulates the FFT

```
fft()
{
        int p,k,im,p11,p22,p33,id,ic,inh,ia,ib;
        int il;
        float bi,br,ti,zi,zr;
        double zsr,q,log2,log(),sin(),cos(),two;
        two=2.00;
        log2=log(two);
        zsr = seglen;
        q = log(zsr);
        zsr = q/log2;
        ti = tp / seglen;
        il= zsr;
        ia= seglen / 2;
        ib = 1;
        inh=ia;
        p = 0;
        while(p < il){
                p++;
                ic = 0;
                id=ia;
                k = 0;
                while(k < ib){
                        k++;
                        q = ti * ic;
                        zr = cos(q);
                        zi = - sin(q);
                        im=ic;
                        while(im < id){
                                im++;
                                p11=im + id;
                                p22=im + ic;
                                p33=im + inh;
                                br= zr * YY1[1][p11] - zi * YY1[2][p11];
                                bi = zr * YY1[2][p11] + zi * YY1[1][p11];
                                YY2[1][im] = YY1[1][p22] + br;
                                YY2[2][im] = YY1[2][p22] + bi;
                                YY2[1][p33] = YY1[1][p22] - br;
                                YY2[2][p33] = YY1[2][p22] - bi;
                        }
                        ic=id;
                        id=id + ia;
                }
                ia=ia / 2;
                ib = 2 * ib;
                im = 0;
                while(im < seglen){
```

```
                    im++;
                    YY1[1][im] = YY2[1][im];
                    YY1[2][im] = YY2[2][im];
                }
            }
        }
```

This suproutine is used by the PSD subroutine

```
magnitude()
{
        register int i;
        float sk;
        FILE *ofp;

        sk= 2.0 / (sr * seglen);
        i = 0;

        printf("\nName of storing psd file:");
        scanf("%s",store_fil);
        if((ofp=fopen(store_fil,"w"))==NULL)
                exit(1);

        fprintf(ofp,"%d,64,0,%cT %c,%cR %c\n",segment1,d1,d1,d1,d1);

        while(i < seglen){
                i++;
         YY2[1][i] = sk * (YY1[1][i] * YY1[1][i] + YY1[2][i] * YY1[2][i]);
                fprintf(ofp,"%f\n",YY2[1][i]);
        }
        fclose(ofp);
}
```

This suproutine performs the peak search of the PSD values

```
peak_search()
{
        register  int i,imax,i1,i2;
        float psdmax,fest;
        i1 = startf * seglen / sr;
        i2 = endf * seglen / sr;
        printf("\nstart,finish= %d,%d\n",i1,i2);
        psdmax = 0.0;
        imax=0;
        i = i1-1;
        while(i<i2){
                i++;
                if(YY2[1][i] > psdmax){
                psdmax = YY2[1][i];
                imax = i-1;
        }
}
fest = (float)imax * (float)sr / (float)seglen;
}
```

This subroutine performs a correlation on an input ASCII segment and compares the first peak after zero to a threshold. If the peak is larger than the threshold, the distance from zero to the first peak is taken as the hearbeat period.

```
#include <stdio.h>


float dat[260],r[260],threshold;
int sr,ns,lmin,lmax,lrmax,lag,nseg,segnum,numseg,locat,c;
int wsize,cwsize,bmin,bmax,imax,rect,izr,ss,ff,sum;
double sqrt(),var,fabs(),rvar,bvar;
float hrest,bavg,ravg;
char fil_name[20],d,store_fil[20],d1="";
float sum1,sum2,sum3,mean,var1,var2,s,rmax,mean2,taper,scale;
FILE *ifp,*fopen(),*ofp;

main()
{
int i,j,k;

bavg=0.0;
bvar=0.0;
rvar=0.0;
ravg=0.0;
i=0;
izr=0;
wsize=256;
printf("file to correlate:");
scanf("%s",fil_name);
if((ifp=fopen(fil_name,"r"))==NULL)
      exit(1);
fscanf(ifp,"%d,%d,%d,%c%c %c,%c%c %c\n",&ns,&sr,&c,&d,&d,&d,&d,&d,&d);
printf("# of samples=%d  # of 256 blocks=%d\n",ns,ns/256);
printf("\nnumber of blocks:");
scanf("%d,%d",&nseg);
sum=nseg*256;
printf("Full wave rectify?(1=yes,0=no):");
scanf("%d",&rect);
printf("rect=%d\n",rect);
lmin=0;
lmax=256;
segnum=0;
mean=0.0;
locat=0;
while(segnum<nseg){
  printf("\nsegment#=%d",segnum);
   i=0;
```

```
    while (i<256){
            i=i+1;
            fscanf(ifp,"%f\n",&dat[i]);
    }

    corr();

        i=0;
        rmax=0;
        i=lmin+1;
        segnum=segnum+1;
}
fclose(ifp);
}
```

This subroutine performs the autocorrelation of the input file and compares it to a threshold.

```
corr()
{

int i,j,k;
float max,dmax,databs;

        i=0;
        mean=0.0;
        while(i<wsize){
                ++i;
                mean+=dat[i];
        }
        mean=mean/(float)wsize;
        i=0;

        sum1=0.0;
        sum2=0.0;
        mean2=0.0;
        while(i<wsize){
                i=i+1;
                dat[i]=dat[i]-mean;
                databs=fabs(dat[i]);
                if(dmax<databs){
                    dmax=databs;
                }
                if(rect>izr){
                        dat[i]=databs;
                        mean2=mean2+dat[i];
                }
        }
        mean2=mean2/(float)wsize;
        if(rect>izr){
                i=0;
                while(i<wsize){
                        i=i+1;
                        dat[i]=dat[i]-mean2;
                }
        }
        cwsize=wsize-lmin;
        i=0;
        lag=lmin;
        while(lag<lmax+1){
                sum3=0;
                i=0;
                j=wsize-lag;
                while(i<j){
```

```
                    i=i+1;
                    sum3=sum3+dat[i]*dat[i+lag];
                }
                s=j;
                s=s*s;
                r[lag+1]=sum3;
                lag=lag+1;
            }

        max=0;
        for(i=30;i<=lmax;++i){
            if(max<r[i]){
                max=r[i];
                imax=i;
            }
        }
    printf("\nmax=%f dmax=%f",max,dmax);
        threshold=max/dmax;
        if(threshold>1600){
            printf("\nheartrate=%d",3840/imax);
        }
        else{
            printf("\nheartrate=0");
        }
    }
```