CHALMERS

Heart rate variability estimation and data visualisation for use in stress level determination in neuro-intensive care patients

Master of Science Thesis

EIJA JOHANSSON
TOBIAS JOHANSSON

Department of Signals and Systems
Division of Signal Processing and Antennas
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden, 2009
Report No. EX058/2009

Abstract

A lot of research is being made on extracting useful information from biomedical measurements, and signal processing has been found to be a powerful diagnostics tool. We describe herein our effort to create a usable software implementation of certain analysis methods. As part of an ongoing project intending to study the effects that stress during care time can have on the final outcome of the treatment of neuro-intensive care patients, our tool is intended to provide estimates of the stress level in a patient. The focus is on the development of three main parts: algorithms for extracting estimates of stress levels based on Heart Rate Variability (HRV), means for visualisation of biomedical signals, and finally, an extendible framework for signal processing of this kind.

The result is a piece of software where the user is presented with three different running estimates of stress-level. One is a statistical measure of HRV and the other two measure the level of correlation between variations in heart rate and the respiratory cycle. The correlation is determined, in one case using Fast Fourier Transform (FFT), and in the other, by a model-based approach using Kalman filtering. The information can be viewed over great ranges of scale in an easily browsable visualisation interface. At the foundation is our configurable and extendible library for signal analysis over large data sets. It is built on a concept of computation graphs of composable sub-algorithms, enabling new processing methods to be built with ease; both by creating new computation graphs, and by implementing new sub-algorithms.

We used our program on recorded data from several patients undergoing neuro-intensive care and found that our two correlation-based estimates show similar trends. The estimates show large variation, both from patient to patient, and over time in the same patient. These variations have not yet been satisfactorily linked to any other information about the patient, e.g. medication dosage or general observations. However, for the patients considered, the amount of stress-related information, was rather low and the medical links are hopefully due for further investigation.

Contents

1 Introduction **3**

2 Technical background **5**
2.1 ECG and heart rate analysis 5

3 Methodology **7**

4 Signal processing algorithms **8**
4.1 Peak detection 8
4.2 Forming the RR-interval signal 11
4.3 Statistical measures 11
4.4 Correlation with breathing cycle 12
4.4.1 Spectral analysis of the RR-interval signal 12
4.4.2 Model-based approach 14
4.5 Removing outlying RR-intervals 17

5 Software implementation **18**
5.1 Extendibility 18
5.1.1 Signals 19
5.1.2 Algorithm configuration 19
5.1.3 Implemented signal types 21
5.2 File formats 26
5.2.1 Saving signals 27
5.2.2 Equidistantly timed samples 27
5.2.3 Non-equidistantly timed samples 28
5.3 Performance tradeoffs 28
5.4 Visualisation 29
5.4.1 Signal plot items 30

5.4.2 Auxiliary plotting tools 30

6 Applications **32**
6.1 Detail studies 32
6.2 Large-scale analysis 34

7 Discussion **43**

1

Introduction

Traumatic brain injury (TBI) affects many, in other respects perfectly healthy, young individuals. Subarachnoid Haemorrhage (SAH) is another serious complication causing damage to the brain. Severe damage to the nervous system from TBI or SAH often leads to a long and complicated care time and an extensive rehabilitation phase. The outcome from treatment has been shown to be getting better and better [7, 11]. More patients recover and the mortality has decreased to about 10%. At the Neuro-intensive Care Unit (NIVA) in Gothenburg, many patients with these kinds of complications are treated in the acute initial time after the onset of the complication. During this phase and in early rehabilitation, the level of sedation is gradually decreased over time. This leads to a significant amount of stress in the patient which can be observed in the form of hyperactivity and confused states. In an effort to improve the situation for these patients, NIVA is participating in a research project concerned with investigating the causes and effects of stress in these situations. It is conjectured that the level of stress in the patients has influence on the final outcome of the treatment. In investigating this relation, the level of stress in the patient needs to be estimated and monitored during the care time. Various indicators and estimates for this purpose are discussed in the following section.

The subject of this report is to account for our effort to develop such an estimate, based on recorded biomedical signals. More specifically the estimate is determined by analysis mainly of ECG signals. In addition, we aim to respond to an often expressed request for easily browsable, lucid presentations of information on patients. Our project includes the development of software for visualisation of these kinds of signals, intended to aid physi-

cians in working with post-analysis of treatment progressions. This goal also implies the development of a signal processing software foundation for the visualisation tool to stand on.

2

Technical background

There are several ways of investigating the stress level of a patient. One way is simply to make clinical observations. However, it can be very difficult to observe the stress level of a patient in deep sedation and/or unconsciousness (which is usually the case during the first weeks of treatment at NIVA). Other methods measures chemical substances in the body, for example cortisol in the saliva [1]. A third method, which we are going to focus on in this thesis, studies the heart rate and especially its variations, which have been known for a long time to have a connection to stress [8]. A non-stressed person usually has a higher variation in heart rate, because the self-regulatory system in the body changes the heart rate to adapt to external and/or internal factors [5, 2]. For a stressed person, the self-regulatory system has a lower possibility to change the heart-rate because it is usually already at a high level [8]. In this chapter, a brief background on measuring heart rate and its relation to stress, is presented.

2.1 ECG and heart rate analysis

Electrocardiography (ECG) is a method often employed when studying the heart. It is a non-invasive method that measures the electrical activity of the heart via electrodes on the body. The resulting graph can be used to study the heart rate and also to look for weaknesses in different parts of the heart muscle (see figure 2.1) [10].

A common way to study the heart rate is to measure the time interval between two consecutive heart beats. This is usually carried out by measuring

the distance between R-peaks in the ECG (see figure 2.1). These intervals are called the RR-intervals. When investigating the stress level of a patient the variation of the RR-intervals, known as the Heart Rate Variability (HRV), is especially interesting. A low HRV in combination with a high heart rate is a well known indicator of (acute) stress [8, 3, 1].

However, there are other explanations of a low HRV than mental stress. It is possible that medication causes the stress or internal factors in the body unrelated to mental stress. One way to investigate if a low HRV is caused by stress is to look at the level of correlation between the variations and the respiratory cycle. Normally, the respiration has a strong influence on the heart rate but under stress this coupling usually decreases [8, 4] .

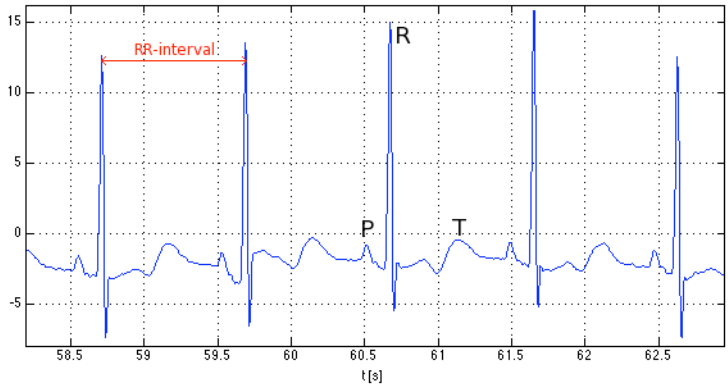

Figure 2.1
ECG illustrating the heart beats with its P,R,T-complexes. The red line indicates one RR-interval.

3

Methodology

When taking on this project we had no prior knowledge on the medical aspects involved, nor did we have any insight about what work had been done in the same area before. To try to satisfy both these needs we initially did a literature study spanning over medical, signal analysis and computer science related papers. During the same period we also had discussions with NIVA personnel, where we got some good input on the task and its relation to their work. We also attended a lecture session on the topic of stress, connected to the research project mentioned in 1.

As we began to get a more clear picture of the problem and had gathered a collection of ideas on how to proceed, we started constructing prototypes. The Matlab environment suited our needs well in this phase as it enabled us to quickly test and tweak our ideas for algorithms.

In parallel with the prototyping we started building a framework for our software in Visual Studio. We employed an iterative development model for the implementation, and ended up with a total of three iterations, excluding the Matlab prototyping. For keeping track of changes during each iteration we set up an SVN server to hold our code base.

Our different fields of knowledge from signal analysis and computer science made the division of labour natural in many cases. Still, we had quite loose divisions on some areas, and we had much collaboration and frequent discussions on the work throughout the entire project.

4

Signal processing algorithms

The main goal of the signal processing is to extract information about the HRV and to investigate its correlation with the respiratory frequency. As mentioned in section 2.1, an RR-interval is found by measuring the distance between two consecutive heart beats, or put more precisely, between two R-peaks (see figure 2.1). Therefore it was essential to design a reliable peak detector to study the heart rate and its variations. Algorithms for analysis of the RR-intervals were developed, extracting both statistical information and spectral information for examining the correlation with the respiration. The main parts of the signal processing are illustrated in figure 4.1. The different parts will be described in more detail in the following sections. Then chapter ends with a discussion on problems caused by outliers in the RR-interval signal and how to reduce these effects. The algorithms are finally tested on data from patients at NIVA and the outcomes are presented in chapter 6.

4.1 Peak detection

In the design of an automatic peak detector one has to consider that it needs to be able to adapt to changes in amplitude or appearance in the ECG of a patient. It should also be applicable to different patients and different recordings. Our algorithm is basically based on threshold search, where the peaks are found by searching after intervals that have larger values than a certain threshold (see figure 2.1). In order to facilitate the choice of threshold a band pass filter is applied to the ECG before threshold-searching. Then, to find the exact maximum of the peaks, a quadratic function is fitted to the

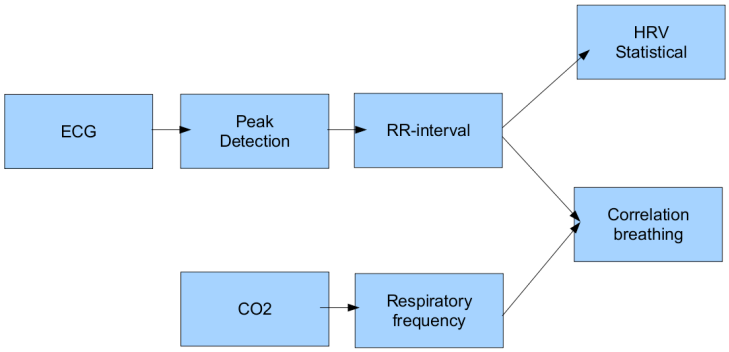

Figure 4.1
Block diagram of the signal processing parts.

peaks, and the maximum is decided analytically. Sometimes the P and T hills are extremely large and will be detected as an R-peak. To take care of this problem we use the statistics of the "peakiness" of the detected peaks to sort out deviating ones. The different steps in the peak detection algorithm are described in more detail below.

Step 1: Filtering

Before threshold search, the ECG is filtered through a band pass filter. This is done in order to have more distinct peaks at the same level, see figure 4.2. The filtering is implemented as an FIR filter of order 61 with a lower cut-off frequency of 4 Hz and a higher of 120 Hz.

Step 2: Threshold search

The beginning and end of peaks are found by sorting out all values which are above a threshold (see figure 4.3). Since the variations in amplitude can be extremely different from one ECG to another it is important that the choice of threshold value is based on the ECG. We implement two ways of

constructing the threshold: one that is based on the mean and one that is based on the maximum point. The user is left to decide which one should be used on a recording of ECG depending on its characteristics.

Step 3: Quadratic curve fitting

The easiest way to find the maximum of a peak is to just pick the sample that has the highest value in a peak. This would, however, put a restriction on the resolution of values due to the discrete sampling of 300 Hz. The difference from values found in the sampled ECG and the true peak values may seem too small to bother about. However, because the RR-interval variations are so small it is actually important to improve this result. Since the peaks are well fit by quadratic function, the precision may be improved by fitting the function to each peak to find the exact maximum analytically (see figure 4.3).

Step 4: Discarding false detections

To get rid of falsely detected peaks, we use information received from the curve fitting to sort out deviating ones, see figure 4.3. This is carried out by looking at the statistics of the second order coefficient in the quadratic curve fitting function. It is used as an indicator of the "peakiness" of a top. If it is too wide or too narrow it will be marked as a "false" peak and discarded.

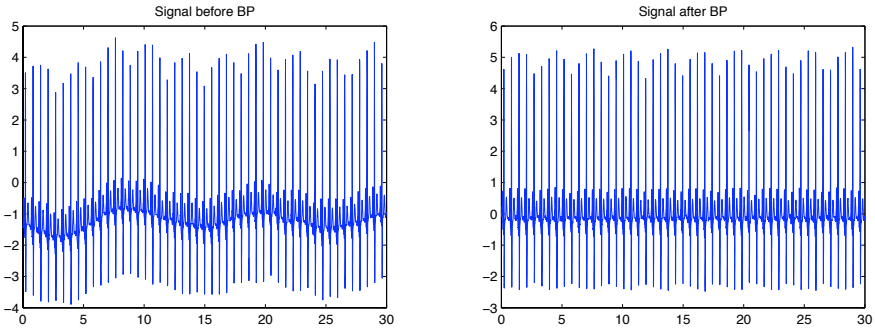

Figure 4.2
1. Band-pass filtering the signal.

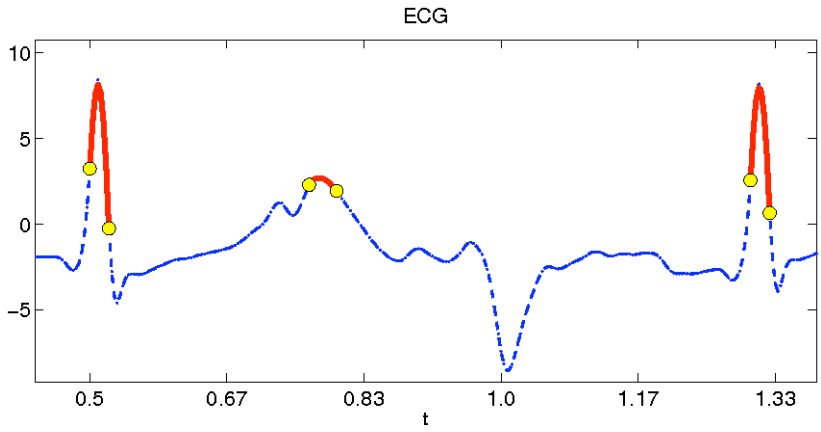

Figure 4.3
2-3. Finding the beginning and end of peaks and fitting a quadratic curve.

4.2 Forming the RR-interval signal

After finding the peaks, the RR-interval signal is formed by taking the time difference between two adjacent peaks. Then, the task is to decide what timing information each value should be connected to. When investigating the distribution of the RR-intervals this is not important to consider. However, for other measures, e.g. when investigating the frequency components of the signal, it is very important to have a correct timing information. We define the RR-interval signal as

$$RR(t_n) = t_{n+1} - t_n, \tag{4.1}$$

where t_n is the time of a peak. See figure 4.4.

4.3 Statistical measures

One way to gain information about the HRV is to investigate the distribution of the RR-intervals. For example, the standard deviation of RR-intervals is commonly used as an indicator of the magnitude of the variations in heart rate (see e.g. [8, 6]). We can also examine the shape of the distribution by investigating its histogram. Is it Gaussian distributed or something completely different?

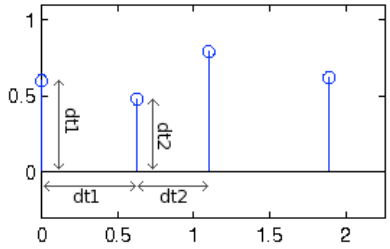

Figure 4.4
Forming the RR-interval signal. $RR(0) = \mathrm{dt1}$, $RR(\mathrm{dt1}) = \mathrm{dt2}$.

4.4 Correlation with breathing cycle

Another way to investigate the RR-interval signal is to find out which frequency components it consists of. Several factors influence the heart beat occurrences in the body. However, here we focus on one cyclic signal that is especially interesting in its relation to the RR-intervals, namely the breathing cycle. As mentioned earlier, the correlation of the RR-interval and the respiratory frequency is an important indicator of a person's stress level. To be able to find the respiratory cyclic period, a signal that is measuring the CO_2 concentration in the exhaled air of the patient is used. To extract the main respiratory frequency we employ a Fast Fourier Transform (FFT) to investigate which component has the largest amplitude in the frequency spectrum.

The algorithms used in this project for examining the correlation between the respiratory frequency and the RR-interval signal can be divided into two different approaches: spectral analysis based, using Fourier methods, and model-based, using Kalman filtering. These two methods will be described in the following sections.

4.4.1 Spectral analysis of the RR-interval signal

FFT is a common method used when investigating the frequency components of a signal. However, it assumes that the time interval between samples of the signal is constant. However, the RR-intervals do not have equidistant sampling and applying FFT directly to the RR-interval does not yield any good results (see figure 4.6). So, to take care of this problem it is neces-

sary to either transform the RR-interval signal into an equidistantly sampled signal or use a transform that contains timing information. Therefore, we consider two cases: one with a regularly sampled RR-interval signal, which is constructed by interpolation and re-sampling of the signal, and one where we use discrete Fourier transform with a time vector.

Regularly sampled

To be able to use FFT to investigate the frequency components of the signal, the irregularly sampled RR-interval signal has to be transformed into a regularly sampled one. This is carried out by creating a continuous signal by linear interpolation between the points and then re-sampling the signal. The question is, how much will the interpolation affect the "true" spectrum of the RR-interval signal? One way to evaluate this is to try other methods that do not require equidistant sampling.

Irregularly sampled

To be able to investigate the spectral content of the signal directly, without having to do the interpolation, we investigated the result of using the Discrete Time Fourier transform with a non-equidistant time vector. Starting with the Discrete Fourier Transform (DFT) and inserting a time vector we get

$$X(f_k) = \sum_{n=0}^{N-1} x(t_n) e^{-j2\pi f_k t_n},\ k = 0, 1, ..., N, \tag{4.2}$$

where $x(t_n)$ is the data at time t_n and N is the number of samples. In the case of a regular time vector the frequencies, normalised to the sampling frequency, are defined as $f_k = k/N$ and the time vector is defined as $t_n = n$, where $n = 0, ..., N-1$. However, when converting this to a case where the time vector is non-equidistant problems arise. Firstly, it is unclear what the sampling frequency is. It seems reasonable to form the sampling frequency based on the mean sample interval, therefore we define

$$f_s = 1/\bar{T}_s, \tag{4.3}$$

where f_s is the sampling frequency and $\bar{T}_s$ is the mean time interval between two samples. Secondly, we know that for a regular DFT the following symmetry property holds:

$$e^{(-j2\pi(-f_k)n)} = e^{(-j2\pi f_{N-k} n)}, \tag{4.4}$$

since $n = 0, 1, ..., N-1$. This property may be used to replace the negative frequencies by positive ones. However, in the non-equidistant case, this does not generally hold:

$$e^{(-j2\pi(-f_k)(n+\epsilon))} \neq e^{(-j2\pi f_{N-k}(n+\epsilon))}, \tag{4.5}$$

unless the deviation $\epsilon \in \mathbb{Z}$. The negative frequencies are essential to be able to form a real valued signal of complex sinusoids (to make the imaginary components cancel out). Therefore, instead of letting the f_k range from 0 to f_s, we let them range from $-fs/2$ to $fs/2$.

To investigate if the DFT algorithm and the FFT yield different results, we run the algorithms on data from three different patients with very different RR-interval signals, see the RR-intervals in figure 4.5. The resulting spectra from the two algorithms have a similar appearance, as may be seen in figure 4.6. Since Measurement Studio contains built-in FFT functions, for simplicity, we decided to use the FFT method for this purpose in our software.

4.4.2 Model-based approach

Another way to solve the problem is to use a model-based approach. Here, a linear Kalman filter is used for this purpose. We postulate the following model, describing the relationship between the RR-interval and the respiratory frequency:

$$RR = \overline{RR} + A\sin(2\pi f_{\text{resp}} t + \phi) + \text{noise}, \tag{4.6}$$

where $\overline{RR}$ is the mean RR-interval and A is the amplitude of the sinusoid of the respiratory frequency, f_{resp}, with a certain phase, ϕ. The noise part contains all the remaining parts of the signal. The unknown parameters are A and ϕ. Although we are only interested in the amplitude, A, it is also necessary to estimate the phase to be able to get an estimate of A. However, this model is not linear since the parameter ϕ has a non-linear relationship with RR. Therefore, the equation is rewritten as

$$RR = \overline{RR} + \alpha\sin(2\pi f_{\text{resp}} t) + \beta\cos(2\pi f_{\text{resp}} t) + \text{noise}, \tag{4.7}$$

where the sinusoid with a phase is converted into one sine and one cosine of the given frequency. Now, the unknown parameters are α and β and the

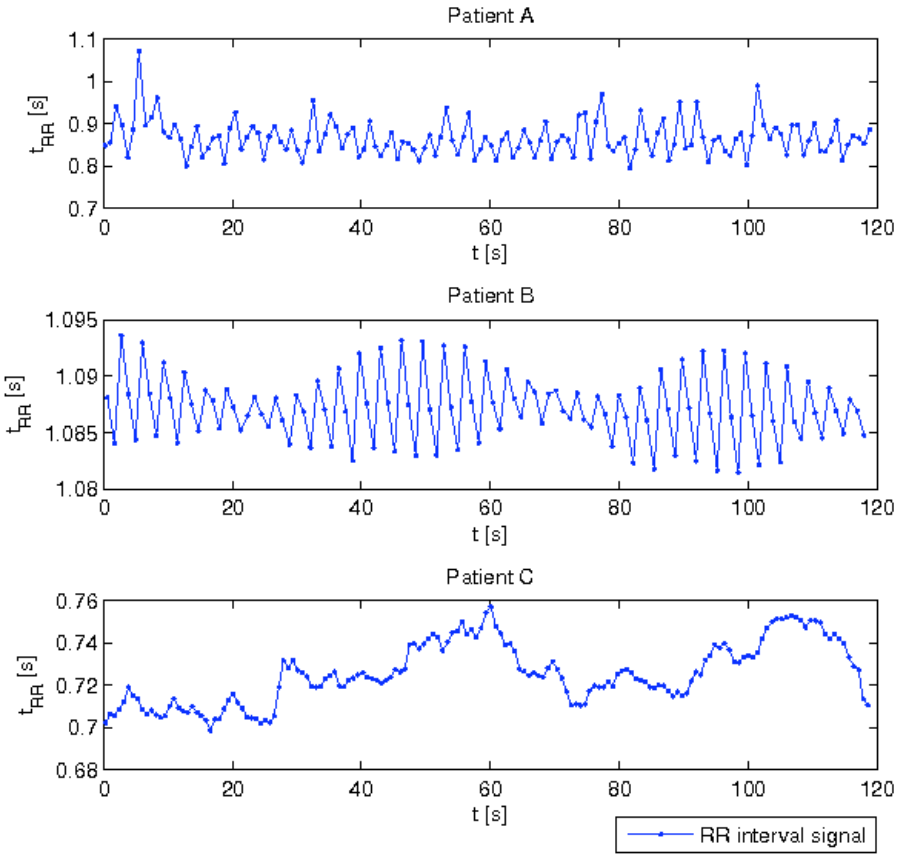

Figure 4.5
The RR-intervals of three different patients.

amplitude is $\sqrt{\alpha^2 + \beta^2}$.

Letting $x = [\alpha, \beta]^T$, the resulting Kalman equations will be

$$x_{k+1} = Ax_k + w_k, \tag{4.8}$$

$$y_k = Cx_k + v_k, \tag{4.9}$$

where y_k is the measured RR-intervals and

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad C = \begin{bmatrix} \sin(2\pi t_k f_{\text{resp}}) \\ \cos(2\pi t_k f_{\text{resp}}) \end{bmatrix}^T \tag{4.10}$$

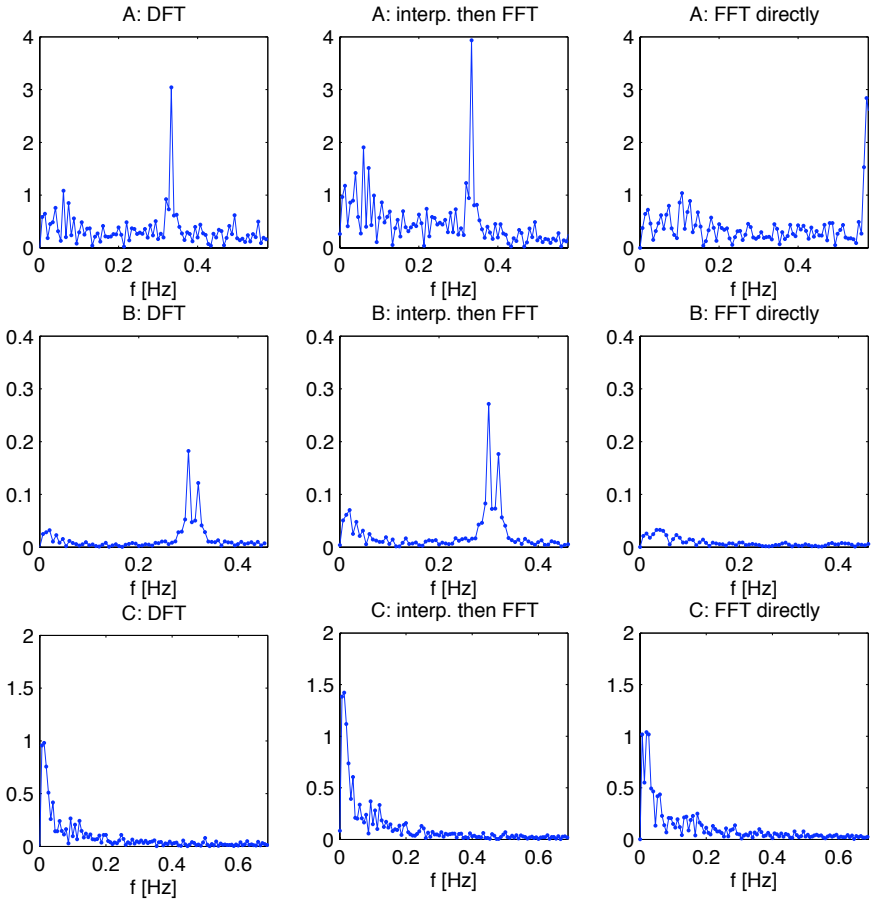

Figure 4.6
The results of applying the different spectral methods to the RR-intervals of three different patients: A, B and C. The left hand column contain the spectra from the DFT algorithm, the middle column contain the spectra from performing FFT on a linearly interpolated and re-sampled RR-interval signal, the right hand column contains the spectra obtained when FFT is applied directly on the RR-interval signals.

and w_k and v_k are measurement and model noise, respectively. The initial values of α and β were set to zero and the initial value for the covariance

matrix was set to the identity matrix.

Results and problems with Kalman

Because the Kalman filter is derived under the assumption of Gaussian noise, it may not give optimal performance applied directly to the data in this case. Since the patients in NIVA are mostly mechanically ventilated we know in what range the respiratory frequency lies (between 0.2 to 0.3 Hz). In those cases, it is possible to filter out the non-interesting frequencies. This is taken care of by band-pass filtering the signal around this area before applying the Kalman filter. However, to be able to band-pass it we first need to interpolate and re-sample to make the samples equidistant. We decided to include both methods in our software to compare the results. The resulting amplitude is estimated by running the Kalman filter and then taking the mean of the amplitude estimates.

4.5 Removing outlying RR-intervals

Sometimes the ECG deviates so much that the peak detector misses or detects an extra peak, and sometimes the heart simply has a large deviating difference from one peak to another. This will occasionally result in RR-intervals that are extremely small or large. Now, even if these extreme values happen rarely, we have noticed that they have a very high impact on the results. Because these values are so much larger than the small variations in the heart rate (they are of the order of magnitude twice or half a regular RR-interval), they will have a large impact on the spectrum of the signal and in the forming of the mean in the Kalman filtering. Therefore, these values are removed. This is carried out by removing those RR-intervals which have a value that is outside a certain factor times the standard deviation from the mean value or the median. The factor is a variable that should be chosen considering the variations in the recording one wants to study and is chosen by the user.

5

Software implementation

This chapter describes the implementation of the methods discussed above. It also describes design choices and their implementation concerning the underlying framework. The first section is an exposition of the framework, its main concepts and components. Next is a section on file format issues and how they were resolved. The third section discusses the time and memory performance of the software and makes some remarks on how these might be improved. In the last section we describe the visualisation interface.

5.1 Extendibility

From the description of our project and discussions thereof it became obvious that creating software specifically for solving the short list of analysis problems that we were presented with might not be of much value in a larger time perspective, but rather lead to deter potential future developers from improving and extending our results. We decided to include in our goals the idea that our software should serve as an easily extendible platform for solving other signals processing tasks as well. The object-oriented approach of modularity and hiding of complexity would serve not only the goal of an extendible program, but would also help us to reason about our own analysis implementations. Striving to reduce the workload of implementing our signal analysis algorithms and visualisations we chose to base our software on the signal processing library Measurement Studio by National Instruments. This proprietary software was made available to us by the Signals and Systems institution, through our supervisor. This led us to use Microsoft's Visual

Studio 2008 as our development environment and Visual C# as our implementation language. The next two subsections describes the design of the framework and are followed by an account of the function of specific parts.

5.1.1 Signals

The main concept in our software is the signal objects. A signal object represents an interval in time from which waveforms over subintervals can be requested. A waveform in turn represents a sample collection consisting of zero or more time-stamped values. Signal objects of different types are implemented and instantiated to represent different kinds of processing tasks. The calculations for producing a requested waveform take place in the signal object. They are parametrised by the requested time interval together with an implementation-specific collection of variables that are set for each signal instance. A signal may also be dependent on one or more other signal instances for computing its waveform. We describe these relations by letting each signal object instance keep an ordered list of sub-signals on which its calculations depend.

5.1.2 Algorithm configuration

The dependency relation on signals allows for complex processing algorithms to be represented as graphs with signal instances as nodes and directed edges describing the data routing. A calculation is initiated in some node, for example by the user requesting a waveform plot from it over some time interval. The request is propagated down the dependency edges until the paths reach signals that have no dependencies and are capable of generating data. As initiated calculations are completed, requests are answered with data flowing upwards until the original initiating calculation delivers its resulting waveform to the user.

We have chosen a method of encoding these algorithm graphs using XML documents. An XML Scheme defines the structure of these configuration files and what parameters are required for the different signal types. The following excerpt from a configuration file gives an idea of the format and gives a peek on the various types of signal implementations.

```
<signals>
  <BinaryFileSignal name="ECG"
```

```
                    fileName="ECG1.dat" />

  <BinaryFileSignal name="CO2"
                    fileName="CO2.dat" />

  <TimeIntervalSignal name="RR-intervals">
    <subSignals>
      <PeakDetectSignal name="ECG_peaks"
                        windowSize="60">
        <subSignals>
          <BandPassSignal name="ECG_bp"
                          cutoffLow="4"
                          cutoffHigh="120">
            <subSignals>
             <signalReference referencedSignal="ECG" />
            </subSignals>
          </BandPassSignal>
        </subSignals>
      </PeakDetectSignal>
    </subSignals>
  </TimeIntervalSignal>

  <CorrelationByKalmanSignal name="HRVtoResp_kalman"
                             Qvalue="1E-1"
                             Rvalue="1E-2">
    <subSignals>
      <StrongestFrequencySignal name="Resp-freq"
                                windowSize="120"
                                minFreq="0.1"
                                maxFreq="1.0">
        <subSignals>
          <signalReference referencedSignal="CO2" />
        </subSignals>
      </StrongestFrequencySignal>
      <signalReference referencedSignal="RR-intervals" />
    </subSignals>
  </CorrelationByKalmanSignal>
</signals>
```

This setup corresponds to the routing graph in figure 5.1.
Allowed child tags under *signals* and *subSignals* tags correspond to the names of signal implementations. To avoid creating multiple instances of the same signal, the *signalReference* tag is used to add a dependency relation pointing to an already defined signal.

When a configuration file is loaded it is first checked for compliance with the XML Schema Definition. If it is found to be correct, the specified signal types are instantiated and their lists of subsignals are populated recursively. To ensure easy extendability, the names of the XML tags are the actual C# class names of the signals they represent. This allows instances of implemented signal types to be created by a general mechanism, using the reflection capabilities of C#.

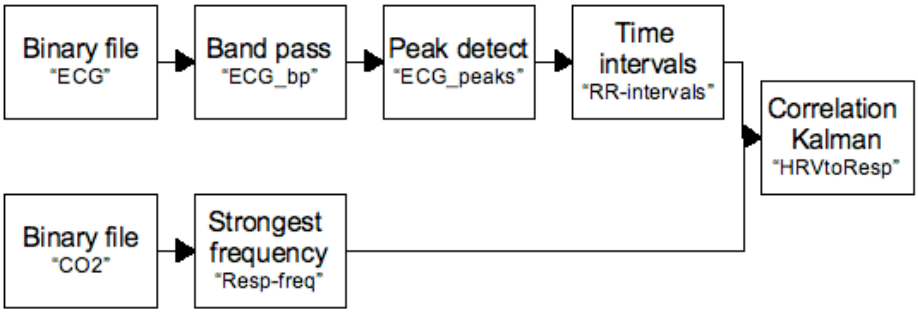

Figure 5.1
A routing graph.

5.1.3 Implemented signal types

Below is a listing of the type-tree containing all implemented signals. Following that is a series of short descriptions of the various signal types and their function.

```
Signal
  RegularlySampledSignal
    AsciiFileSignal
    BandPassSignal
    BinaryFileSignal
    CorrelationByFFTSignal
    CorrelationByKalmanSignal
    DownsampledSignal
    InterpolatedSignal
```

```
    MovingMeanSignal
    MultiscaleSignal
    StandardDeviationSignal
    StrongestFrequencySignal

  IrregularlySampledSignal
    GetOutliersSignal
    InverseSignal
    IrregularlySampledBinaryFileSignal
    OutliersDetectorSignal
    OutliersDetectorWindowedSignal
    PeakDetectSignal
    RemoveOutliersSignal
    TimeIntervalCorrectionSignal
    TimeIntervalSignal
```

RegularlySampledSignal is the abstract base class of signals containing values at regular intervals in time.

AsciiFileSignal represents values read from an ASCII file. The data is expected to be formatted into tab-separated columns where each column contains the values for one signal. The rows are assumed to represent samples taken at regular time intervals at a sampling frequency specified in the file header. The date and time of the data collection is also assumed to be found in the file header. This signal does not accept any subsignals.

BinaryFileSignal represents values read from a file in our binary format, more thoroughly covered in 5.2.2.

BandPassSignal represents a FIR bandpass filter with the specified passband. This signal accepts a single regularly sampled subsignal.

DownsampledSignal takes a regularly sampled signal as input and reduces the sampling rate of it by a predefined integer factor. The downsampling is achieved by simply removing appropriate sequences of samples and no further filtering is incorporated in this algorithm. Letting y denote a *DownsampledSignal*, then its value at time $t_i = i * T_y = i * k * T_x$, for integer sample indexes i, is

$$y(t_i) = x(k * i * T_x),$$

where i is the index of a sample in y, T_y is the sample interval of y, T_x is the sample interval of x and k is the integer downsampling factor.

InterpolatedSignal is capable of upsampling a signal by piecewise linear interpolation of the samples from its subsignal. It is parametrised by a requested output frequency at which the piecewise linear function is sampled. If y is the interpolated signal, its value at time t_i, for integer sample indexes i, is defined by

$$y(t_i) = \begin{cases} x(t_i) & \text{, if x has a value at } t_i \\ x(t_j) + \dfrac{(t_i - t_j)(x(t_{j+1}) - x(t_j))}{t_{j+1} - t_j} & \text{, otherwise} \end{cases}$$

where x denotes the subsignal, t_j is the time closest below t_i where x has a value, and t_{j+1} is the time closest above t_i where x has a value.

MovingMeanSignal represents the convolution of its regularly sampled subsignal with a square function of height 1 and a predefined width, divided by the width of the square. The output is shifted and cut to give the result of a mean window that is timewise centered around each sample. If y is a *MovingMeanSignal* and i is a sample index, the signal values are described by

$$y[i] = \frac{1}{w} \sum_{j=i-w/2}^{i+w/2} x[i],$$

where x denotes the subsignal, and w is the width of the mean window.

MultiscaleSignal represents a multiplexer over a collection of signals that occupy the same segment in time. Based on an extra parameter, an incoming waveform request is relayed to one of the signals in the collection. The extra parameter is a target sample count for the resulting waveform and the *MultiscaleSignal* selects the subsignal that, for the specific request, gives a waveform with a number of samples closest to the target count.

StandardDeviationSignal calculates the standard deviation of the values from its input signal over consecutive segments of time. Letting y be the

StandardDeviationSignal and x be its subsignal, the values are described by

$$y(t_i) = \sqrt{\sum_{t_j \in W} \frac{(x(t_j) - \mu)^2}{|W| - 1}},$$

where $\mu = \sum_{t_j \in W} \frac{x(t_j)}{|W|}$, $W = \{t \mid t_i \leq t \leq t_i + w\}$ and w is the set length of the time segments.

StrongestFrequencySignal is a signal whose values are calculated as the frequency of the strongest spectral component in consecutive time segments of its subsignal.

CorrelationByFFTSignal takes two regularly sampled subsignals. Its values are calculated as the energy of a selected component of the power spectrum of its second subsignal. The frequency of the component that is considered is given by the value of the first subsignal.

CorrelationByKalmanSignal has a basic semantic interpretation that is similar to that of the *CorrelationByFFTSignal*, except that the strength of the component is determined by a Kalman filtering method, covered in 4.4.2.

IrregularlySampledSignal is the abstract base class of signals containing values at arbitrary points in time.

IrregularlySampledBinaryFileSignal represents values read from a file in a variation of the binary format used for *BinaryFileSignal*, covered in 5.2.3.

PeakDetectSignal represents a sequence of peak values taken from its subsignal using an algorithm described in 4.1.

TimeIntervalSignal is a signal whose values are the time intervals between consecutive samples of its subsignal. Letting y be the *TimeIntervalSignal* and x be its subsignal, the values are described by

$$y(t_i) = t_{i+1} - t_i,$$

where t_i and t_{i+1} are the times of two consecutive samples in x.

OutliersDetectorSignal returns, using an extension to the standard waveform data flow, a list of exceptional points along with the unchanged waveform of its subsignal. A sample is marked as exceptional if it is found to be an outlier compared to the other samples in the waveform, using configurable statistical measures. The predicate e_i denoting whether the sample with index i, from the subsignal x, is an outlier can be determined by two methods. Using mean

$$e_i = (x[i] < \bar{z} - c\sigma_z) \vee (x[i] > \bar{z} + c\sigma_z),$$

or using median

$$e_i = (x[i] < \tilde{z} - c\tilde{z}) \vee (x[i] > \tilde{z} + c\tilde{z}),$$

where c is a parameter, $z = \{x_i \mid v_{min} < x_i < v_{max}\}$, v_{min} and v_{max} are parameters, σ_z is the standard deviation of z over the requested interval, and $\bar{z}$ and $\tilde{z}$ represents mean and median over the interval, respectively.

OutliersDetectorWindowedSignal works just like *OutliersDetectorSignal* except that all statistics are taken on consecutive intervals of configurable length.

GetOutliersSignal requests a list of exceptional points along with the waveform data from its sub-signal, and returns only the values that are marked in that list. Letting y and x stand for the *GetOutliersSignal* and its subsignal respectively, and using e_i as the exceptional predicate for sample i as in *OutliersDetectorSignal*, then

$$y = \{x_i \mid e_i\}.$$

RemoveOutliersSignal requests a list of exceptional points along with the waveform data from its sub-signal, and filter out the values that are marked in that list. Letting y and x stand for the *RemoveOutliersSignal* and its sub-signal respectively, and using e_i as the exceptional predicate for sample i then

$$y = \{x_i \mid \neg e_i\}.$$

TimeIntervalCorrectionSignal also requests a list of exceptional point along with the waveform data from its sub-signal. It then tries to correct the exceptional intervals by replacing them with a fitting number of standard

intervals. This is done by first grouping consecutive outliers and summing the values of samples in each group. This amount is then divided by the median value of the signal and rounded to the closest integer. The group of samples is then replaced by that number of equidistant samples of equal value. The time-value pairs forming the inserted samples for each group G_j of consecutive outliers, can be written as

$$(t_0,\ y_i) = (t_0,\ \frac{s_j}{n_j})$$
$$(t_i,\ y_i) = (t_{i-1} + \frac{s_j}{n_j},\ \frac{s_j}{n_j})\ \text{, for all } i \in [1, n_j]$$

where t_0 is the original time of the first sample in G_j, $n_j = \left[\frac{s_j}{\tilde{x}}\right]$, the integer number of inserted samples,$s_j = \sum_{k \in G_j} x_k$, x is the sub-signal, and $\tilde{x}$ is the median of x over the requested interval.

InverseSignal computes the inverse of each value from its sub-signal. Having y as the *InverseSignal* and x as its sub-signal, then

$$y(t_i) = x(t_i)^{-1}$$

5.2 File formats

Measurements from the bedside patient monitors at NIVA can be recorded over the hospitals local network using a piece of software from Datex-Ohmeda, called S/5 Collect. The recorded data is saved in an ASCII format allowing for transfer to other tools for offline analysis. Easy to read as this text format is, it does have some serious downsides.

The amount of data that these recordings can, and in this context, often do contain, is quite large. As an example of this, consider that the instruments at NIVA supply sample data at a rate of $f_s = 300$ Hz and that it is common to make continuous recordings of several such signals at once, spanning many days. Using some common values for the number of simultaneous signals, $n_s = 7$, and the length of the recording, $n_d = 5$ days, one reaches the following total number of samples recorded: $n_s * f_s * n_d * 60 * 60 * 24 = 907,200,000$.

The samples are written in text format as floating point numbers of varying length. Each line in the text format contains one sample from each of

the recorded signals, all taken at the same point in time. The lines do not contain any absolute timestamps or explicit indices. Therefore, the time at which a sample was taken must be inferred from: the known start time of the recording, the known sample rate and the number of lines preceding a given sample. This fact, together with the varying length text representations of the sample values, makes it impossible to perform random access on the sequence of samples in a file, containing n sample occasions, with time complexity any less than $O(n)$. Using text representations of numbers also make the actual action of reading a value from file, into a format on which to perform arithmetics, unnecessarily time consuming. Although the implementation details of this functionality in the .NET framework are not known to us, the action is so common in our program that it introduces a notable performance reduction.

Taking all of the above into account, a new file format was defined that uses direct binary representations of values, and where the byte length of values is fixed. A file in this format stores the data of a single signal, over some time interval. The format was branched into two, as the need for new features arose. Both formats specify a header section at the beginning of the file, containing general information, followed by a sequence of sample representations.

5.2.1 Saving signals

A utility was implemented for importing signal data from the ASCII recording format to the binary signal format. The utility lets the user select signals from a recording, then converts and saves them in a folder of choice, and constructs a basic configuration file containing the newly converted signals.

There is also a utility for writing derived signals to file. The functionality is polymorphic over signal objects and it is therefore possible to route the output of any node in a computation graph to a file on disc.

5.2.2 Equidistantly timed samples

For signals sampled at regular time intervals, timing data is not stored for each sample. The following table specifies the data stored in each word of a file in this format.

32-bit word	Description	Unit	Type	Data
0	File format specifier	-	int	0
1	Sample data start	words	int	6
2	Sample data end	words	int	var.
3-4	Acquisition time	s	double	var.
5	Sample frequency	Hz	float	var.
6	Sample value 0	-	float	var.
7	Sample value 1	-	float	var.
8	Sample value 2	-	float	var.
9	Sample value 3	-	float	var.
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.

5.2.3 Non-equidistantly timed samples

In this format, each sample is stored as a pair of a time-stamp and a value. Here information is stored as doubles for increased precision. The following table specifies the data stored in each word of a file in this format.

32-bit word	Description	Unit	Type	Data
0	File format specifier	-	int	1
1	Sample data start	words	int	6
2	Sample data end	words	int	var.
3-4	Acquisition time	s	double	var.
5	Sample frequency	Hz	float	var.
6-7	Sample time 0	s	double	var.
8-9	Sample value 0	-	double	var.
10-11	Sample time 1	s	double	var.
12-13	Sample value 1	-	double	var.
14-15	Sample time 2	s	double	var.
16-17	Sample value 2	-	double	var.
18-19	Sample time 3	s	double	var.
20-21	Sample value 3	-	double	var.
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.

5.3 Performance tradeoffs

The size of the typical underlying data sets in the context of the analysis problems in this project makes the resource usage of the implementation a pressing issue. In many cases time complexity is the attribute that is first considered for optimisation, but here, memory resource usage was considered to be of higher priority.

In general, the choice to make a computation conservative in memory usage leads to drawbacks in time complexity, and so also in this case. The actual algorithm implementations are not affected to any large extent on the lower levels, but the compromise is more significant on a higher level. In particular, the data management in the computation graphs, mentioned in 5.1.2, is subject to performance tradeoff. For starters the modular approach that we have decided to use, and the signal object abstraction that it is realised by, leaves the sub-algorithms isolated from each other, except through a small interface. This means that one sub-algorithm cannot interfere with the progression of another, except through the predefined data channels. While gain is made in reducing the up front semantical complexity of the program, some limitations are set on optimisability.

Signal objects in the computation graphs operate on a principle of 'calculate, respond and forget'. As a request for a waveform over a time interval is given to a signal, it collects the data needed, performs its calculation, returns the result as response to the request, and finally releases all its allocated memory to the .NET Garbage Collector. Only the data in the response is kept in memory, as it is now referenced to by the requester. This behaviour means that at any step throughout the progression of a complete calculation, there is a maximum of two levels of waveform data kept in memory at once. On the downside it means that some identical calculations are performed more than once, depending on the structure of the computation graph. It would therefore be advantageous to employ some mechanism for dynamically cacheing data, with respect to its access rate and the available memory resources. This is left as a suggestion for further improvement of the system, as it could not be included into the time-frame of this project. As a substitute the user can make manual use of the signal file writing utility for achieving the same basic goal.

5.4 Visualisation

We built our visualisation tools by augmenting existing Measurement Studio plotting utilities. The design is based on a number of plot areas to which the user can add and remove plots by drag and drop, and context menu commands. The plot areas allow for free, or axis-wise constrained, panning and zooming, as well as text input specification of the plotted range.

The main concept when designing our means of visualisation was the idea that biomedical signals, of the kind that we are presented with, are interesting to a user on greatly different time scales. As examples of this, consider the study of QRS-complex morphology, where ECG signals are analysed on scales of seconds, compared to the study of trends in intracranial pressure (ICP) where time scales of hours, and possibly days, are of interest [9].

5.4.1 Signal plot items

Different subsets of computed intermediate and top-level signals are probably going to be of interest to different users. We also need a way of specifying the individual appearances for plotted signals. We implemented a layer between the set of signals and the plotting mechanism, where signal plot items are specified. The definitions reside together with the algorithm configuration in the XML files. Each such signal plot item references a signal object and contains a set of attributes, determining its appearance when plotted. Making use of the idea of scale based presentation, attributes in signal plot items specify in what range of scales it is visible. Other attributes specify the plot color, line style, etc. It is possible to define several signal plot items all referencing the same signal, thus letting a plot change its appearance over different scales. As well as being a model for displaying relevant information to the user, this system saves resources through its lazy, compute on demand, design. Below is an excerpt from a configuration file showing the definition of two signal plot items.

```
<SignalPlot signalName="HRVtoResp_fft"
            minRange="3E2" maxRange="1E5"
            lineColorR="155"
            lineColorG="25"
            lineColorB="62"
            lineStyle="Dot"
            yScale="1E6"/>
```

```
<SignalPlot signalName="HRVtoResp_fft_mean"
            minRange="1E3" maxRange="1E7"
            lineColorR="245"
            lineColorG="85"
            lineColorB="122"
            yScale="1E6"/>
```

5.4.2 Auxiliary plotting tools

The program also has spectrum and histogram plotting capabilities. On the signal object level, two interfaces are implemented to enable this functionality for a signal. The system recognises what signals can plotted in this way on loading a configuration. On user request, a spectrum or histogram plot can be drawn. In addition to being browsable in the same way as the main plot utility, the interface lets the user browse along the time axis of the underlying signal while watching the plot continually update.

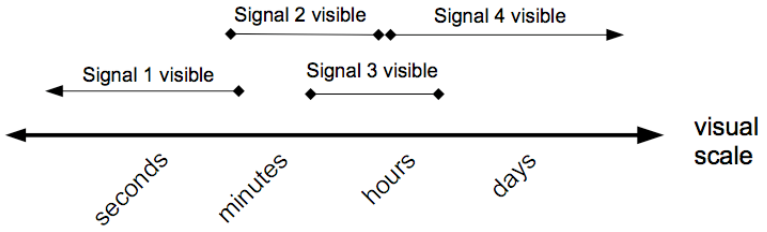

Figure 5.2
A visualisation of the scale-based model for displaying plots.

6

Applications

In this chapter we are going to present results from running the software on data from 4 different patients from NIVA. First we show some plots intended to highlight results from different processing steps and investigating the different signals that were presented in Chapter 4: the standard deviation of HRV, correlation with respiration, both using Kalman and using FFT. The chapter ends with a large scale analysis of the patient data. We chose to analyse the RR-intervals over 2 minutes intervals since the recommended standard is 2-5 minutes for short time analysis of HRV [6]. The plots are scaled and shifted and their magnitudes should be considered as having arbitrary units.

6.1 Detail studies

RR-intervals

Figure 6.1 shows two plots over the same time interval. In the bottom plot we see the ECG signal that the other signals are derived from. It also shows the output from the peak-detecting algorithm, which is performing sub-optimally on this particular segment. It fails to identify a couple of R-peaks, seen to the right. Near the left edge of the view, however, there is a disturbance in the ECG which is correctly discarded. The top view shows the calculated time differences between the peaks (RR-intervals). The effects of the disturbance and the missed peaks are clearly visible as large peaks among the otherwise very small variations. Drawn on top of that line is the same RR-interval

signal after passing through our outliers filter. The vertical axis for the RR-intervals is in seconds, and will be so henceforth, unless otherwise stated.

In figure 6.2 we look at an RR-interval signal on a larger time scale, showing the kind of variation that is present. We also see the outliers filter in effect. By looking at the overall trends in this plot we can infer that the mean heart rate varies between about 65 and 85 bpm over this interval.

Respiration

The respiratory frequency as extracted from recordings of the level of CO_2 in exhaled air, is also available for plotting. In figure 6.3 such a signal is plotted. We can observe a section where the frequency is more or less constant due to completely artificial respiration. To the right the frequency starts fluctuating, probably because the patient starts to take breaths of his/her own.

Stress Level Estimators

In figure 6.4 the time scale has been increased even further. The bottom view shows the RR-interval signal over this time interval. In the top view we can follow the total amount of variability in the HRV by looking at the standard deviation plot. Also in the top view we can observe changes in the measure of correlation to respiration that are independent of the total variability, thus suggesting that the HRV, as expected, "synchronises" differently with respiration at different times.

Examining the cause of the independently varying correlation measure in figure 6.4 we study the RR-interval on a smaller scale at two points in time. Figure 6.5 shows the RR-interval signal at time 11.01 where we have a relatively high variability but low correlation. Indeed this plot looks somewhat random and can be said to have periodicity in only the vaguest sense. At time 07.00, shown in figure 6.6, on the other hand, we see a clear periodic variation. A very crude estimation of the frequency content in this signal can be formed by simply counting the number of peaks in the plot and dividing by the length of the time segment, yielding the frequency $29/(60 + 41)\text{Hz} \approx 0.29\text{Hz}$. This estimate is very close to the respiratory frequency, which is known to be at around 0.3 Hz at this time.

In figure 6.7, a comparison between the different estimators of respiratory correlation is presented. The plots show that the three different methods

yield similar results in general but also that they have individual differences on smaller scales.

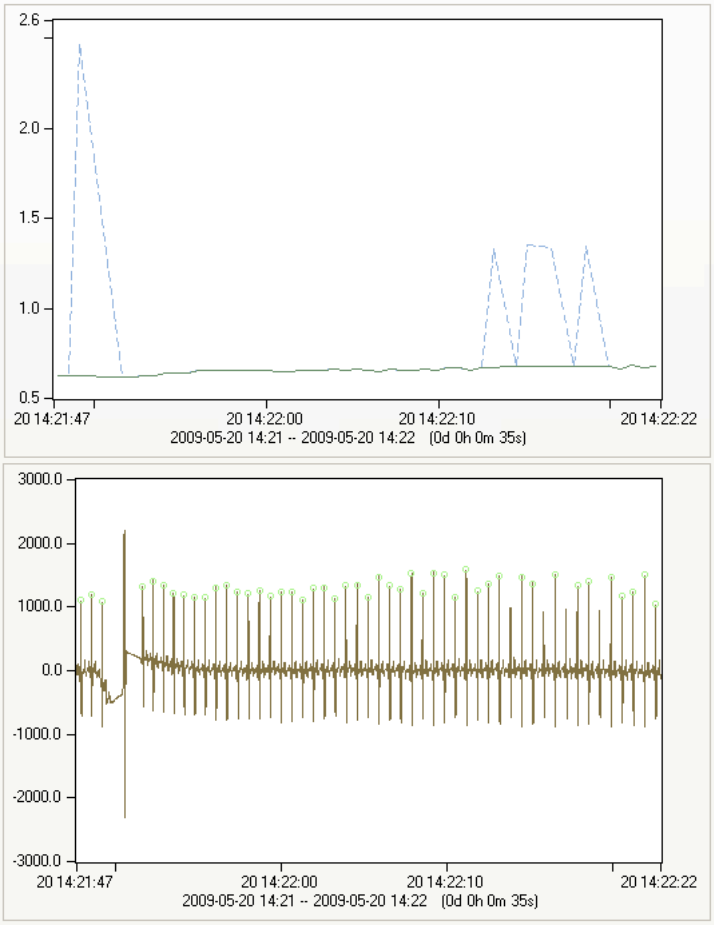

Figure 6.1
Top: RR-interval signal (dashed). Corrected RR-interval signal (solid). Bottom: ECG signal (solid). Detected peaks (circles).

6.2 Large-scale analysis

In figure 6.8 to 6.11 are presented four plots of large scale analysis results from four different patients who were treated at NIVA, and whose biomedical

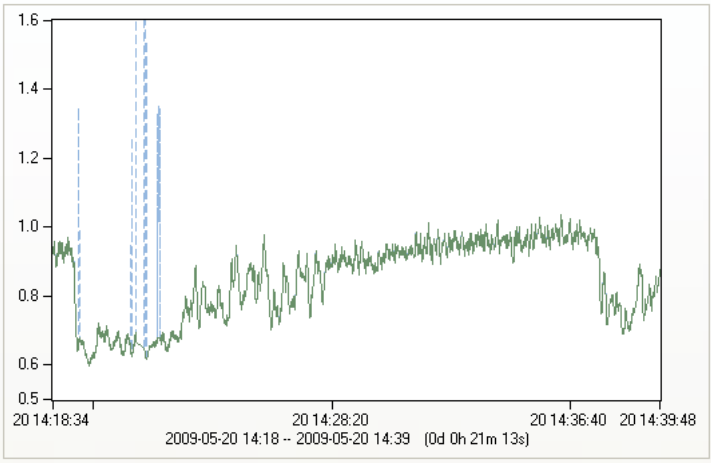

Figure 6.2
RR-interval signal (dashed). Corrected RR-interval signal (solid)

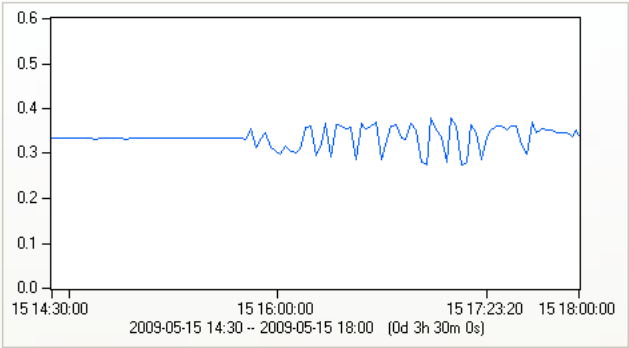

Figure 6.3
Respiratory frequency (solid)

signals were recorded. Please note that the time scales vary greatly between these plots as a result of the lengths of the recordings. Also bear in mind that the vertical scales differ for some plots. The indicators of respiratory correlation used here are Kalman without band-pass filtered input and the FFT method. Patient A (figure 6.8) shows a comparatively low HRV all in all, while patient B has a rather high variability. Patients C and D have comparable levels of HRV while their correlation measures look more differ-

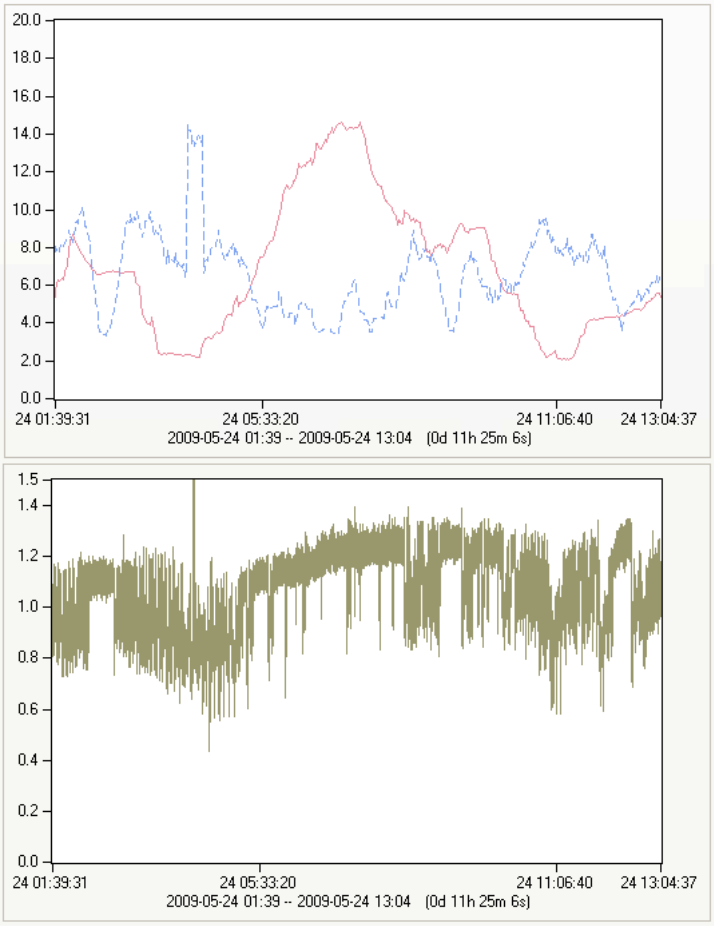

Figure 6.4
Top: Standard deviation of RR-interval signal (dashed). Correlation with respiration using the FFT method (solid).
Bottom: Corrected RR-interval signal (solid).

ent. In most patients we see disturbances in the signals where our filters have not been able to sort out anomalous input. These disturbances correspond to times when the measurement equipment has been disconnected from the patient, etc. In all patients we see transitions from ranges of low correlation to ranges of high correlation and vice versa. The big transitions seem to happen over the course of one or a few hours, with ranges of several hours in between. In discussions with physicians at NIVA we tried to link these

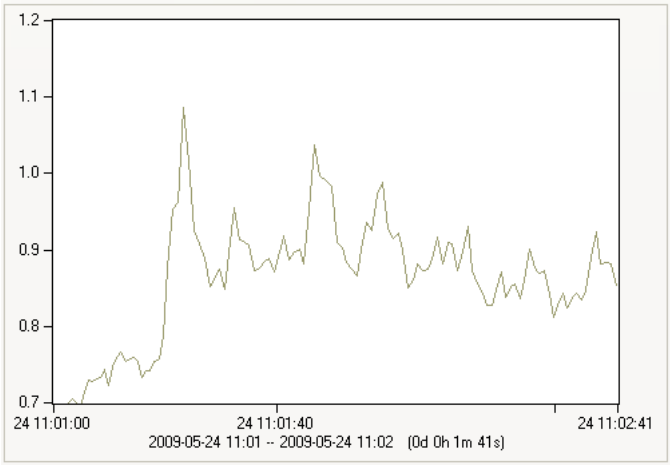

Figure 6.5
RR-interval signal (solid).

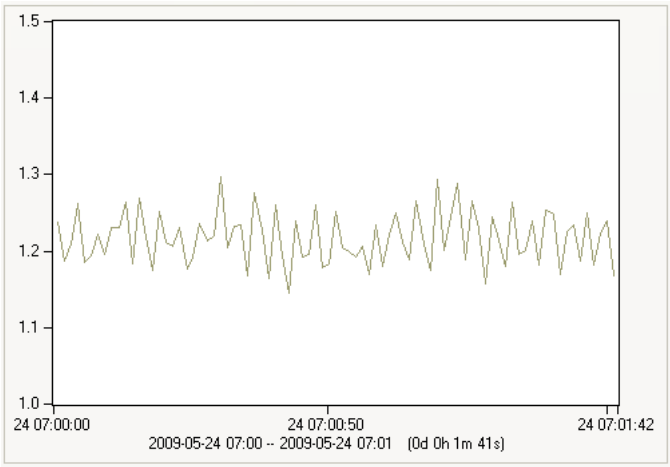

Figure 6.6
RR-interval signal (solid).

variations to data collected in the patients journals. We looked at sedation levels, medications and body temperature readings etc., as well as general clinical observations, but could not find any clear connections.

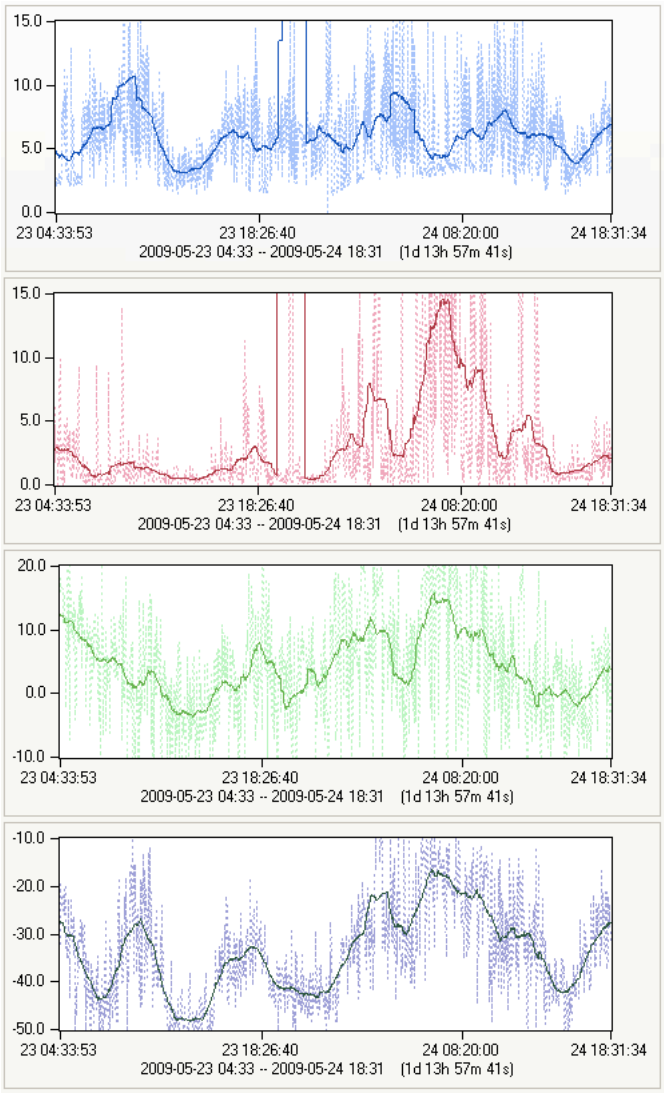

Figure 6.7
1st: Standard deviation of RR-interval signal (solid) and its mean (dashed).
2nd: Correlation to respiration using the FFT method (solid) and its mean (dashed).
3rd: Correlation to respiration using the Kalman method (solid) and its mean (dashed).
4th: Correlation to respiration using the Kalman model with band-passed input (solid) and its mean (dashed).

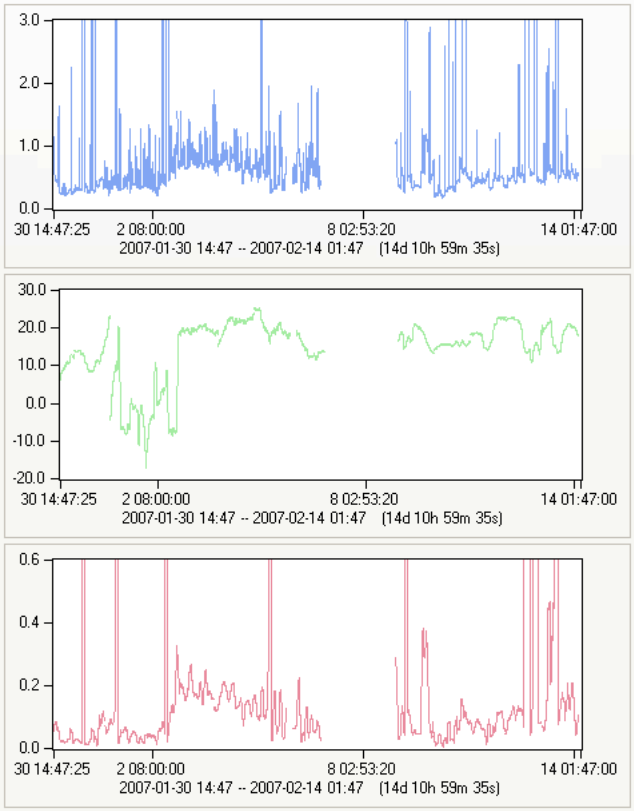

Figure 6.8
Patient A
Top: Standard deviation of RR-interval signal.
Middle: Correlation to respiration using Kalman filtering.
Bottom: Correlation to respiration using FFT.

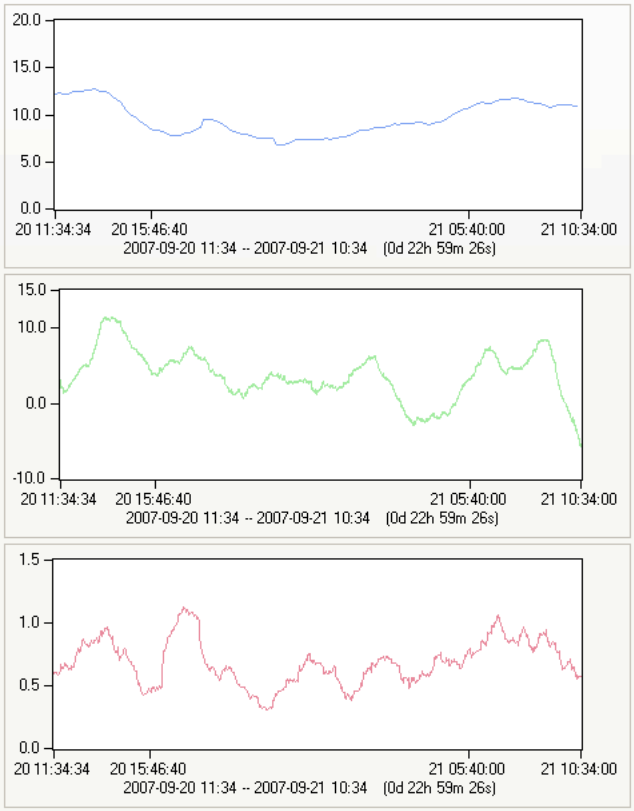

Figure 6.9
Patient B
Top: Standard deviation of RR-interval signal.
Middle: Correlation to respiration using Kalman filtering.
Bottom: Correlation to respiration using FFT.

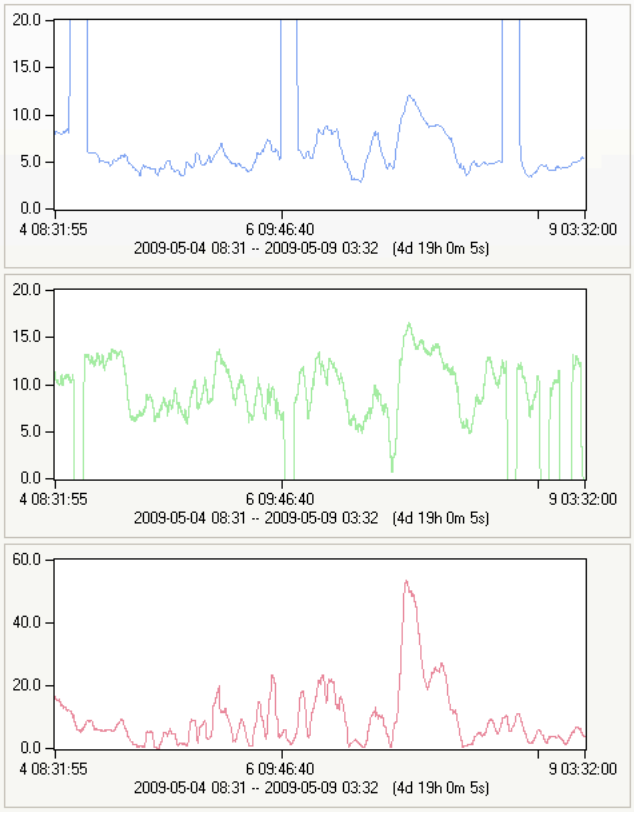

Figure 6.10
Patient C
Top: Standard deviation of RR-interval signal.
Middle: Correlation to respiration using Kalman filtering.
Bottom: Correlation to respiration using FFT.

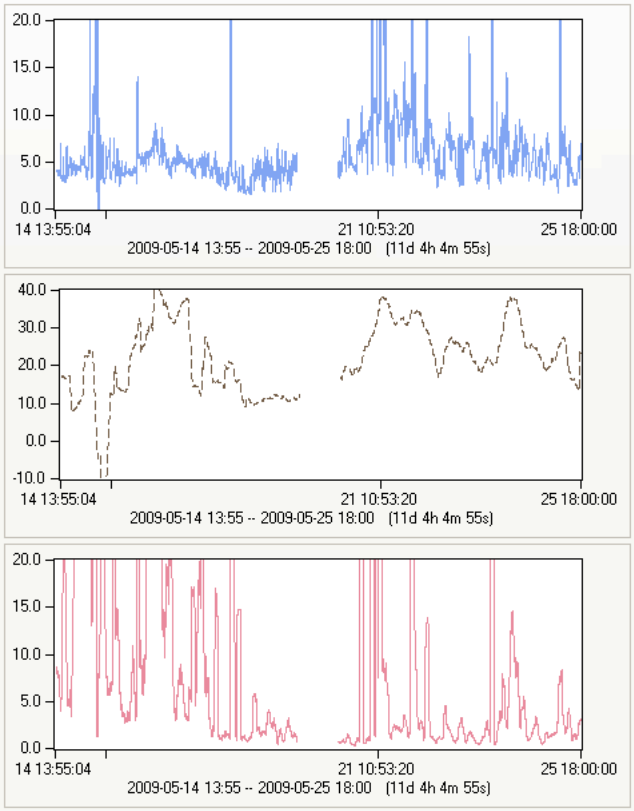

Figure 6.11
Patient D
Top: Standard deviation of RR-interval signal.
Middle: Correlation to respiration using Kalman filtering.
Bottom: Correlation to respiration using FFT.

7

Discussion

As mentioned in section 6.2, we had little success verifying the final results by linking them to other available information on the patients. However, to do this in a satisfactory manner one needs some other estimate of the stress level to compare with. This was not available to us, but we believe that a more thorough investigation will be possible as more and better data is collected in the NIVA stress project. They have just recently started making records directly related to stress, such as measurements of salival cortisol (section 2) and clinical observations. Furthermore, as we know that the human body is an extremely complex system, and particularly since the patients all have severe complications together with mixes of high dosages of medication, it is even unclear what accuracy to expect from our estimates. In this light we suggest that, for reference, our methods should be tested under more controlled circumstances on subjects with much lesser, or no complications and medications.

Concerning our signal processing algorithms, we see that they perform well under good conditions and that they also handle many anomalies with satisfying results. Even so, there are still many cases where the results are not reliable, and where there is room for much improvement. An example of a specific such case is when the ECG electrodes are disconnected for some time in a recording, during which the ECG signal should be discarded completely. As anomalies of varying severity can occur on different levels in the analysis we suggest the implementation of some kind of quality measure associated with signals at different times. Currently the behaviour of the algorithms needs to be tuned to achieve good performance on some input recordings. This is due to big differences in signal characteristics between patients and

between ECG setups. With some effort we believe it should be possible to automatically deduce many of these parameters, thus simplifying usage of the program.

The goal discussed in section 5.1 of avoiding a static software system discouraging from further development has led to a shift of focus. Because of the limited time given to us, we chose to put more energy than initially planned into making the program extendible, and less into the user interface design, etc. In proceeding with the development of the user interface some proper design input should be collected. This could be done, for instance by setting up user tests with people from intended user groups, or by interviews, resulting in use cases. At present some parts of the program are implemented as stand-alone features, sometimes making the use of the system cumbersome. An example of this is the feature to save a derived signal to a file, which has to be done manually for each signal, and includes some manual editing of configuration files. The user experience would be greatly improved by integrating different features better with each other, and creating more automated work-flows.

Bibliography

[1] P. Bakvis, K. Roelofs, J. Kuyk, P.M. Edelbroek, W.A.M. Swinkels, and P. Spinhoven. Trauma, stress, and preconscious threat processing in patients with psychogenic nonepileptic seizures. *Epilepsia*, 50(5):1001–1011, 2009.

[2] L. Bernardi, J. Wdowczyk-Szulc, C. Valenti, S. Castoldi, C. Passino, G. Spadacini, and P. Sleight. Effects of controlle dbreathing, mental activity and mental stress with or without verbalization on heart rate variability. *Journal of the American college of cardiology*, 35(6):1462–1469, 2000.

[3] S.H. Boutcher and D. Stocker. Cardiovascular response of young and older males to mental challenge. *Journal of Gerontology*, 51B(5):261–267, 1996.

[4] J.H. Houtveen, S. Rietveld, and E.J.C. De Geus. Contribution of tonic vagal modulation of heart rate, central respiratory drive, respiratory depth, and respiratory frequency to respiratory to respiratory sinus arrhythmia during mental stress and physical exercise. *European Heart Journal*, 17(3):354–381, 1996.

[5] D. Lucini, G. Di Fede, G. Parati, and M. Pagani. Impact of chronic physiosocial stress on autonomic cardiovascular regulation in otherwise healthy subjects. *Hypertension*, 46:1201–1206, 2005.

[6] M. Malik, T. Bigger, J Camm, R.E. Kleiger, A. Malliani, A.J. Moss, and P.J. Schwartz. Heart rate variability: Standards of measurement, physiological interpretation, and clinical use. *Psychophysiology*, 39(4):427–436, 2009.

[7] S. Naredi, Edn, S. E., Zall, H. Stephensen, and B. Rydenhag. A standardized neurosurgical/neurointensive therapy directed toward vasogenic edema after severe traumatic brain injury: clinical results. *Intensive Care Medicine*, 24(5):446–451, 1998.

[8] C. Schubert, M. Lambertz, R.A. Nelesen, W. Bardwell, J-B. Choi, and J.E. Dimsdale. Effects of stress on heart rate complexity - a comparison between short-term and chronic stress. *Biological Psychology*, 80(3), 2009.

[9] A. L. Steiner and P. J. D. Andrews. Monitoring the injured brain: Icp and cbf. *British Journal of Anaesthesia*, 5 2006.

[10] M. S. Thaler. *The Only EKG Book You'll Ever Need*. Lippincott Williams And Wilkins, 2006.

[11] M. R. Wahlstrom, M. Olivecrona, L.O. D. Koskinen, B. Rydenhag, and S. Naredi. Severe traumatic brain injury in pediatric patients: treatment and outcome using an intracranial pressure targeted therapythe lund concept. *Intensive Care Medicine*, 31(6):832–839, 2005.