Esaote

The assessment of blood vessel properties by means of automatic ECG detection

Graduation internship

Hubert Bessems

Supervision:

Dr. Ing. Peter. J. Brands (company tutor)

Ing. W. Schrijen (school tutor)

2010

Preface

This paper describes my activities during my graduation internship for the faculty of Electrical Engineering at Zuyd University during the spring of 2010. The report describes my development and other activities in a technical sense, but also describes the non-technical aspects. The main body is aimed at people with basic understanding of analog and digital electronics and the human anatomy (mostly the cardiovascular system). For non-technical readers I would recommend reading the summary in English or Dutch.

I would like to thank the following individuals for their time, effort and cooperation during the months of my internship:

- Dr. Ing. Peter. J. Brands – company tutor and supervisor
- Ing. J.J.W. Schrijen – school tutor and supervisor
- David Sontrop – technical mind in my office who I could always talk to
- Ing. L.M.C. Muijtjens – coordinator of internships at Zuyd University
- Jan Selanno – always willing to lend me tools he shouldn't give out
- Michel Vaessen – student who helped getting me started properly in my first week

Summary (English)

As part of their study of Electrical Engineering, students have two six month windows in which they do an internship at a company in the field. For my second window – my graduation internship – I chose Esaote Europe B.V. in Maastricht. This company specialises in development, sales and service of medical diagnostic equipment, specifically ultrasound and MRI scanners. The department at which I came to work during the early part of 2010 is related to ultrasound.

ART.LAB and Time Point Extraction

My direct superior is doctor Peter J. Brands, head of Advanced Projects. An important product of Advanced Projects is the computer programme ART.LAB, with which physicians and researchers can measure different blood vessel wall properties, and save the recorded radio frequency data (RF-data) and other measured properties into files for later analysis.

There are several different possible ART.LAB measurements, one of which was most relevant for my development project: the arterial wall velocity[1] and related change in diameter (distension) of the arterial walls. These properties, in the form of time-based signals, indicate the stiffness of a blood vessel, which is of use to researchers and physicians.

To make diagnosing with and researching on these signals easier, my predecessor developed a signal processing programme, extracting nine characteristic time points from these signals. The results from this programme can be used to extract temporal relationships between the different points.

Objective: Automatic ECG detection and processing

My main objective for this internship was to expand the MATLAB version of this programme, so that an ECG[2] signal could be used in this analysis. The ECG signal has several peaks and waves, of which the R-peak is largest and most recognisable. My programme should correlate the existing time points to their nearest ECG R-peak. To facilitate this objective I had to accomplish the following sub-objectives:

- Develop an automatic ECG detection and validation routine, to discard ART.LAB measurement files without valid ECG signal
- Make a programme that corrects the time base of the extracted time points, so that they use the ECG R-peak as a starting point
- Shape this into a programme that can automatically process an entire folder of ART.LAB measurement files

If I finished them ahead of schedule, more objectives would be added. All software development I did took place in MATLAB.

When I started at Esaote firstly I worked on the ECG detection and validation. When trying to relate the time points to the ECG signal I encountered problems with the existing programme, mainly concerning software safety. The existing programme would stop unexpectedly. I spent quite a lot of time ensuring the programme would either continue, or stop safely with a controllable error condition. Eventually I was successful, and was able to continue working on the automatic

[1] Arterial wall velocity: the speed at which the blood vessel walls distends from one another when the heart muscle contracts

[2] ECG (electrocardiogram): shows the electronic activity of a human heart

processing of an entire folder. In between, I was asked to add a tenth time point to the existing routine. I added this point and verified the results, together with Peter.

Three programmes

After that I went along and integrated everything into three programmes:

- ART.LAB Waveform viewer – to easily evaluate sets of measurements, per file
  This programme is meant to be used at first, to filter out unusable files
- AbsoluteTPE batch processor – to perform time point extraction, ECG correlation and statistics on an entire folder (meaning a set of measurements)
  This programme is meant to be used second in the order, to process all useful files
- AbsoluteTPE single file viewer – to perform time point extraction and ECG correlation on a single file and plot the results to the computer screen
  This programme is meant to be used last in the chain, to manually verify the results of the batch processor

During the development, Peter and I tested the software at many points and printed out many lists of values, to compare the results to those of earlier measurements on the same files. This way I ensured that I did not break things that worked earlier. Sometimes I also had to fix bugs or problems, but I ended up with a finished product, and created stand-alone executable versions next to the existing MATLAB scripts, so that users of my programs did not need MATLAB on their computers.

Other activities

During my time at Esaote my work was not limited to developing MATLAB-programmes alone. I was also responsible for documenting new products and programs, all around ART.LAB, releasing these documents and tending to our own design archive.

Next to that, whenever a new beta version of the ART.LAB software was released by the programmer David Sontrop I was asked to test this version from the perspective of a user, and report my findings, in the sense of bugs as well as things that had improved.

The ART.LAB project was also in a phase of going commercial, which means the Production department of Esaote was exercising the assembly of ART.LAB systems. I was asked to supervise this process, to make sure the instructions for the workers were clear, and update them when necessary.

Also, I visited the Maastricht University Medical Centre (MUMC+) with regular software updates for their ART.LAB systems, and collected feedback from the users, which I relayed to Peter Brands.

Next to that, I had to test the multi path latency in ART.LAB systems, to make sure this was acceptable. I used a signal generator and some electronics to generate a spike on all ART.LAB inputs.

Finally, related to my development, I visited *l'Hôpital Européen Georges-Pompidou* in Paris, where I delivered, demonstrated and explained the software I had created for my main objective. Later on I changed certain things to my programmes that were requested by the people at *Georges-Pompidou*, and sent the updated programmes to them via e-mail.

Conclusions

All in all, during the 5 months I was at Esaote, I believe I learned a great deal, and have seen a lot of different aspects of working in an engineering company. I feel my main objective was relevant to my education at Hogeschool Zuyd, especially the digital signal processing involved.

I was able to complete my main objective, but had no more time for extra objectives, as my other activities took quite some time as well. These extra objectives will be passed on to my successor. I do not see this as a problem, because the other activities were good experiences in a more general sense. They really amount to my personal development.

Summary (Dutch) – Samenvatting (Nederlands)

Als onderdeel van hun studie Elektrotechniek doen studenten twee stages bij een bedrijf in het werkveld. In het kader van mijn tweede stage – mijn afstudeerstage – koos ik Esaote Europe B.V. in Maastricht. Dit bedrijf specialiseert zich in de ontwikkeling, verkoop en het onderhoud van medisch-diagnostische apparatuur, specifiek in ultrageluid- en MRI-scanners. De afdeling waar ik in het eerste deel van 2010 te werken kwam heeft te maken met ultrageluid.

ART.LAB en Time Point Extraction

Mijn directe leidinggevende is Peter Brands, hoofd van de afdeling Advanced Projects. Een belangrijk product van Advanced Projects is het computerprogramma ART.LAB, waarmee artsen en onderzoekers verschillende eigenschappen van de wanden van bloedvaten kunnen meten. Met ART.LAB kunnen ze de opgenomen radiofrequente data (RF-data) en andere gemeten eigenschappen opslaan in bestanden, die ze later kunnen analyseren.

Er zijn verschillende ART.LAB-metingen mogelijk, waarvan er één soort het belangrijkst was voor mijn ontwikkelproject: de wandsnelheid[3] van bloedvaten en de daaruit volgende uitzetting (distensie) van deze wanden. In de vorm van tijd-gebaseerde signalen geven deze eigenschappen informatie over de stijfheid van een bloedvat. Deze informatie is bruikbaar voor artsen en onderzoekers.

Om de diagnose op en het onderzoek met deze signalen te vereenvoudigen heeft mijn voorganger een signaalverwerkingsprogramma ontwikkeld, dat negen karakteristieke tijdpunten uit deze signalen bepaalt (Time Point Extraction). De resultaten van dit programma kunnen worden gebruikt om tijdrelaties tussen de verschillende punten te leggen.

Doelstelling: Automatische ECG-detectie en –verwerking

Mijn hoofdopdracht voor deze stage was om het bestaande MATLAB-programma voor de Time Point Extraction zodanig uit te breiden, dat een ECG-signaal[4] gebruikt kon worden in deze analyse. Dit signaal heeft verschillende pieken en golfvormen, waarvan de R-piek het duidelijkst herkenbaar is. Mijn programma moest de bestaande tijdpunten correleren aan de dichtstbijzijnde R-piek in het ECG. Om dit mogelijk te maken moest ik de volgende subdoelen realiseren:

- Ontwikkel een automatische ECG-detectie- en validatieroutine, om ART.LAB-metingsbestanden zonder geldig ECG-signaal af te keuren
- Maak een programma dat de tijdbasis van de bestaande negen tijdpunten aanpast aan het nieuwe startpunt: de R-piek in het ECG-signaal
- Gebruik bovenstaande componenten in een programma dat automatisch een hele map met ART.LAB-metingsbestanden automatisch kan verwerken

Als ik vroegtijdig klaar zou zijn met deze opdracht zouden er nog meer deelopdrachten toegevoegd worden. Ik ontwikkelde alle software in MATLAB.

Toen ik bij Esaote begon werkte ik allereerst aan de detectie en validatie van het ECG-signaal. Toen ik vervolgens de tijdpunten probeerde te verschuiven aan de hand van het ECG kwam ik problemen tegen in de bestaande software. Soms stopte het programma zonder reden. Ik heb veel tijd besteed

[3] Wandsnelheid van een bloedvat: de snelheid waarmee de wanden van een bloedvat van elkaar af bewegen wanneer de hartspier samentrekt

[4] ECG (elektrocardiogram): geeft een beeld van de elektrische activiteit van een menselijk hart

aan het 'veilig' maken van dit programma, zodat er maar twee resultaten waren: het werkt goed of het belandt in een beheersbare fouttoestand. Uiteindelijk is dat gelukt en kon ik verder gaan met het realiseren van de automatische verwerking van een hele map. In de tussentijd werd ik gevraagd om een tiende tijdpunt aan de bestaande routine toe te voegen. Dit heb ik vervolgens gerealiseerd en de resultaten samen met Peter geverifieerd.

Drie programma's

Na het bovenstaande ben ik verder gegaan om het geheel in drie programma's te verwerken:

- ART.LAB Waveform viewer – bedoeld om gemakkelijk een set van metingsbestanden per bestand te beoordelen. Hiermee kunnen onbruikbare bestanden worden geweerd
- AbsoluteTPE batch processor – om Time Point Extraction mee te doen, de resultaten te correleren aan het ECG-signaal, en statistieken toe te passen op een hele map met metingsbestanden (een map met metingen staat hierin gelijk aan een set met metingen)
- AbsoluteTPE single file viewer – om Time Point Extraction mee te doen, de resultaten te correleren aan het ECG-signaal en de resultaten op het scherm te laten zien
  Dit programma is voornamelijk bedoeld om als laatste in de ketting te worden gebruikt, om na afloop de resultaten van de batch processor na te kijken

Tijdens de ontwikkeling hebben Peter en ik de software op vele punten getest en vele lijsten met resultaten uitgeprint om deze te vergelijken met de resultaten van eerdere metingen. Op deze manier stelde ik zeker dat ik dingen die eerder correct werkten niet alsnog stuk maakte. Soms moest ik bugs of andere problemen verhelpen, maar uiteindelijk kwam er een afgewerkt product uit. Hiervan heb ik naast MATLAB-scripts ook los uitvoerbare versies van gemaakt, zodat gebruikers niet per se MATLAB op hun computer hoefden te hebben staan.

Overige activiteiten

Ik heb gedurende mijn tijd bij Esaote niet uitsluitend aan MATLAB-programma's ontwikkeld. Ik was ook verantwoordelijk voor het aanleggen van documentatie voor nieuwe producten en programma's rondom ART.LAB, het vrijgeven van deze documenten en het op orde houden van ons eigen ontwerpdossier.

Daarnaast werd ik, wanneer er een nieuwe betaversie van het ART.LAB-programma vrijgegeven werd door de programmeur David Sontrop, gevraagd om het programma te testen vanuit een gebruikersperspectief. Ik rapporteerde mijn bevindingen zowel in de vorm van gevonden bugs, als dingen die verbeterd waren in de nieuwe versie/

Buiten dat stond het ART.LAB-project stond op het punt om commercieel te gaan. Dit betekende dat de productie-afdeling van Esaote oefeningen draaide in het bouwen van ART.LAB-systemen. Ik werd gevraagd om deze oefeningen te overzien, te zorgen dat de instructies voor de medewerkers duidelijk waren en ze waar nodig aan te passen.

Ook bezocht ik regulier het MUMC+ (Maastricht University Medical Centre) met software-updates voor de daar aanwezige ART.LAB-systemen. Hierbij verzamelde ik ook feedback van de gebruikers, die ik mee terug bracht naar Peter Brands.

Verder heb ik tests gedaan aan de vertraging in de meerdere paden van het ART.LAB-systeem. Door de verschillende paden ontstaan verschillen in de vertraging en het was belangrijk om te bepalen of

deze verschillen acceptabel waren. Hiervoor gebruikte ik een functiegenerator die op alle ingangen een impuls genereerde.

Tenslotte werd ik gevraagd om in *l'Hôpital Européen Georges-Pompidou* in Parijs de drie programma's van mijn hoofdopdracht uit te leggen en te demonstreren. Na het bezoek heb ik nog zaken aangepast op basis van de feedback van de mensen in het *Georges-Pompidou* en de aangepaste programma's heb ik via e-mail aan hen gestuurd.

Conclusies

In het geheel gezien heb ik tijdens mijn 5 maanden bij Esaote veel geleerd en veel verschillende aspecten van het werken bij een technisch bedrijf leren kennen. Ik heb het gevoel dat mijn hoofdopdracht goed aansluit bij mijn opleiding aan de Hogeschool Zuyd, met name op het gebied van digitale signaalverwerking.

Ik was in staat om mijn hoofdopdracht te volbrengen, maar vanwege de tijd die werd opgeëist door mijn andere activiteiten ben ik niet meer toegekomen aan nieuwe deelopdrachten. Deze doelen zullen worden doorgegeven aan mijn opvolger. Ik zie dit niet als een probleem, omdat de andere activiteiten voor mij goede ervaringen waren in een andere zin dan ontwikkeling. Ik heb er als persoon veel van geleerd.

Contents

Chapter 1 - Introduction

Students who follow the entire programme at the faculty of Electrical Engineering, at Zuyd University in Heerlen, have two six month windows in which they are expected to work at companies in the field in the form of internships. The first internship – after two years of studying - is meant to let students acquaint themselves with different aspects of work in the field. The second window is at the end of the four year education period, and is considered a graduation internship. In this internship, the focus is mostly on testing a student's factual knowledge on the subject of electrical engineering, in the form of one central assignment.

This paper was written in context of my graduation internship, which was from February until and including June 2010 at Esaote Europe B.V. in Maastricht. I established first contact with them after receiving their telephone number from the internship coordinator at school, ing. Leon Muijtjens.

In the first telephone conversation, I was invited to meet with dr. Peter J. Brands, head of the department *Advanced Projects* to see if I would fit in at his department, and if the work he had in mind for me would in turn suit me. When I met with him a few days later, the assigment he had in mind seemed quite interesting to me and I realized that working at Esaote for a while would suit the profile for a graduation internship, as well as enhance my experience in the work field. I quickly received mr. Muijtjens' consent to begin at Esaote.

In chapter two of this paper I shed some light on the context of my activities and how they fit in the activities of Esaote. Chapter three explains my main assignment and chapters four, five and six go into more detail on the realisation of my main assignment. Chapter seven describes my other activities for Esaote, which took place next to my main assignment. And in the last chapter I draw conclusions based on my work and place recommendations for my successor or any other person who would like to continue where I left off.

I would recommend reading this paper in the order that it has been written. Persons with little interest in technical information could skip chapters four, five and six except for their conclusion paragraphs.

Chapter 2 - Esaote Europe & ART.LAB ultrasound research

To place my work in context, first I will introduce the company I worked for during my internship and secondly the project my activities were for.

Esaote Europe B.V.

Figure 1: The Esaote Europe building in Maastricht

Esaote Europe B.V. is situated next to the A2 motorway in Maastricht, and started out in 1982 with the name *Pie Medical Benelux* as a manufacturer and seller of ultrasound equipment. In 1997 *Esaote Group* acquired *Pie Medical Benelux* and in 2006 the Maastricht unit was renamed to *Esaote Europe B.V.*.

Esaote Group is based in Italy, with its headquarters in Genoa, and has been producing several types of biomedical equipment, most notably ultrasound and MRI scanners. Currently, Esaote Europe in Maastricht develops, sells and services many Esaote ultrasound products. To facilitate these activities, the Maasticht centre has two laboratories, a production and testing facility and several other departments, including logistics, marketing and documentation. The department where I worked during my internship is *Advanced Projects*. This is a research and development department with connections to, among others, CARIM[5] and the faculty of Biomedical Technology at the Technical University of Eindhoven. Both institutes are involved in the ART.LAB ultrasound research project.

[5] CARIM: Cardiovascular Research Institute Maastricht, which is part of the Maastricht University Medical Centre

ART.LAB ultrasound research

The ART.LAB project offers a platform for various types of research based on ultrasound scans. After briefly explaining ultrasound I will introduce the ART.LAB platform.

Brief introduction to ultrasound

Ultrasound in the literal sense of the word means sound with a frequency beyond human hearing range (>20 kHz). Within certain materials, these high-frequency acoustic waves behave like radio waves, and therefore are called radio frequency signals (RF signals). RF signals in the order of 250 kHz to 10 MHz travel quite well through human tissue.

This property is used in ultrasound. An ultrasound transducer is equipped with piezoelectric crystals, that resonate in a certain frequency when a voltage impulse is applied to them. The resonant energy travels outward from the crystal, into the tissue, following a straight path. Whenever these waves trespass between two types of tissue, each with a different acoustic impedance, a reflection is sent back. The piezo crystal will in turn convert this received reflection wave into a voltage. Using the time between sending of the pulse and the reflections, combined with the speed of sound in human tissue, the depth of every reflection is known.

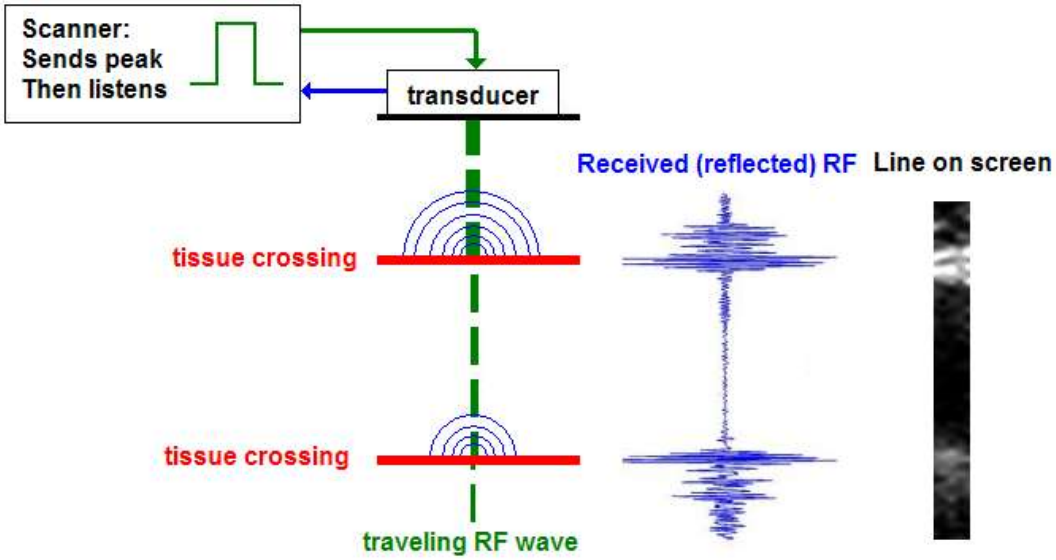
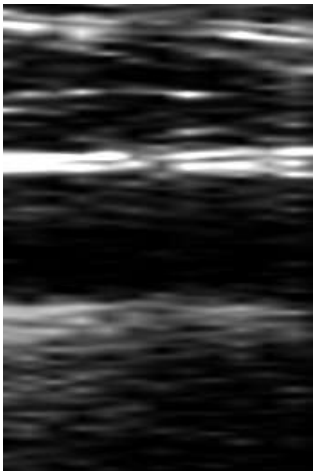


Figure 2: Ultrasound principle



By use of signal processing, ultrasound scanners transform the reflected radio frequency signals (RF signals, in blue) into a monochrome line on screen. This line is light when the reflected RF-signal is intense and dark when there is almost no reflected RF-data.

By combining several of these lines in an array, one can form a two-dimensional image, and by using different transducer shapes and on-screen projections it is also possible to get images from unborn children in the womb of their mother, which is a well known application of ultrasound.

Figure 3: 2D echo image of blood vessel

ART.LAB platform

The ART.LAB platform consists of a high-end ultrasound scanner device made by Esaote, specifically the MyLab 70 XVG. This scanner is upgraded with a digital optical output and software. With these upgrades, the scanner is able to output the unprocessed RF-data from its scan over an optical interface. At the other end of this interface we have a high end PC, with optical receiver, and the ART.LAB software. To start and stop the transfer of RF data between the scanner and the PC, the TCP/IP Ethernet link is used. The ART.LAB PC can also be fitted with physiological inputs, for ECG signals and/or blood pressure measurements. These external signals can be useful when measuring blood vessel properties, which is the main goal of the ART.LAB platform.

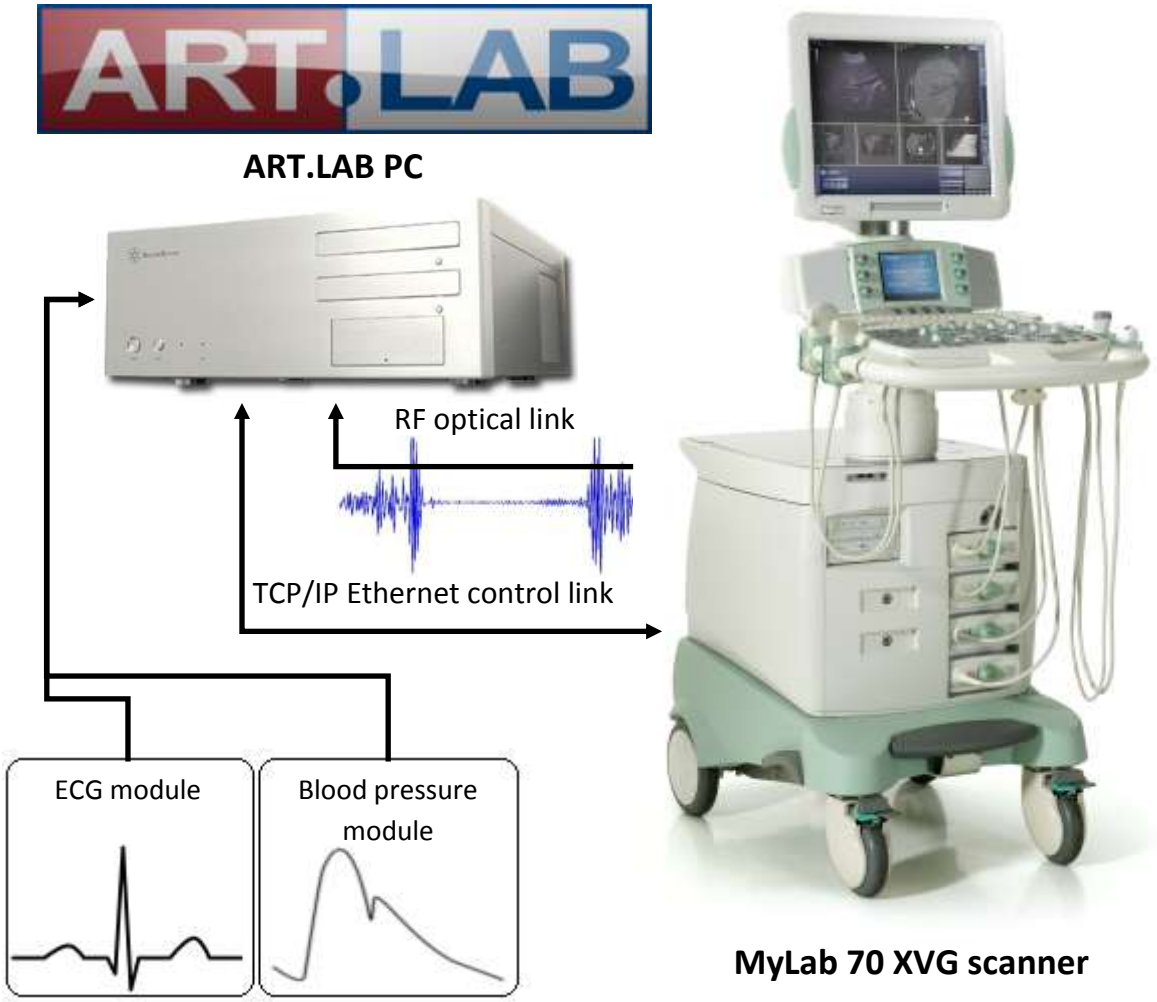


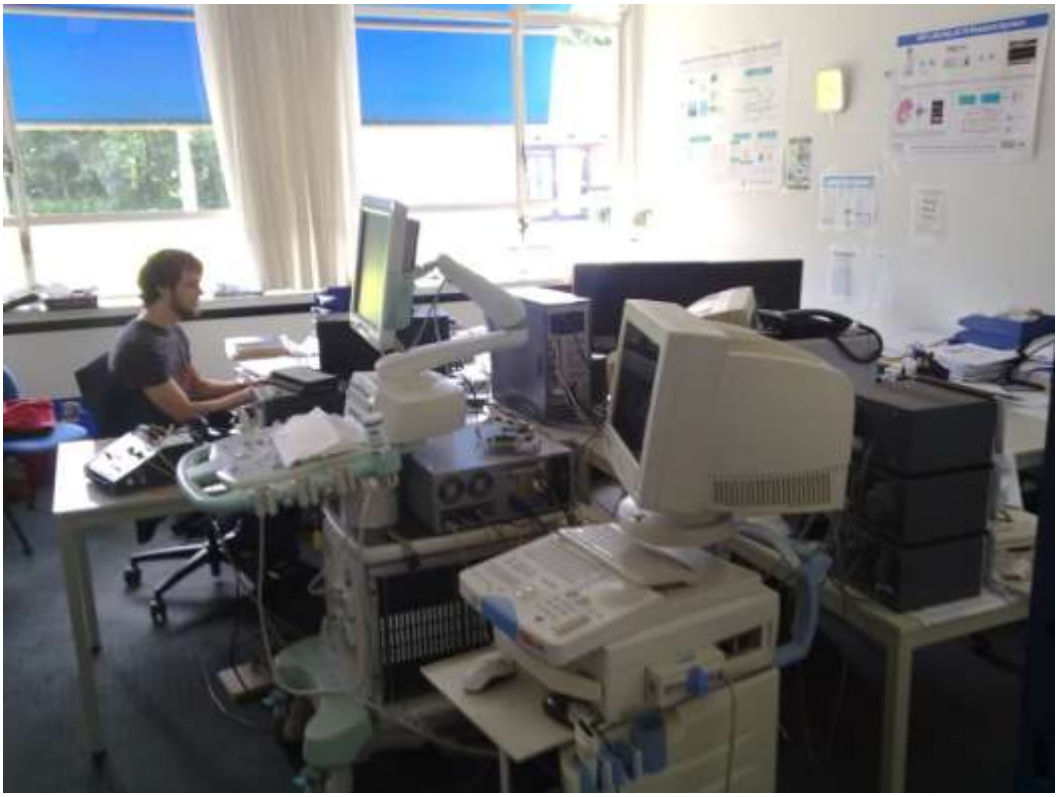
Figure 4: Schematic overview of ART.LAB platform

When combined, the ART.LAB PC, the scanner and software are able to perform real-time measurements and signal processing. The ART.LAB PC can save the results of these measurements, but also export raw, unedited RF data, which can be used by researchers to investigate applications for ultrasound, and use it for several different applications. Still, the subtitle for ART.LAB is *The arterial analyzer*, meaning that most users will use ART.LAB for scanning and analyzing blood vessel properties. The main objective for my internship is related to blood vessel properties as well. I will elaborate on this in the next chapter, where I will also reveal my main objective.

The Advanced Projects Room

Our test scanner

The 'to do' whiteboard



A view of the entire room

Most of the time I would sit, in at my desk in the far corner of the last picture. On the left you see graduating intern Sytse, from the ICT Faculty of Zuyd University, and the desk in the right corner belongs to David Sontrop, the main programmer of ART.LAB.

Chapter 3 - Main assignment

My main objective or assignment is meant as an addition to a specific application of the ART.LAB platform, specifically the measurement of arterial wall velocity[6], and the related change in diameter of the arterial walls. In this chapter, I will first introduce this application of the ART.LAB platform, and then reveal my main assignment for this internship.

Arterial wall motion measurement

Using the basic principles of ultrasound described in the previous chapter, the ART.LAB software can construct a two-dimensional cross section echo image of a blood vessel based on the unprocessed radio frequency (RF) data that enter the ART.LAB PC over the optical link. The software itself is capable of detecting the position and speed of the artery walls from this stream of RF data. It places markers on screen, on top of the echo image. Using its tracking mechanism, the ART.LAB software recognizes when the artery walls move away from one another or back together under influence of blood flow and blood pressure. The walls move away from one another when the heart beats, and back together when the wave of blood from the heart beat has passed. The change in artery diameter (distension) is directly integrated from the wall velocity. In the below image you can see how the ART.LAB shows a blood vessel expanding. The orange line is the actual distension, the blue line is an amplified version of the orange one.

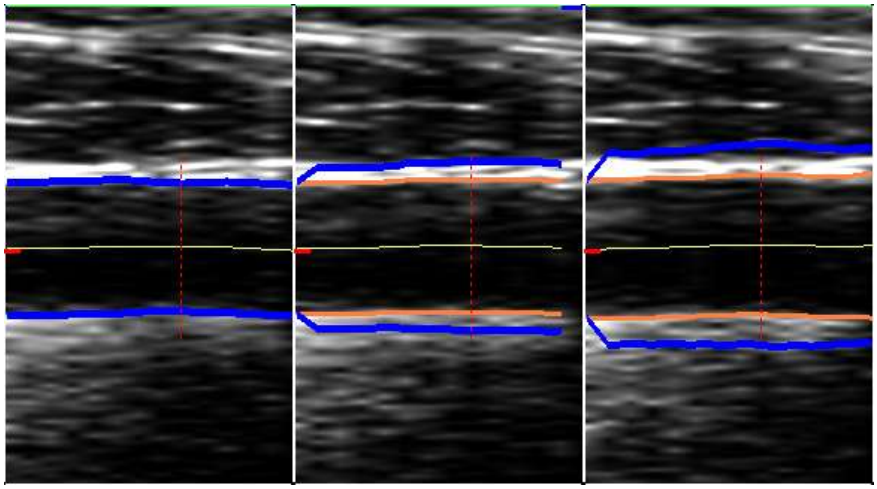


Figure 5: ART.LAB arterial wall motion measurement

To answer a question from physiological researchers on the flow speed of blood pulse waves, Esaote needed a way to correlate multiple measurements based on time. These measurements could only be correlated if they had a common time marker. My predecessor had developed a programme in MATLAB to do relative time analysis (Time Point Extraction) on the wall velocity signals, but he had no chance to add an absolute time marker to every measurement.

In ART.LAB measurements with a correctly recorded ECG[7] signal, the R-peak in that signal can be used as a common and absolute time marker between files. The R-peak is the point in time when the heart muscle begins to contract. When the heart muscle contracts, it pumps out blood, leading to a wave front in the arteries. From that point, there is a certain time until the blood wave front reaches

[6] Arterial wall velocity: the speed at which the blood vessel walls distends from one another when the heart muscle contracts

[7] ECG (electrocardiogram): shows the electronic activity of a human heart

the locations where measurements took place. With the times between the R-peak and the existing time points, researchers could calculate the pulse wave velocity. This research would be based on a large data set, consisting of hundreds of measurements. It may be clear that the existing programme needed expansion.

My main assignment

From there my main assignment was formulated as follows: extend the existing MATLAB-based Time Point Extraction programme to use the ECG R-peak of every heart beat as a time base. To facilitate this objective I had to accomplish the following sub-objectives:

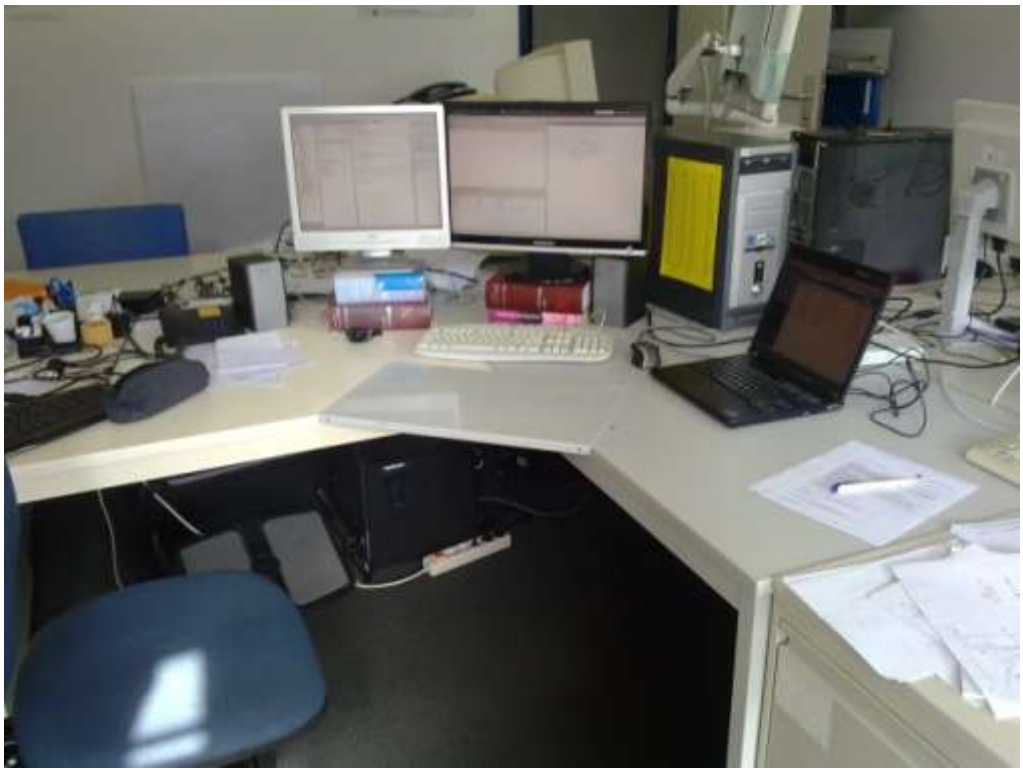
- Develop an automatic ECG detection and validation routine, to discard ART.LAB measurement files without valid ECG signal
- Make a programme that corrects the time base of the extracted time points, so that they use the ECG R-peak as a starting point
- Shape this into a programme that can automatically process an entire folder of ART.LAB measurement files

The first and last sub-objective are both in the interest of processing large sets of data, with many measurements. Having to process hundreds of files by hand, one by one, would be cumbersome. So the need to make things automatic is quite clear.

If I finished them ahead of schedule, more objectives would be added. All software development I did took place in MATLAB. The ECG validation, detection and time base correction are explained in chapters four and five, and the batch processing routine in chapter six.

My workspace

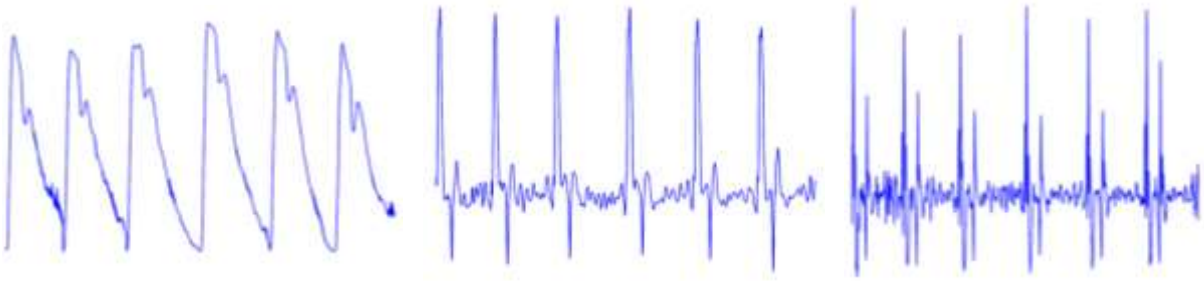
When I was in the office, this was my work space. Multiple screen setup to simplify making and testing larger programs.



Chapter 4 - Time Point Extraction development

This chapter describes the development of Time Point Extraction in great detail. The majority of this work has been carried out by my predecessor at Esaote, Marc van Dijk.

Introduction



Distension Velocity Acceleration

Figure 6: The three characteristic waveforms

When a human heart beats, it influences blood flow and pressure, causing arteries to distend. By digital signal processing on RF data from ultrasound scans, the waveforms shown above can be found. Distension tells us how far an artery has expanded in time. The velocity signal is the first derivative of the distension signal, telling us how fast the artery walls are moving. Finally, the acceleration signal tells us about the acceleration of the artery walls, hence, how fast the speed of the artery walls is changing.

All three signals are important, because together they can be used for the calculation of waveform parameters, in the form of several sample points. These sample points can in turn be used to diagnose cardiovascular conditions.

At first, Esaote wanted the following sample points to be known

SIC	Start of isovolumic contraction
AVO maximum	Aortic valve opening maximum
AVO minimum	Aortic valve opening minimum
Maximum distension	Maximum distension of the artery being measured
AVC minimum	Aortic valve closure minimum
AVC maximum	Aortic valve closure maximum
DN	Dicrotic notch
PR	Lower body peripheral reflection wave

Esaote asked Marc van Dijk, graduate student performing an internship at Esaote Europe B.V. to develop an algorithm that could extract these sample points, which can be converted to time points.

MATLAB-programme

During his time at Esaote, my predecessor Marc van Dijk used MATLAB to develop an algorithm capable of extracting the mentioned sample points. The following section will explain the working of this algorithm, which is called *TPE* (acronym for *Time Point Extraction*).

Step 1: Data input

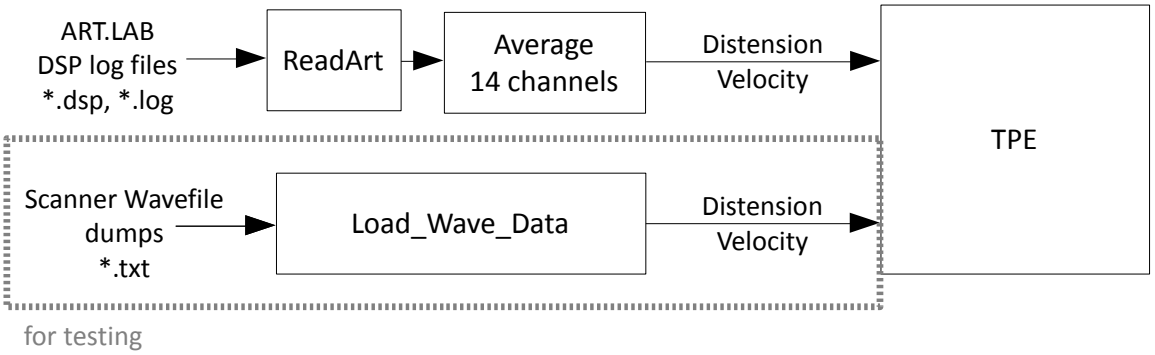


Figure 7: TPE input path

The three characteristic waveforms that TPE needs usually come from a DSP log file made by ART.LAB. This text file contains DSP data calculated by ART.LAB during an ultrasound scan. The data is formatted in comma separated values and needs some pre-processing before sending it to the TPE routine. This pre-processing is taken care of by ReadArt and a function to average the 14 Distension and Velocity signals to one of each.

During testing Marc used wave data files from other ultrasound scanners instead of ART.LAB files. These were text files, containing hexadecimal representations of Distension and Velocity curves. They need a different kind of preprocessing.

Both ways input a single Distension and a single Velocity signal into the TPE routine.

At the input, TPE applies two filters to the Velocity signal:

1. Median filter with filter window 3. This means it takes the middle value out of three neighbouring values, and puts that value in the middle, leaving the rest as they were.
2. Moving average filter with filter window 5. This means it adds the current value, two values before and two values after it, divides this sum by 5 and puts it in place of the current value.

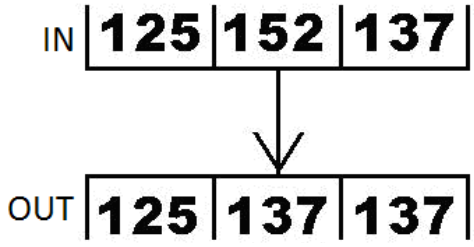


Figure 8: Median filter

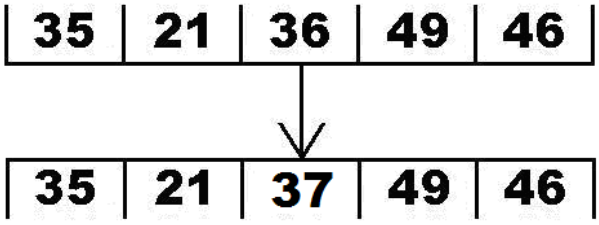


Figure 9: 5 point moving average filter

Step 2: Find Velocity maximums

To separate the beats that exist in the input signals, TPE searches for Velocity maximums. Using their position and interval, the program can establish beats.

The programme takes the maximum value in the whole Velocity signal, multiplies it by 0.7 and takes that value as a dynamic threshold.

For every range of sample points above the threshold, it searches the maximum value, which is the Velocity maximum of that beat.

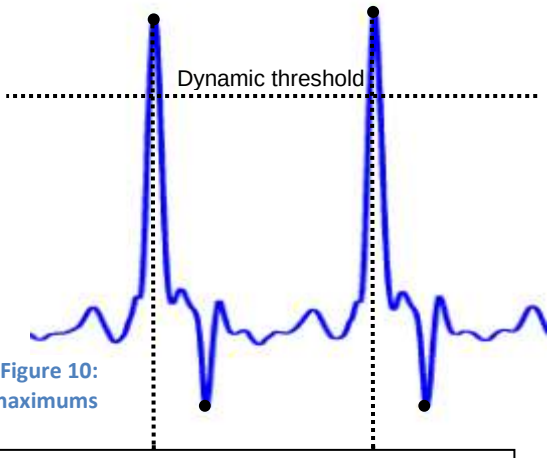


Figure 10: Finding Velocity maximums

For these and other searches,

```
% search maximum Velocity value sample point in range from Start to End

VelocityMaximum = find(Velocity(Start:End) == max(Velocity(Start:End)));
```

it uses the find function in Matlab. An example is shown below:

In this example, VelocityMaximum becomes the sample point (somewhere from 1 to *n*, where *n* is the length of Velocity in samples).

Then it determines the smallest period between two Velocity maximums, which becomes the reference for all search windows, and will be called smallest Velocity period from now on.

Step 3: Find Velocity minimums

After the maximums, we need to find the Velocity minimums. These will facilitate the search of other points along the way.

The programme finds the minimums by establishing the length of search window 2, which is half of the smallest Velocity period. Then it searches for the lowest Velocity value in search window 2 after the Velocity maximum, as seen in this figure.

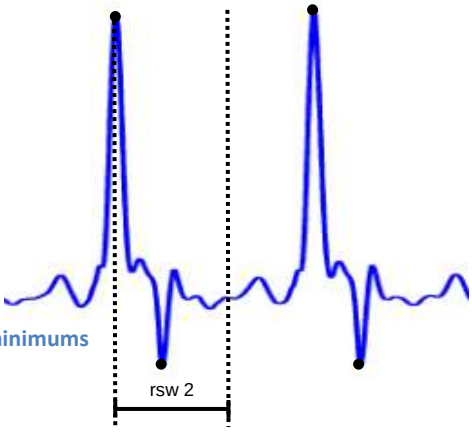


Figure 11: Finding Velocity minimums

Again, it uses the find function, in the following fashion:

```
% search minimum Velocity value sample point in range from
VelocityMaximum to VelocityMaximum + SearchWindow2

VelocityMinimum = find(Velocity(VelocityMaximum:(VelocityMaximum +
SearchWindow2)) == min(Velocity(VelocityMaximum:(VelocityMaximum +
SearchWindow2))));
```

The result here is relative to the Velocity maximum. To make the value absolute, it is offset by the Velocity maximum.

If there are insufficient Velocity maximums or minimums found, the programme stops here, else it goes on to find the other points.

Step 4: Establish beats

To search the other points in every beat, the programme needs to know limits for its search. To this end, the programme sets start and end points for the beats.

It establishes search window 3, which is 0.15 * the smallest Velocity period. The beat's beginning is set one search window 3 before its Velocity maximum. The ending is set one smallest Velocity period after the beginning, so 0.85 * smallest Velocity period after its maximum.

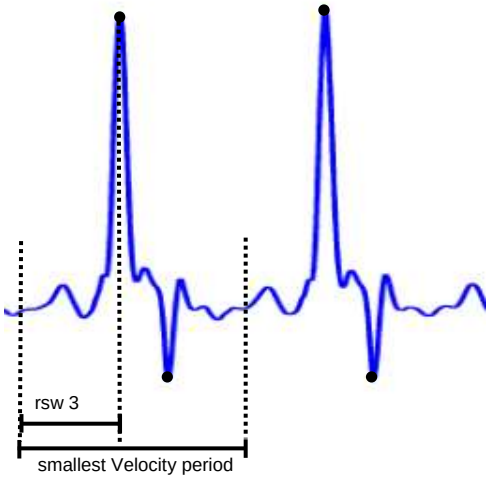


Figure 12: Establish start and end points

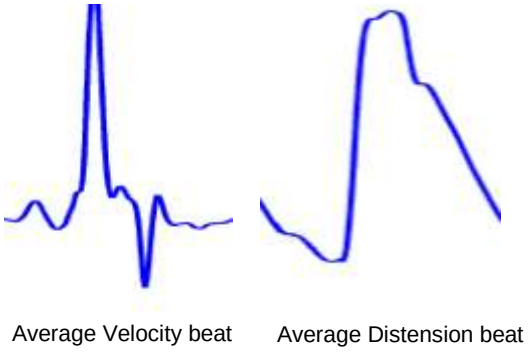
Here, the programme also checks if any beat's beginning is set before the beginning of the Velocity signal, or if an ending is set after the end of the Velocity signal. If this is the case, it discards the entire beat, so that the rest of the programme does not search there.

After this step, the programme checks if it has at least 3 beats. If not, it stops and reports TPE was not possible on these signals.

Figure 13: Average beats

Step 5: Determine average Velocity and Distension beats

The beat ranges in step 4 are all one smallest Velocity period long. Using these beat ranges on the Distension and Velocity signal, the programme can establish several separate beats of identical length. It averages these beats to output one average Velocity beat, and one average Distension beat.



In the previous steps all prerequisites have been gathered, so now the programme can start searching for the wanted sample points. It searches one sample point for every beat, and then moves on to the next sample point.

Step 6: Find AVO maximum and minimum

The first sample points to be searched for are the AVO maximums and AVO minimums.

First, the programme establishes the length of search window 3b, which is 0.2 * smallest Velocity period.

AVO maximum is the maximum Acceleration point within one search window 3b before the Velocity maximum.

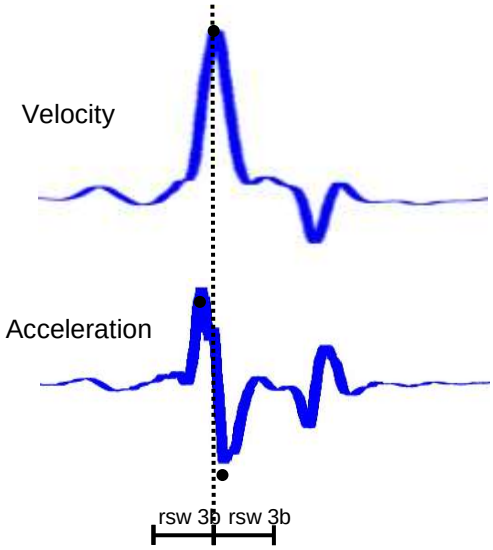


Figure 14: AVO maximum and minimum

AVO minimum is the minimum Acceleration point within one search window 3b after the Velocity maximum.
Both points are searched for using MATLAB's find function, which has been demonstrated earlier in this chapter.

Step 7: Find SIC

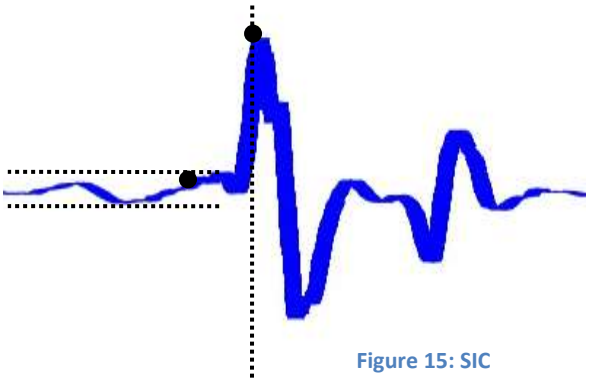


Figure 15: SIC

Next to be found is the SIC. The programme looks for all minimums and maximums in the Acceleration beat before the AVO maximum.

Then it searches the biggest amplitude difference between any minimum and its following maximum point, regardless of time between them.

The maximum of this transition is taken as SIC.

Step 8: Find AVC minimum and maximum

Two related points the programme also tries to find are the AVC minimum and maximum.

Before starting the search, the length of search window 5 is established, at 0.1 * smallest Velocity period.

AVC minimum is found by taking the minimum in Acceleration in one search window 5 before the Velocity minimum.

AVC maximum is found by taking the Acceleration maximum in one search window 5 after the Velocity minimum.

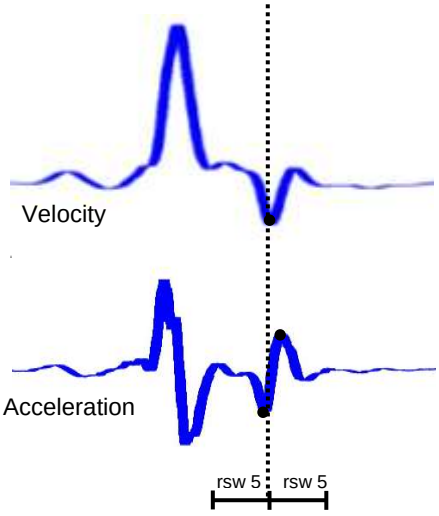


Figure 16: AVC

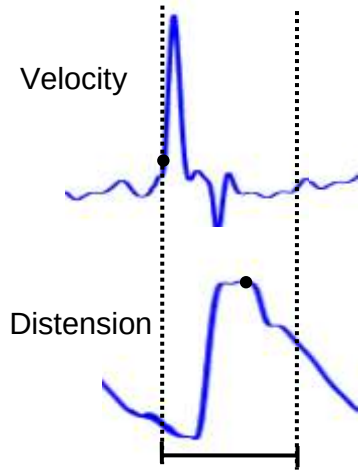


Figure 17: Maximum distension

Step 9: Find maximum in Distension

Some time after the Velocity peak, the Distension cycle has its maximum. In order to search this point, the programme re-establishes the length of search window 2, which is half of the smallest Velocity period.

The Distension maximum point is found by taking the maximum Distension point in one search window 2 after the AVO maximum. The programme could search the entire beat, but limiting the search reduces the chance of invalid peaks interfering.

Step 10: Find PR and DN

The last set of points TPE has to find are the PR and DN points. The search window established is 3b, which has a length of 0.2 * smallest Velocity period.

First the programme finds the Velocity maximum value within one search window 3b after the Velocity minimum from step 3.

From that point, it can find PR and DN.

PR is found by taking the maximum in Distension within one search window 3b after the secondary Velocity maximum.

DN is then found by taking the minimum in Distension within one search window 3b before the PR point.

If PR and DN are too close in time, both points are discarded.

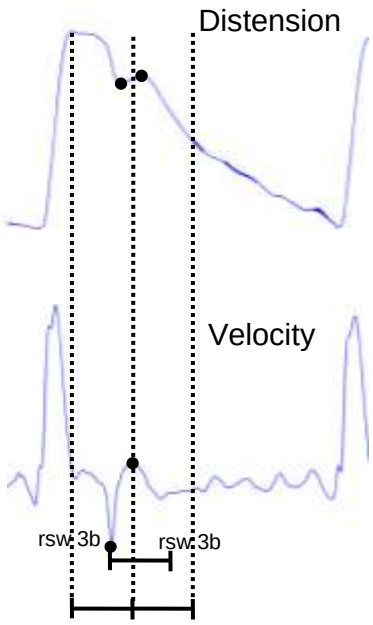


Figure 18: PR and DN

Data output

When all these searches have taken place, the programme calculates averages for every sample point. It then puts out an array with all sample points found, the average Distension beat, the average Velocity beat and the average sample points.The table below shows how the original programme returns the time points. After running the programme, there are two tables in memory:

- *ExtractedTimePoints*: 10 elements high, n elements wide (one for every beat)
  This table contains all found time points, relative to the beat start point. This table uses a different row for every type of time point, and a different column for every beat.
- *AverageTimePoints:* 10 elements high, 1 element wide
  This table contains the average values of the first 8 time points in the other table. It averages these time values over all the beats. So *AverageTimePoints(1,1)* is the average of all elements in *ExtractedTimePoints(1,…)*

ExtractedTimePoints (10 x n)

	1	2	3	...	n
1	SIC				
2	AVO maximum				
3	AVO minimum				
4	Maximum distension				
5	AVC minimum				
6	AVC maximum				
7	DN				
8	PR				
9	Beat end Point				
10	Beat start Point				
	Beat 1	Beat 2	Beat 3	Beat ...	Beat n

AverageTimePoints (10 x 1)

	1
1	SIC
2	AVO maximum
3	AVO minimum
4	Maximum distension
5	AVC minimum
6	AVC maximum
7	DN
8	PR
	Averages

Figure 19: Original TPE output format

An example of this output format:

ExtractedTimePoints (10 x 4)

	1	2	3	4
SIC	40	43	39	0
AVO max	65	66	64	66
AVO min	104	100	104	100
Max dist	118	115	118	116
AVC min	235	216	232	220
AVC max	251	231	247	238
DN	266	243	258	249
PR	315	291	304	304
Start pt	119	735	1271	1832
End pt	536	536	536	536
	Beat 1	Beat 2	Beat 3	Beat 4

AverageTimePoints (10 x 1)

	1
SIC	41
AVO max	65
AVO min	102
Max dist	117
AVC min	226
AVC max	242
DN	254
PR	304
	Averages

Problems with existing TPE code

While developing the *ECG to TPE correlation* program (see chapter 7 for details), I found out that the existing Matlab TPE routine would succeed on several text files, but it would unexpectedly stop when processing certain ART.LAB DSP-logfiles, giving out an error about the program trying to access an index that was out of a given array's bounds.

The TPE routine is meant to be used in batch processing, so these sudden stops were unacceptable. The routine should give out valid results, or stop neatly, reporting that its results are unusable. Perhaps even a detailed error description could be printed to the screen or an output file.

Trying to fix the bugs in the code seemed impossible to me. Most variable names were very short and uninformative. Also, there were very few comments in the code, and they all were very short. I could rebuild a new safe routine from scratch, but I feared that output results would not be the same as the old *TPE* routine. The exact identicalness of these outputs was a requirement for this project, because it was originally the intention to use the original *TPE* routine for this.

When the original TPE is started, some checking is performed before parameter search starts, and if TPE is deemed impossible the programme should stop, return zeros and end. Alas, this mechanism did not always work.This was an issue, and I thought about possible ways to tackle this issue. I devised the following approach:

1. Thoroughly test the reliability of the existing TPE code
2. Study the TPE method outside MATLAB, using the existing documentation (PowerPoint presentations etc.)
3. Read through the code using the documentation, and try to find out what everything means
4. As I go through the code, try to recreate it, with clearly named variables, and more elaborate comment in the code
5. Compare *TPE's* original results to my version's results throughout development
6. Add extra safety features in the code and retest

Rewrite of TPE code

A few weeks into February I started working on this, expanding the project step by step.

While rewriting TPE, I recreated the same order of doing things as in the old version. Also, I kept in mind that every process could go wrong on its own, and some were interdependent. So I implemented more rigorous checking at the front, and also, along the road, the programme checked if results from previous searches were valid. If not, the programme would not search new parameters, because I consider a search based on invalid data quite unsafe.

Furthermore, I removed a design choice that seemed strange to me. The old programme used the number of parameters of the previous search to determine how many of the next parameter had to be found. But, this number was always equal to the number of maximums. A code example:

```
if count < length(top)/2;

    for a = 1:length(avo);

        parameters(2,a)   = 0;

    end

end
```

This code bases the number of SICs to be searched on the number of AVO maximums. In the rewrite of TPE the number of valid beats is established at the start, and every parameter is searched for every beat, according to the given number of valid beats. This is clearer to the reader, and more reliable if something would go wrong along the way.

Other changes were mostly checking if a search would stay within its array bounds before deciding to start it, and using clearer, longer names for variables. I also split the calculation of averages. In the old routine, all averages were calculated at the end, but in the rewritten version of TPE, every parameter's average is calculated directly after these parameters have been found.

Comparison of output

During the development I added extra output variables to the original TPE and my rewrite, to see if the results along the way were the same. Every time I added a variable to the rewrite, I would run a program to see if the output of both routines was the same.

Matlab code of comparison program

```
[Distension, Velocity] = DSPtxtToMem();

% original TPE

[av_vel, av_dist, avo, avo_min, sic, avc, avc_min, start_point,
end_point, avg_parameters, parameters] = TPE(Distension, Velocity);

% rewritten TPE

[AverageVelocityBeat, AverageDistensionBeat, AVOtops, AVOminimums,
SICs, AVCtops, AVCminimums, BeatStartPoints, BeatEndPoints,
AverageTimePoints, ExtractedTimePoints ] = RevisedTPE(Distension,
Velocity);

[…]

if(isequal(AVOtops,avo))

    fprintf('TPEcompareResults: AVO tops equal\n');

else

    fprintf('TPEcompareResults: AVO tops UNEQUAL\n');

end
```

Example output

```
TPEcompareResults: Average velocity beat results equal

TPEcompareResults: Average distension beat results equal

TPEcompareResults: AVO tops equal
TPEcompareResults: AVO minimums equal
```

This example is from when I had not completed all parameters yet. The AVO-tops and minimums were the same, but the rest was not. This means that the ‘Total parameters’ (big output array) was not exactly the same yet, and the average values were not as well.

As development progressed, every point was filled in to the point that the results between both versions were exactly the same.

Batch comparison

To ensure this backward compatibility and functional equality, I developed a Matlab programme which enables me to process a whole folder of .txt files, called *DSPbatchCompareTPE*. These .txt files contain hexadecimal representations of Distension and Velocity signals. Marc van Dijk received these files from Esaote Italy and used them for testing while he was developing the original TPE. *DSPbatchCompareTPE* runs the old and new TPE on every file and compares the results. This makes it easy to verify if the routines produce consistent equal results.

The order of working of *DSPbatchCompareTPE* is as follows:

1. The programme checks the validity of the directory, and searches it, recursively or not, in accordance with specified input parameter. A list of files results

2. For every file it tries to read out the waveforms, using DSPtxtToMem.m and Load_Wave_Data.m

3. If read succeeds, it lets the old and the rewritten TPE run on the read file

4. The results are (when possible) printed to a file called *DSPbatchCompareTPE-results.csv*, which is placed in the input folder. If this file cannot be placed, it asks the user where it wants to store the output file. All time points are written to this file. Also, all time points, averages and average Distension and Velocity beats are compared, and the programme writes to file if these average beats are the same.

5. After the comparison of all files, the program prints how many files it was able to process, and how many had equal results.

I rebuilt Marc's code according to the plan that I mentioned at the start of the document, and when I had completed the rebuild until step 5, I ran a *DSPbatchCompareTPE* test on the folder with test files. The results on every file were exactly equal, even when files could not be processed.

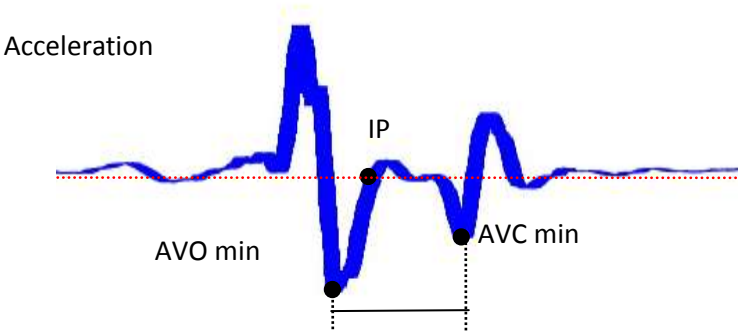
After this test I added more safety checks in the routine, and ran another round of *DSPbatchCompareTPE* on the same folder. All results were again exactly the same as in the old *TPE*, so my extra safety routines have not negatively influenced the accuracy of the programme.

Addition of extra time point

During my work, I was asked to add another time point to the detection. This point would yield more diagnostic information than the existing AVO minimum point. It is called the Inflection Point (IP). The detection rules for this point were simple, and applied beat to beat.

- Only search for this point when an AVO minimum and AVC minimum have been found
- The point is at the first negative to positive zero crossing in the Acceleration signal, between AVO minimum and AVC minimum
- If no zero crossing is found within the specified range, no IP is found

To facilitate the addition of this point, I had to change the output format of the TPE routine. The two-dimensional array that puts out all found time points needed another row for IP points. The one-dimensional array for the average time points needed one extra element for the average TPE point.



New output format

Knowing I had to use this routine in batch processing, I added other outputs as well. Data I deemed to be very useful were the standard deviations on valid time points. So, I modified the routine to calculate this as well, and it resulted in the following output format (also see the old format and explanation at figure 19). The new format includes the Inflection Point, and standard deviations on the time points.

ExtractedTimePoints (11 x n)

	1	2	3	4	...	n
1	SIC					
2	AVO maximum					
3	AVO minimum					
4	Maximum distension					
5	Inflection point					
6	AVC minimum					
7	AVC maximum					
8	DN					
9	PR					
10	Beat start point					
11	Beat end point					
	Beat 1	Beat 2	Beat 3	Beat 4	Beat ...	Beat n

AverageTimePoints

	1
1	SIC
2	AVO maximum
3	AVO minimum
4	Maximum dist.
5	Inflection point
6	AVC minimum
7	AVC maximum
8	DN
9	PR
	Averages

StdDevPoints

	1
1	SIC
2	AVO maximum
3	AVO minimum
4	Maximum dist.
5	Inflection point
6	AVC minimum
7	AVC maximum
8	DN
9	PR
	Standard deviations

These changes ment the new version could not simply be dropped in place of the old one. Users had to change their routines accordingly, to make sure the new information was used correctly.

Example of found Inflection Points

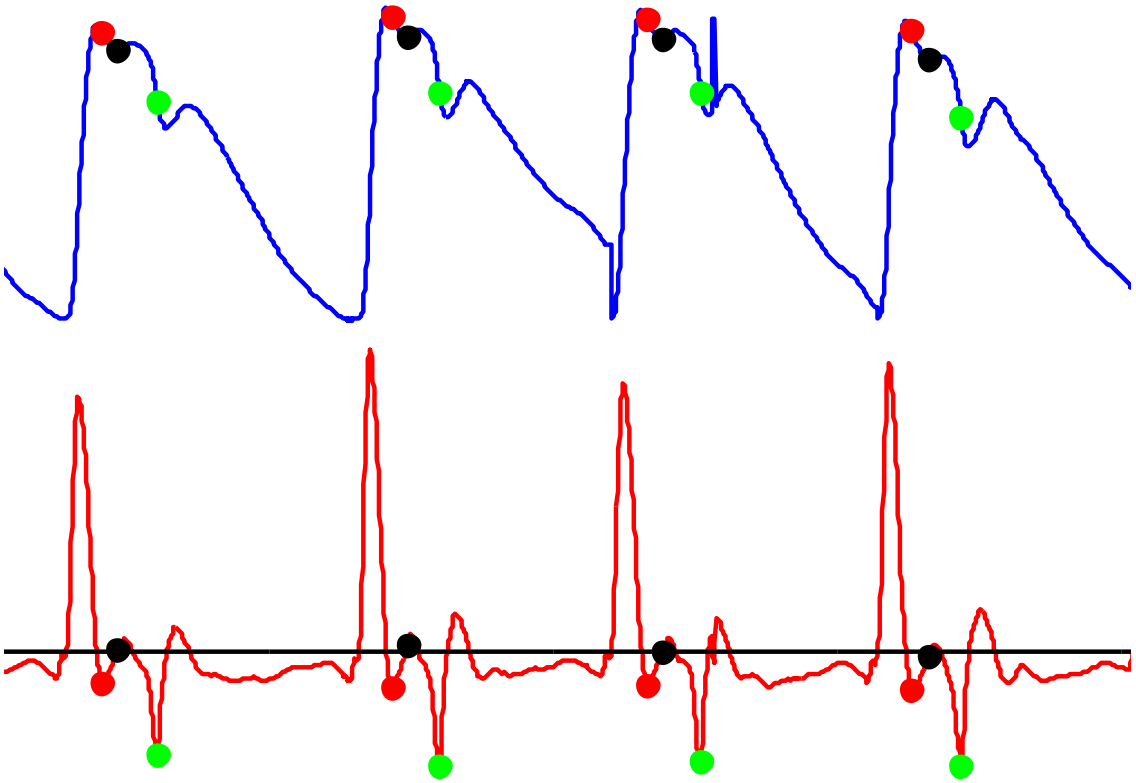


Figure 20: DSP signals with Inflection Points

This graphic plot shows a Distension signal and Velocity signal, and the work that has been done by the new TPE routine.

Red points indicate the AVO minimum
Green points indicate the AVC minimum
Black points show the new Inflection Point

The new routine was successful in finding Inflection Points in 26 of 28 valid test files.

Chapter 5 - ECG validation and correlation

Introduction

Normally the points extracted by the TPE routine are relative to a start point, influenced by the place and frequency of Velocity peaks in the signal. This point is called the TPE beat start. If the ECG signal in a file is valid, the time points that have been extracted by TPE have to be adjusted, to be relative to the R-peak in the ECG signal. The temporal relationship between these points and the ECG R-peak can give valuable physiological information.

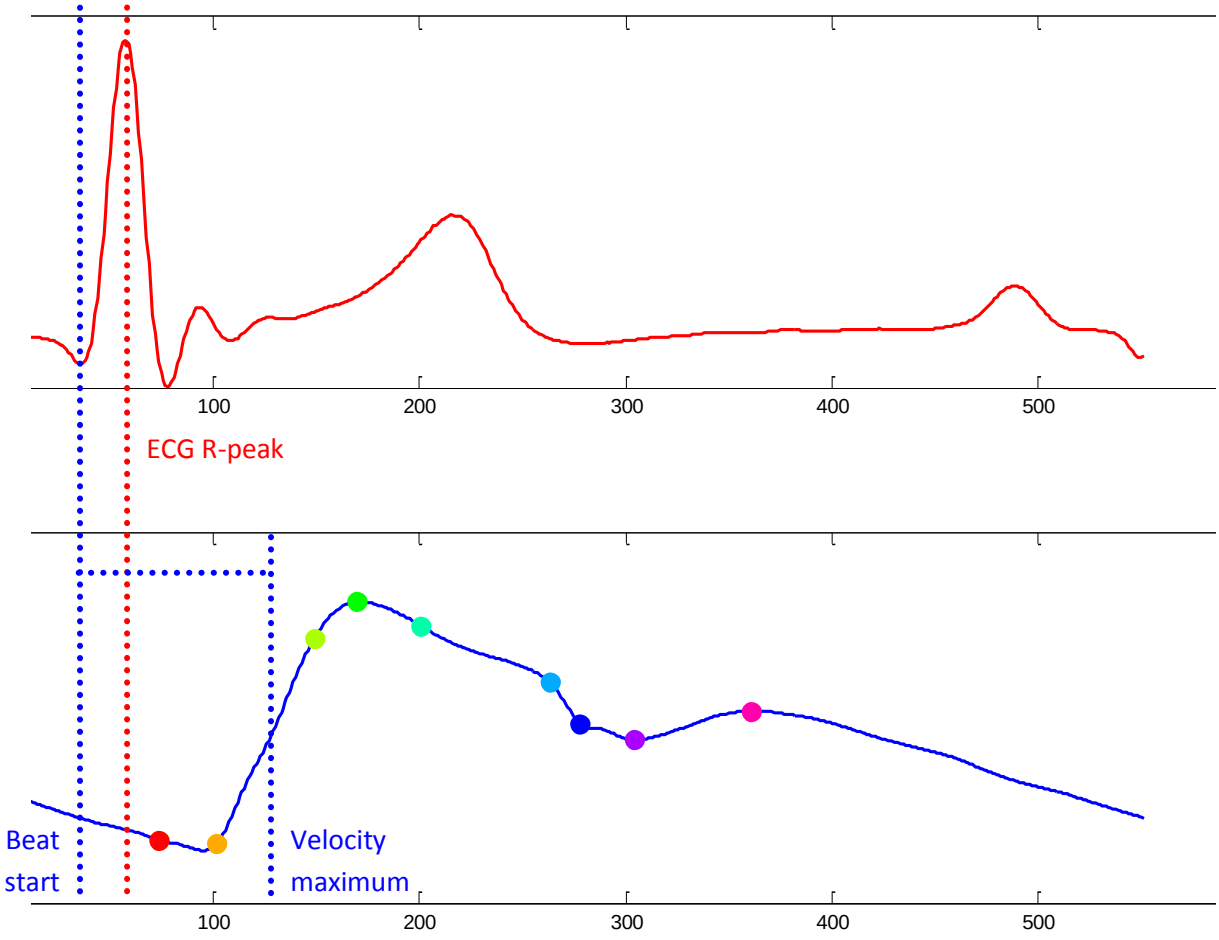


Figure 21: ECG and Distension beats <Can you indicate name of the time points>

The colored dots are the nine time points extracted by TPE. The sample time of these points is given relative to the beat start. To get the absolute value, counting from the ECG R-peak, the points have to undergo the following shift:

ECG shift = ECG R-peak – TPE beat start

Absolute sample point = TPE sample point – ECG shift

To facilitate the above goal, which was set for my internship, there should be a programme able to automatically determine if an ECG signal is present in a measurement's file, and if it is really a valid ECG signal. If the ECG signal is valid, then the programme should shift the time points according to the above procedure. This amounts to the process schematic in appendix 1.

ECG validation – TPE preprocessing

The most important step in the above procedure is: checking the validity of the ECG signal. Not falsely reporting an ECG signal as valid is important, but rules shouldn't be so strict that the smallest deviance should yield an invalid flag.

When developing, I used DSP log files from many sources, and came up with three routines, that are subsequently processed in two big steps. The first two passes are performed in step 1, the third in step 2. If step 1 fails, the validation in step 2 is tried. If any routine declares the ECG is valid, the programme jumps to step 3, for physiological validation.

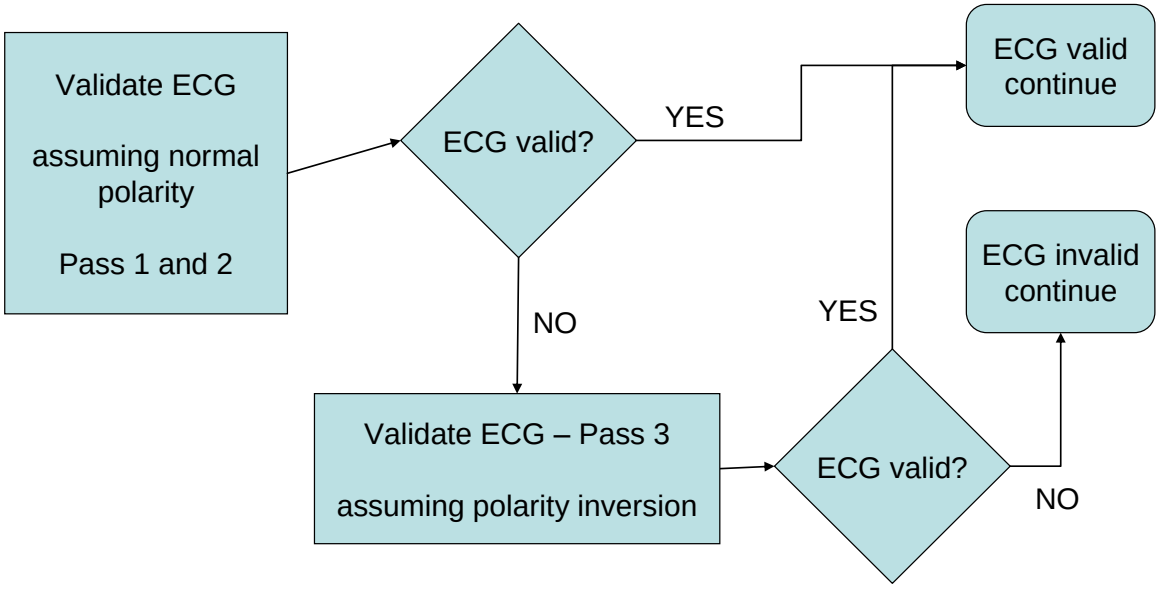


Figure 22: ECG validation flow chart

This plan is necessary, because of the diversity of ECG signal recordings. Most ECG signals in the log files are quite readable and recognisable, or completely useless, with little examples in between.

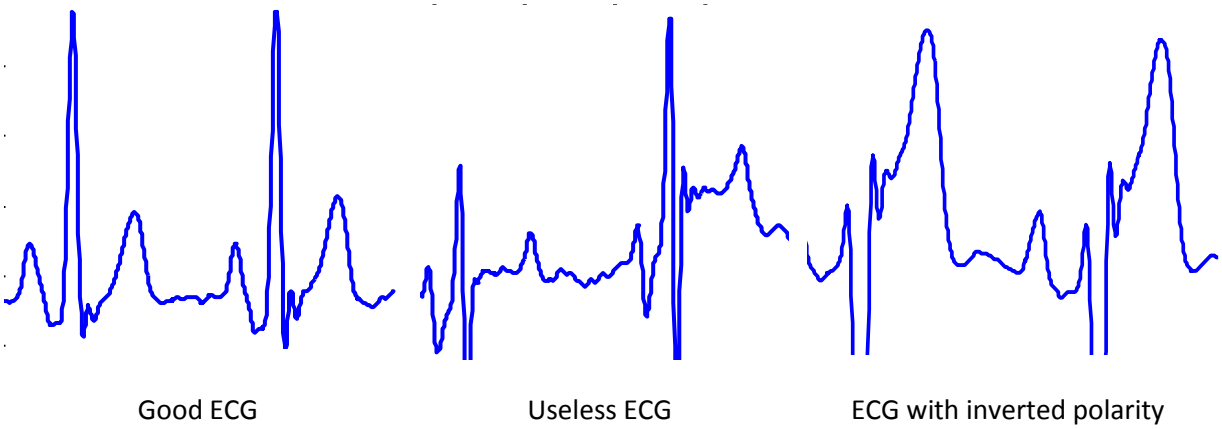


Figure 23: 3 examples of ECG signals

An example of a good file is shown on the left, the middle shows a completely useless file. On the right there is a special case, in which the ECG measurement leads have not been connected properly, and the signal has been inverted. The R-peak that should have been high, has now clipped below

zero. This occurs quite often and the need to recover ECG peaks from files with this problem is quite high.

Step 1: Normal ECG polarity

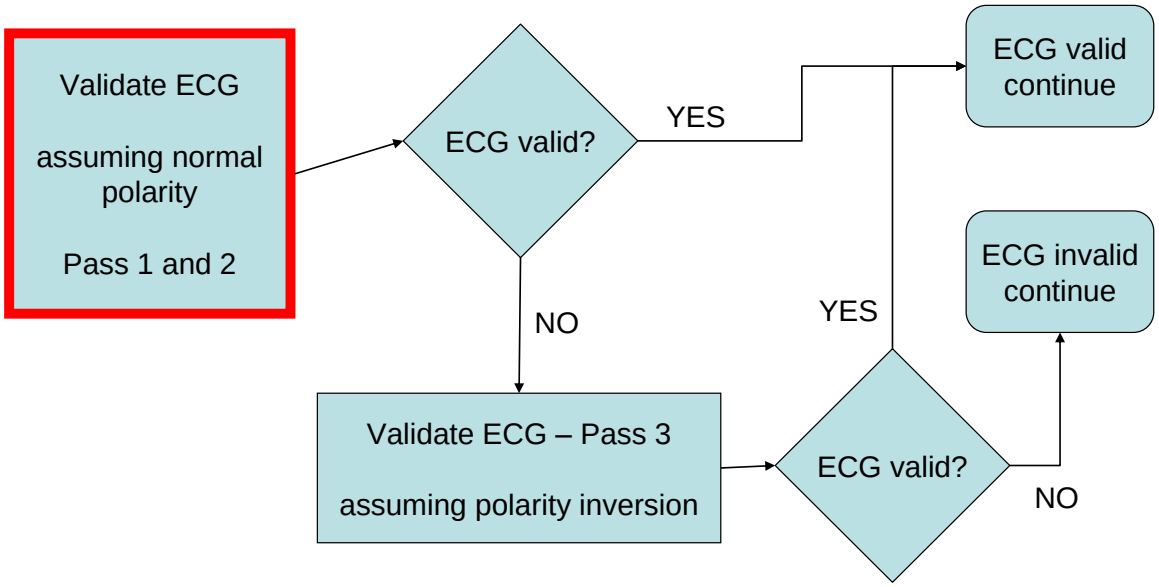


Figure 24: ECG validation flow chart

Pass 1: ECG with R-peak relatively strong to T-wave

The ECG validation routine was originally written to approve these files. The most important property is that the R-peaks are by far the highest peaks in the signal. This part of the programme proceeds as follows:

- The programme gets the ECG peaks by applying a constant search threshold, at 0.7 * the maximum signal amplitude, and finding the maximums of the ranges that are above this threshold. This is also done for the Velocity signal.

- Then, it makes sure the file's peaks are in the right order, and the file starts with an ECG peak, and ends with a Velocity peak. If needed, it removes unwanted peaks at the beginning or the end of the signal.
  The right order of peaks is as follows:
  ECG – Velocity – ECG – … – Velocity – ECG – Velocity

- At this point the programme checks if the number of ECG peaks equals the number of Velocity peaks, and there are 3 or more sets of peaks. If not, the programme stops

- The final check of this stage:
  Every Velocity peak should be closer to its previous ECG peak than to its following ECG peak. If all this checks out, the ECG is classified valid, pass 2 and step 2 are skipped, and the programme goes on to step 3.

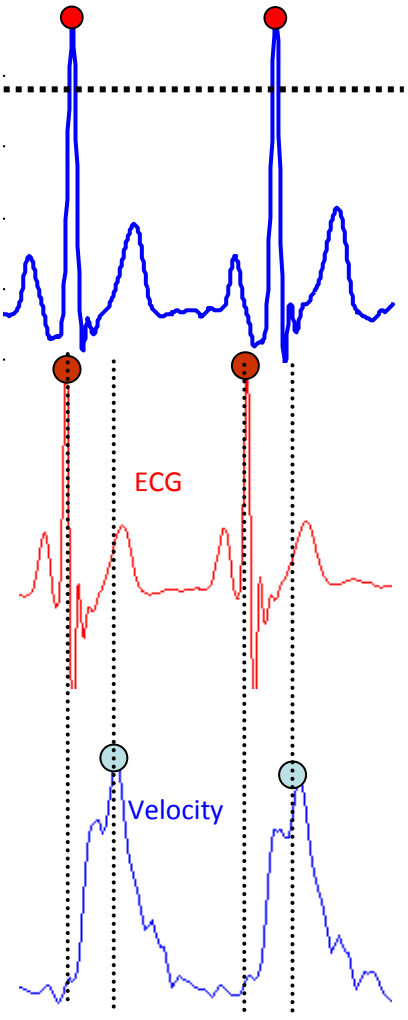


Figure 25: Pass 1

Pass 2: ECG with smaller difference between R-peak and T-wave

In some cases, the T-wave maximum is closer to the R-peak maximum, and the peak detection in Pass 1 finds the double amount of ECG peaks. To make sure we can still use these files, a way to detect this situation was added. If pass 1 reports the ECG is invalid, this method is tried.

The way it works:

- It searches the peaks in a file's ECG and Velocity signals
- Then it tries to match them, according to the following rules:
  - Every Velocity peak should have an ECG peak shortly before and after, within given ranges
  - The before peak should be higher than the after peak
  - The base of the after peak should be wider than the base of the before peak

If this method reports the ECG signal as valid, Step 1 reports success, and the programme skips Step 2 to go straight to Step 3. If not, the programme will continue to Step 2 (which can be considered pass 3).

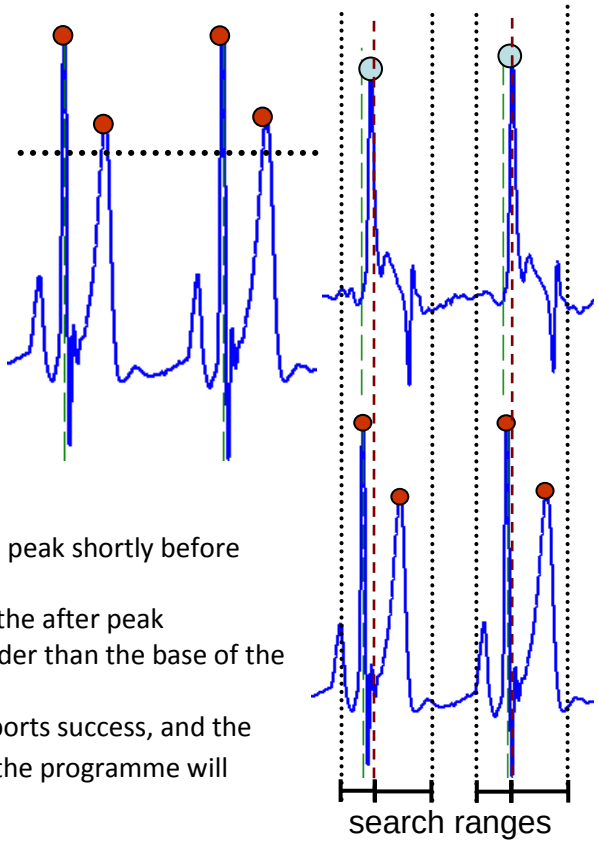


Figure 26: Pass 2 methods

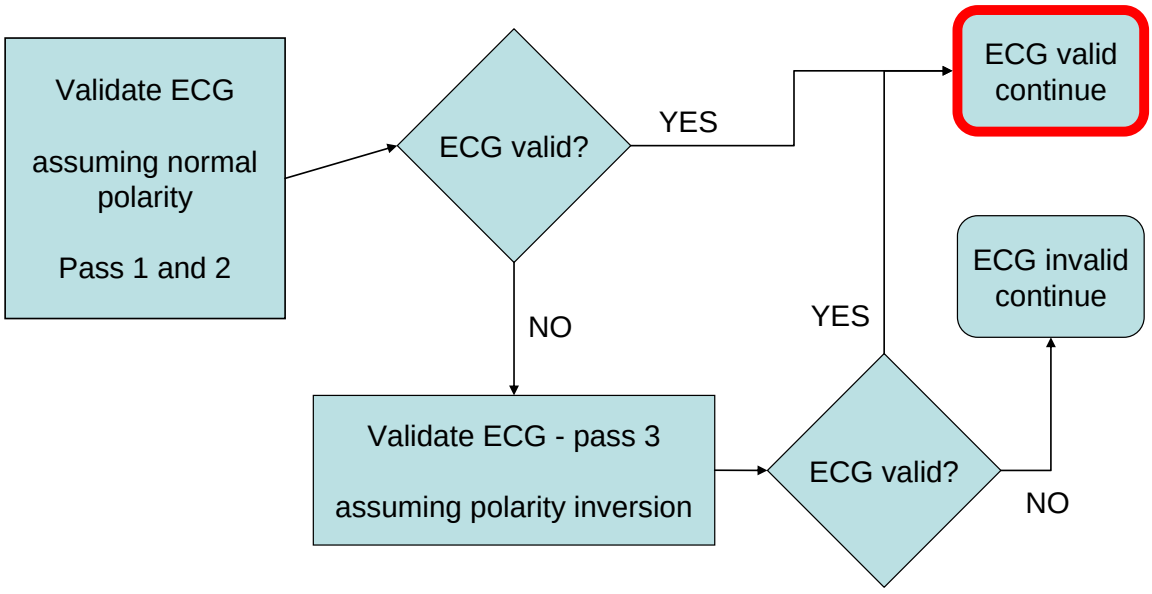
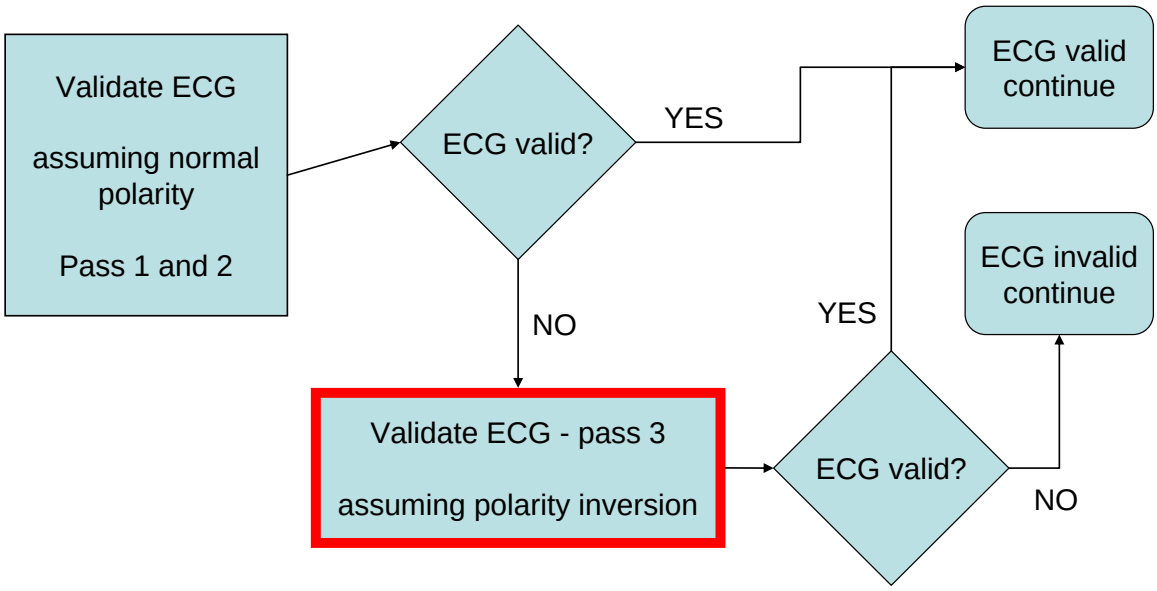


Figure 27: ECG validation flow chart

Step 2: Inverted polarity

As written before, if Step 1 is unsuccessful, Step 2 is tried.



This part of the process is specialised in recognizing ECG signals that have been inverted during the recording. This problem showed up in many of the test sets and is caused by the ultrasound operator not connecting the leads properly. The result is an ECG signal that looks like this, upside down from the normal situation:

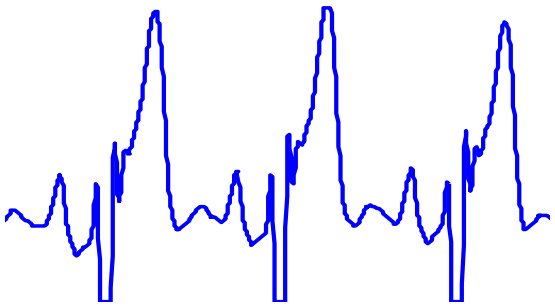


Figure 28: Inverted ECG signal

Here, the signal clips to zero at the point where the R-peak should have been, and the S-dip shows up as a peak. Fortunately, I could use this pattern for detection.

At first the programme searches all Velocity peaks and ECG zero ranges. In every zero range, the middle is taken as an estimated peak.

Next, all estimated ECG peaks are verified using the following rules:

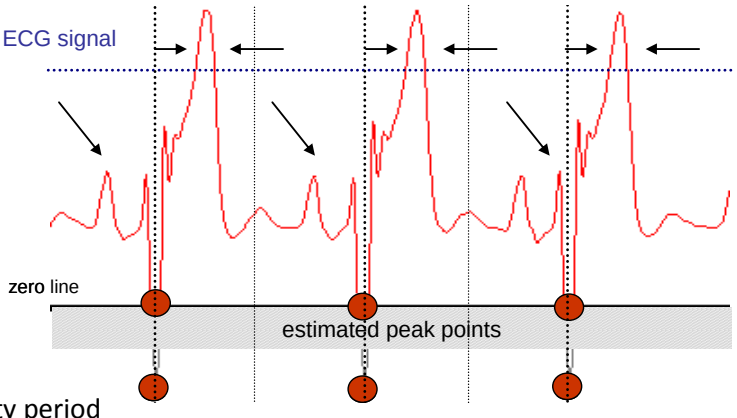


Figure 29: finding peaks

- A threshold is set at 0.7 * the maximum value of the ECG signal.
- ECG signal must go above the threshold within half of the smallest Velocity period **after** the projected peak
- The signal must also stay below the threshold during half smallest Velocity period **before** the projected ECG peak

The ECG peaks that have been found valid are matched to the peaks in the Velocity signal. Velocity peaks without a matching ECG peak are discarded.

Step 3: Physiological checks

If at any point the ECG is marked valid, this is only based on the order of valid peaks. There are more criteria we have to account for, to make sure only ECG signals that are really valid are processed as such. To make sure this happens, at the end the programme performs several checks, related to the regularity and speed of the peaks. They are called physiological checks, because the regularity and speed of the peaks are related to the cardiovascular properties of the human body.

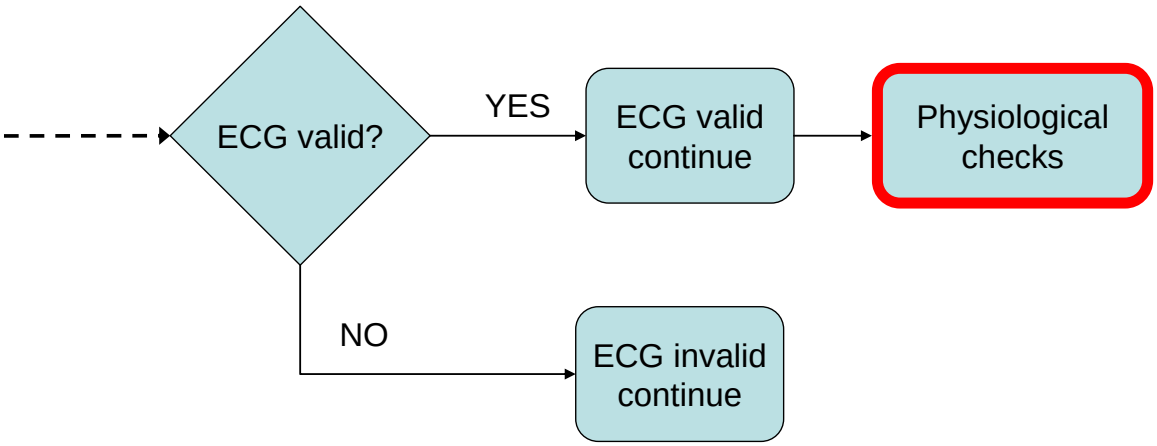


Figure 30: End handling flowchart

The ECG signal and its peaks can only be used if the programme passes these checks as well.

Criteria:

- The programme calculates the average ECG frequency and checks the following:
  Average ECG frequency should be between 20 and 300 bpm (between 1/3 and 5 Hz)

- The programme calculates the median ECG period and checks the following:
  Every ECG period should stay within ±20% of the given median

- The programme calculates the median of the delays from ECG peaks to their respective Velocity peak
  Every delay between an ECG peak and its corresponding Velocity peak should stay within ±10% of this median value

If these criteria are met, the ECG signal has been fully validated, and the programme can proceed with TPE, and correlating the results to the found ECG peaks.

If these criteria are not met, or the ECG has been found invalid in any other stage, the programme gives back the original Velocity peaks and checks them for regularity like so:

- The programme calculates the median Velocity period and checks the following:
  Every Velocity period should stay within ±20% of the given median

If this is criterion is not met, the Velocity signal is unusable as well, and the programme ensures this file is skipped in further processing.

Summary on ECG validation

In short the following happens:

1. The routine evaluates the order of ECG peaks based on the assumption it has normal (non-inverted) polarity
   a. First it tries to validate the peaks for the condition where R-peak is significantly stronger than the T-wave
   b. If the above fails, it will do a second pass, where it expects that the T-wave is closer in height to the R-peak, but still a bit lower

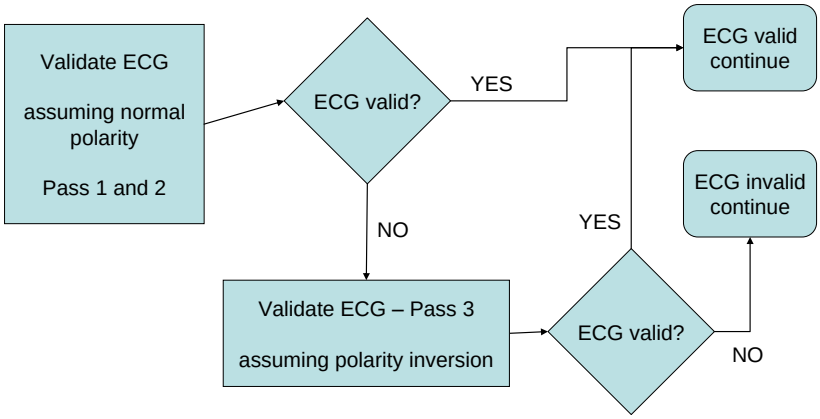


Figure 31: ECG validation flow chart

2. If both tries above yield no result, it tries to find an inverted ECG signal like I have seen in many files.
3. If at step 1 or 2 the ECG has been found valid, the programme checks if the found peaks are physiologically valid. This means the following:
   a. The ECG peaks should have regular intervals
   b. The delay between ECG and Velocity peaks should be regular as well
   c. The heart rate should stay between 20 and 300 bpm

Conclusions on ECG validation

All in all, the development of a proper programme for the validation of ECG signals took more time than I expected. This is due to the fact that I had to develop different algorithms with different approaches. The ECG signals that have been recorded in the real world have different forms and properties, and as I went along, the number of correct recognitions increased, while the number of false positives went closer to zero. If I had only kept one routine, the programme would only validate the ECG signal in one of every six files presented. By using multiple detection routines, this has been doubled, to approximately one third of the files. This increase is quite significant, and greatly improves the research possibilities, because there are more valid measurements now.

The whole ECG validation routine has been worked out into one MATLAB routine, called *DSPvalidateECG*. This routine places calls to several other routines, but can be used in other programs, as long as the user takes along all the other routines the programme needs. By use of this routine, the ECG signal can now be validated automatically.

All of this forms the pre-processing stage in the TPE process schematic of appendix 1.

ECG correlation – TPE postprocessing

When the ECG has been found valid, the time points found by TPE have to be modified accordingly. TPE returns time points in the following format. For a better understanding of this schematic see figure 19, which demonstrates the old format, without the Inflection point. This figure shows the new format, with the added Inflection point and standard deviations.

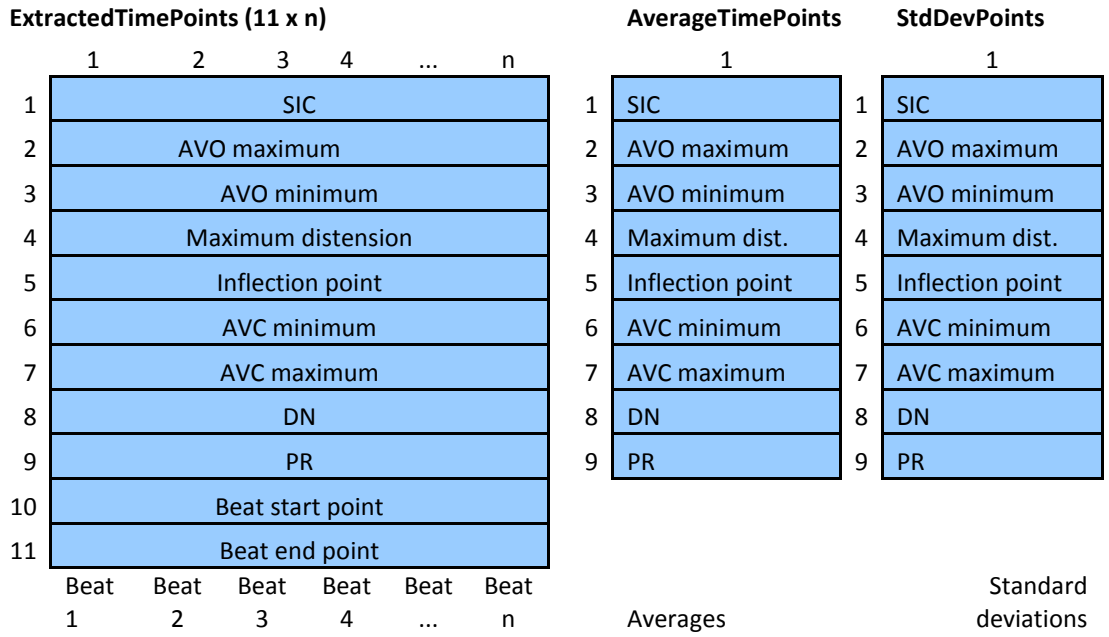


Figure 32: TPE output format

All time points except the beat start point are output in sample times relative to the beat start. This is more convenient for physiological analysis, because the absolute time in a measurement isn't particularly useful. Researchers prefer to know the relative time between points on a per-beat basis. Every beat is considered equal in value, so it is useful if numbers of different beats are usable without correcting offsets, explaining why time points are made relative to the beat starts here. For more detailed information on the time points, see chapter 4.

When researchers want to know the time it takes for blood to flow between the heart's valve and a certain artery, it is quite useful to know the time between the R-peak of the ECG and the other time points. To make this possible, I created a routine that time-relates the found TPE points to the R-peaks in the ECG signal included in the log files. All of this forms the post-processing stage in the TPE process schematic of appendix 1. A demonstration of this correlation is shown on the next page.

Example - shifting of values

Every beat *n* has a start point, that can be found in *ExtractedTimePoints(10,n)*. This start point is an absolute sample value. All other sample values are relative to this start point. The example below shows how the values change when this start point changes to an ECG peak. In the example below, the ECG peak is located at 1226, while the regular TPE beat starts at 1208. Using the below formula, we get the ECG shift, which is 1226 – 1208 = 18.

ECG shift = ECG R-peak – TPE beat start
Absolute sample point = TPE sample point – ECG shift

Firstly the table shows how the new points can be calculated through their absolute values, and the last column shows how to recalculate the relative values, using the ECG shift.

All points except the end point are changed. Keeping the relative end point the same was necessary to keep the beat lengths the same for later processing. This means the absolute end point shifts, but this is insignificant to the results, because it is far beyond all the useful time points.

	Point	Old TPE result	Absolute sample pt	ECG-based start point	Shifted relative pts
1	SIC	26	26 + 1208 = 1234	1234 – 1226 = 8	26 – 18 = 8
2	AVO maximum	43	43 + 1208 = 1251	1251 – 1226 = 25	43 – 18 = 25
3	AVO minimum	71	71 + 1208 = 1279	1279 – 1226 = 53	71 – 18 = 53
4	Maximum distension	107	107 + 1208 = 1315	1315 – 1226 = 89	107 – 18 = 89
5	Inflection point	118	118 + 1208 =1326	1326 – 1226 = 100	118 – 18 = 100
6	AVC minimum	171	171 + 1208 = 1379	1379 – 1226 = 153	171 – 18 = 153
7	AVC maximum	183	183 + 1208 = 1391	1391 – 1226 = 165	183 – 18 = 165
8	DN	193	193 + 1208 = 1401	1401 – 1226 = 175	193 – 18 = 175
9	PR	235	235 + 1208 = 1443	1443 – 1226 = 217	235 – 18 = 217
10	Beat start point	1208	1208	1226 (1226 – 1208 = **18**)	-
11	Beat end point	402	-	402	-

Figure 33: Example of TPE values shifted for ECG peaks

This procedure will of course be performed on every beat of the TPE output and their corresponding ECG peaks. After that, the average time points, their standard deviations and the average beat waveforms will be calculated again, replacing the old values. All of this of course only happens when the ECG has been recognized as valid. Else everything is left as it is.

When the ECG is valid, the points have been adjusted to the ECG and the file's sample rate is known, one can also calculate the real time between the ECG peak and the time points. This information can be very useful in physiological research.

Chapter 6 - Batch processing of DSP data sets

In research conditions, the data gathered per file by the TPE routine and ECG correlation programme is most useful when combined with data sets from other files. These files will mostly be from the same patient, or from patients that are in the same group of research.

Since 2006, MUMC+ and other universities and hospitals have gathered large sets of vascular ultrasound data, combined with ECG signals. These sets all have to undergo the processing outlined in chapter 4 and 5. The sets consist of thousands of separate measurements. There is no practical way to process these files one by one, so a way to process them automatically is called for.

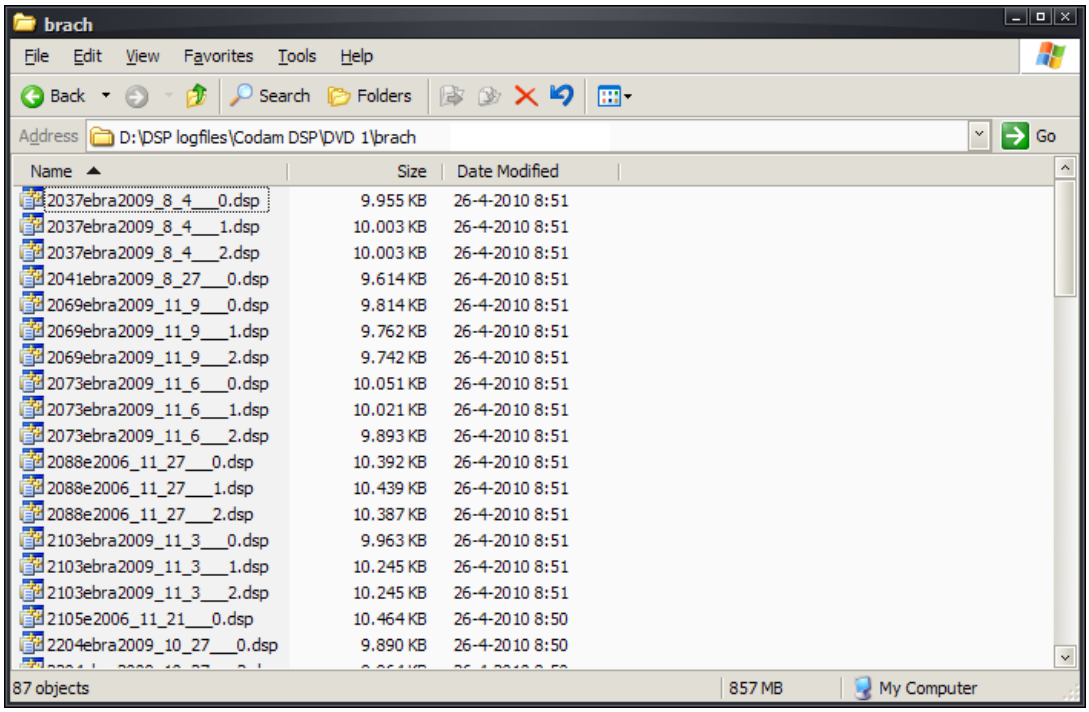


Figure 34: Directory full of DSP log files

To facilitate this need in the field, I created several programmes in MATLAB, with a philosophy behind it. Peter Brands and I agreed on the following model for use:

1. First, users will want to go through their data set and see which files are useful (without corruption, bad measurements, glitches etcetera)
   This process would have to be simple and fast, 1 user action per file was our goal. This way, users could filter their sets and create subsets with only valid data.

2. Whenever users had sorted out their data and created subsets with usable files, they would want to run this through an automatically processing programme, possibly leaving this to run overnight. Output should go to a file, not to the screen.

3. Users might want to check the results of the batch processor in step 2. Therefore, there would have to be a programme that does the same as the batch processor, but to one file. The output of this would preferably be on screen instead of in files.

This idea was worked out as follows, in three MATLAB programmes, one for each step. Each programme is described in functionality as well as how to use it.

Step 1 - ART.LAB Waveform viewer (DSPbatchEvaluateLogFiles)

This programme opens all DSP log files in a folder, and all its subfolders. Then, one file at a time, it analyses the signals present. It checks the validity of the Velocity signal, which in turn determines if a file is valid as a whole, and the ECG signal, which is optional, but beneficiary for the analysis by AbsoluteTPE. The results of the analysis are printed to screen, with the signals themselves right next to it.

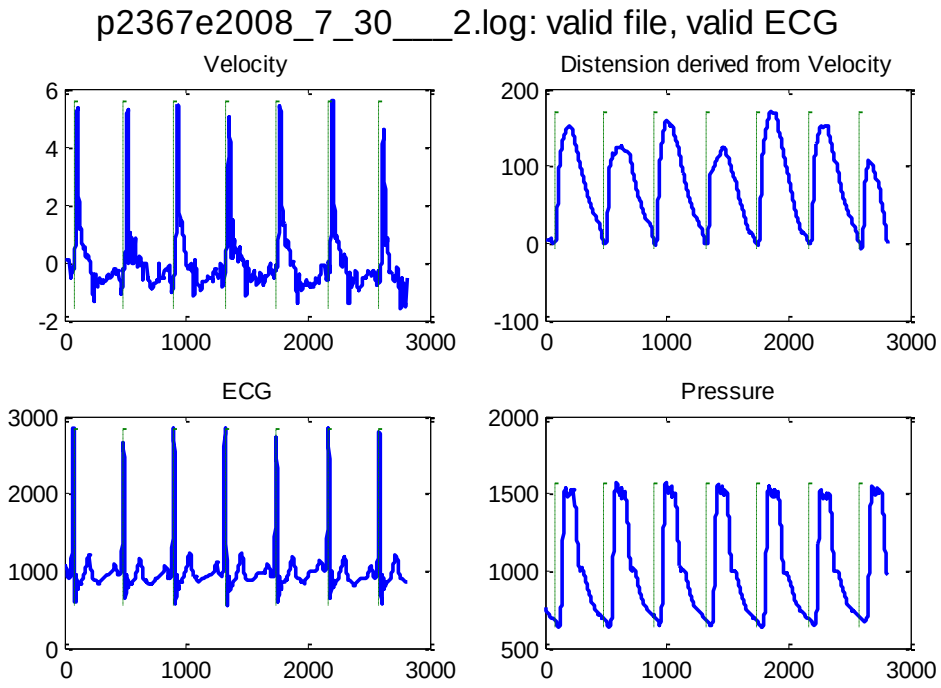


Figure 35: Example of graphs displayed by Waveform viewer programme

The combination of signals and analysis results gives users the possibility to verify if the file was correctly analysed. If the programme says a file is invalid, the user can see why. On the other side, if the file has been marked valid valid, the user can verify this to make sure the programme was correct.

Analysis results on every file are written to an output file as well, so users can get back to the results after the programme has ended. At the end, the programme shows the contents of several counters.

STATISTICS:		Number of files in directory…
Files processed:	11	…that could be read successfully (11)
Files with valid ECG:	5	…of which the ECG could be recognized (5)
Files with valid Velocity:	7	…of which the Velocity signal could be recognized (7)
Total number of files:	12	…that have been attempted to load (12)

Based on all the results returned by the programme, the user should be able to filter the data in such a way that almost all of his data will be successfully processed by the AbsoluteTPE batch processor.

How to use the ART.LAB Waveform viewer

The user can start the ART.LAB Waveform viewer in two ways, either from MATLAB, by calling the main script, or by running the appropriate executable file.

In MATLAB, the user should navigate to the folder with DSPbatchEvaluateLogFiles.m and the other .m-file scripts in it, and type the following command at the MATLAB console:

```
>> DSPbatchEvaluateLogFiles();
```

When not using MATLAB, the user should open the folder with DSPbatchEvaluateLogFiles.exe, and then launch DSPbatchEvaluateLogFiles.exe.

Regardless of how the programme is started, it will ask the user for a folder with measurements it has to process.

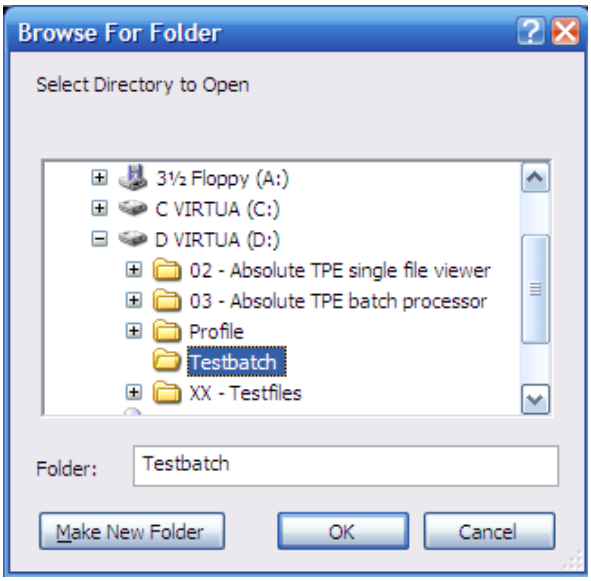


Figure 36: Folder selection dialog

After selecting a folder, it will show the user the analysis on all the files in the folder, like in figure 35.

At the end of the analysis, a file called *DSPbatchEvaluateLogFiles-result.csv* will be created. This file can be found in the folder that was chosen at the beginning. If the file cannot be written to this folder , the user will be asked where to put the file at the beginning of the programme.

```
File 2 of 7: .\AS_fbmCCA2009_3_05___4.log
DSPvalidateECG: Trying to validate ECG
(pass 1 of 3)
DSPvalidateECG1 INFO: Average ECG
frequency: 67.94 Hz
DSPvalidateECG1 INFO: ECG valid
DSPvalidateECG RESULT: ECG valid


RESULT: valid file with valid ECG
```

Figure 37: Example of text in output file

Step 2 - AbsoluteTPE batch processor

The inner workings

When the data sets have been sifted through, and only the valid files are left, the user is now at the point where he or she wants to let the data set be processed according to the processes specified in chapters 4 and 5. This means the programme performs the following on each file:

- Check validity of the Velocity signal
- See if the ECG signal is valid (chapter 5)
- Perform time point extraction (TPE - chapter 4) on the data from the file
- When the ECG is valid, the programme correlates the TPE results to the ECG peaks

The above things can only be done on a per file basis. But a folder with data files will most of the times be part of a larger data set. According to Peter, users wanted to average the results over all files in this data set into global statistics. Averages and standard deviations of time points per file are available as seen in chapters 4 and 5. But to make sure they can be averaged over all files I had to attend to the following things:

- **Beat start points**
  The previously existing TPE routine had a quite complicated way of setting the start point for its analysis. This meant that the start point was different for every file, mostly determined by the heart beat frequency. In the programme I developed, the ECG R-peak is used as a temporal reference. As mentioned above, the TPE programme itself correlates the time points to the R-peak (also see chapter 5). When the time points are averaged, they all need the same reference point. In this case the R-peak of the ECG. So I made a mechanism to automatically discount files with non-valid ECG signals from the averaging process.
- **Time base**
  All time points, including averages, are given in sample intervals. The sample rate per file is a fixed number, so every sample point represents a certain time interval. The sample rate itself can be different for every file, leading to different time bases. Using the sample rate specified in the file, I was able to convert the time points to milliseconds.

With these measures, only useful files are used in the global statistics, and they are used with the right time base, so that the global averages on a folder were correct.

To obtain the average data I made the following calculations:

$$GlobalAvgTimePnt = \overline{FileAvgTimePnt}$$

The global average value of any time point is equal to the mean of the averages per file:

$$\sigma(TotalAvgTimePnt) = \sqrt{\overline{(\sigma(FileAvgTimePnt))^2}}$$

The standard deviation cannot simply be averaged, only the variance can. To determine the variance, the standard deviation of any point has to be squared. This variance can be averaged over all files. The result is the average variance, which in turn can be made into the standard deviation by extracting the square root again.

The user experience

Given that this process may take a while, this process should not require any user interaction once it has started. The user can then leave the computer alone, for several hours if necessary.

In MATLAB, the user should navigate to the folder with AbsoluteTPE.m and the other .m-file scripts in it, and type the following command at the MATLAB console:

```
>> AbsoluteTPE();
```

When not using MATLAB, the user should open the folder with AbsoluteTPE.exe, and then launch AbsoluteTPE.exe.

Regardless of how the programme is started, it will ask for the folder with measurements it has to process.

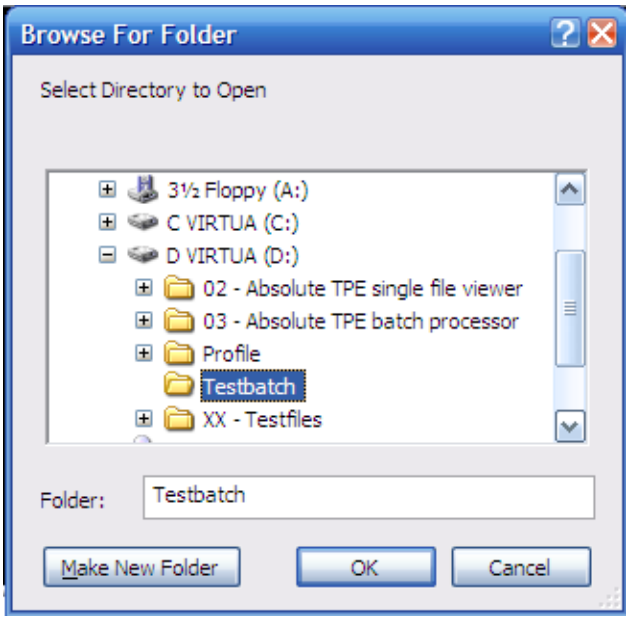


Figure 38: Folder open dialog

If the folder cannot be written to, it will also ask the user where it has to place the output files. Else, the output files will be placed in the same folder. From then on, the processing starts and the programme outputs only text.

```
...
DSPvalidateECG1 INFO: Average ECG frequency: 92.34 bpm
DSPvalidateECG1 INFO: ECG valid
DSPvalidateECG RESULT: ECG valid
Running TPE...
ECG signal valid, running post-processing...

File 7 of 7
Reading...
Running pre-processing...
DSPvalidateECG: Trying to validate ECG (pass 1 of 3)
DSPvalidateECG1 WARNING: Last ECG peak after last Velocity peak, discarded last
ECG peak
DSPvalidateECG1 INFO: Unequal numbers of peaks after first stage, invalid ECG
DSPvalidateECG: Trying to validate ECG (pass 2 of 3)
DSPvalidateECG2 INFO: Average ECG frequency: 58.63 bpm
DSPvalidateInvertedECG RESULT: ECG valid
DSPvalidateECG RESULT: ECG valid
Running TPE...
ECG signal valid, running post-processing...

END of AbsoluteTPE output, press any key to close...
```

Figure 39: Example of console output by batch processor

When the processing is done, the user can access the results through several result files, which are in the Comma Seperated Values format (CSV). The files can easily be opened in any spreadsheet programme and are named like below.

- AbsoluteTPE-verbose.csv
  Per measurement file:
  - All time points, distension amplitudes, averages (time points in samples and milliseconds).
  - On all files: Global statistics - time points in milliseconds averaged over all files
- AbsoluteTPE-physiological.csv
  - Per measurement file: Average time points (in milliseconds) and distension amplitudes
  - On all files: Global statistics - time points in milliseconds averaged over all files
- AbsoluteTPE-rows.csv
  - Per measurement file: Average time points (in milliseconds) and distension amplitudes. The results of one file are in one row.
  - Global statistics are not included

Examples of output files can be found in appendices 2, 3 and 4.

Step 3 - AbsoluteTPE single file viewer

After the processing has been done, and the user has evaluated the results from the whole set of files, he or she may want to get more details on the results of a specific file. To make this possible, I created a programme that does everything the previously mentioned batch processor does, but only for a single file. This of course means there are no global statistics, as these were meant for multiple files in one folder.

In MATLAB, the user should navigate to the folder with AbsoluteTPE.m and the other .m-file scripts in it, and type the following command at the MATLAB console:

```
>> AbsoluteTPEviewer();
```

When not using MATLAB, the user should open the folder with *AbsoluteTPEviewer.exe*, and then launch *AbsoluteTPEviewer.exe*.

When started, either through the executable or MATLAB, the programme presents itself as follows:

It asks for the file that should be opened

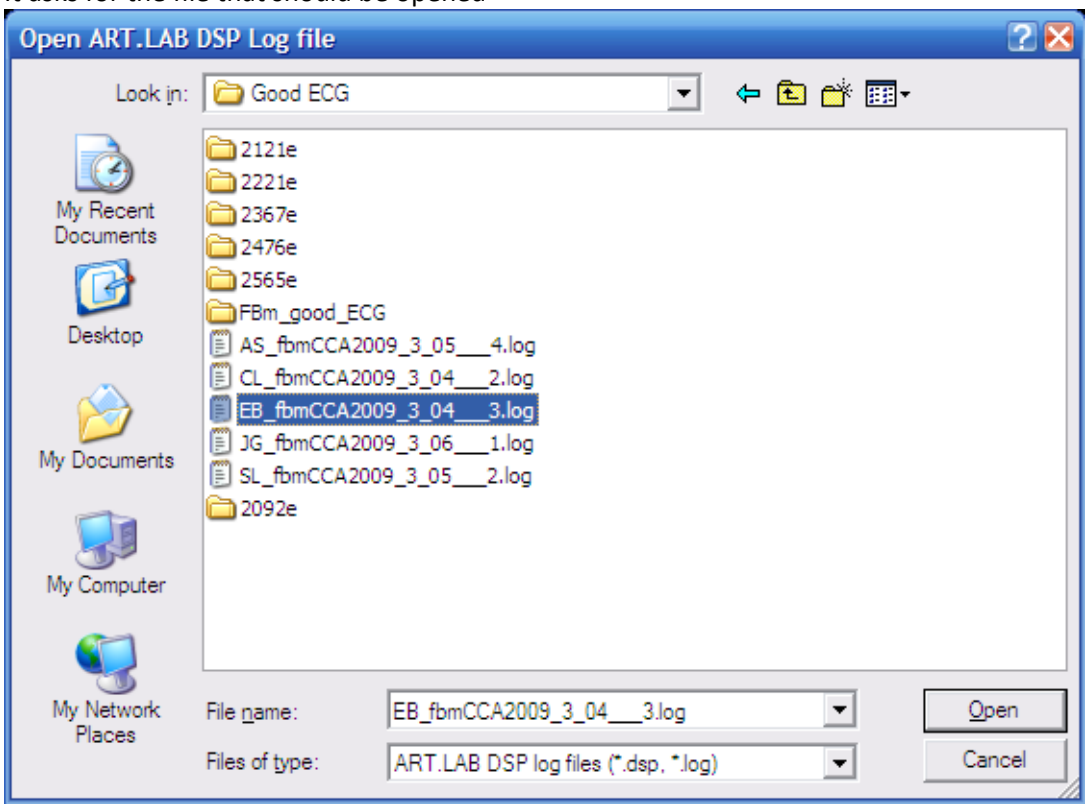


Figure 40: File open dialog

From then on it processes the file like the batch processor, and shows the results on screen. All time points are shown in the console.

Example of console output

```
AbsoluteTPEviewer output below
File:
EB_fbmCCA2009_3_04___3.log
Reading...
Running pre-processing...
DSPvalidateECG: Trying to validate ECG (pass 1 of 3)
DSPvalidateECG1 INFO: Equal numbers of peaks when entering second stage
DSPvalidateECG1 INFO: Average ECG frequency: 28.65 bpm
DSPvalidateECG1 INFO: ECG valid
DSPvalidateECG RESULT: ECG valid
Plotting the four signals..
Valid file with valid ECG
Running TPE...
ECG signal valid, running post-processing...
ECG valid, time points counting from R-tops
Framerate: 242.57 fps

TIME POINTS
Beats                1        2        3        4        5
SIC                 15       18       15       14       14
AVO max             38       46       35       37       38
AVO min             87       93       84       91       86
MaxDist             99      114       99      107       99
IP                 126      145      112      125      115
AVC min            183      207      189      197      180
AVC max            193      222      202      223      196
DN                 218      248      227      250      219
PR                 238      305      280      304      283
Start               93      556     1066     1584     2119
End                471      471      471      471      471

AVERAGE TIME POINTS
Point          Avg smp   StdDev Relative   Avg ms   StdDev Relative
SIC                 15     1.64     0.11    61.84     6.77     0.11
AVO max             39     4.21     0.11   160.78    17.34     0.11
AVO min             88     3.70     0.04   362.78    15.26     0.04
Max dist           104     6.77     0.07   428.74    27.90     0.07
IP                 125    12.93     0.10   515.31    53.32     0.10
AVC min            191    10.96     0.06   787.40    45.20     0.06
AVC max            207    14.34     0.07   853.36    59.13     0.07
DN                 232    15.57     0.07   956.42    64.17     0.07
PR                 282    27.18     0.10  1162.54   112.03     0.10

DISTENSION AMPLITUDES
Beats                1        2        3        4        5
SIC            -26.41   -33.80   -33.14   -27.32   -25.03
AVO max        -19.92   -33.51   -32.99   -21.16    -5.41
AVO min        569.99   538.05   630.27   614.75   584.26
MaxDist        603.50   630.74   689.49   675.63   628.57
IP             523.47   559.98   662.41   635.45   585.53
AVC min        354.99   400.99   462.48   476.75   379.96
AVC max        283.50   287.73   362.43   320.08   264.03
DN             265.89   246.68   313.03   287.09   229.28
PR             305.29   327.57   382.76   366.96   341.21
Start           -1.43    -1.24    -2.12    -2.16    -1.77
End               -14       70       59       93       67

AVERAGE DISTENSION AMPLITUDES
```

```
Point          Avg smp    StdDev Relative
SIC             -29.14      4.05    -0.14
AVO max         -22.60     11.53    -0.51
AVO min         587.46     36.52     0.06
Max dist        645.59     35.75     0.06
IP              593.37     56.12     0.09
AVC min         415.03     52.66     0.13
AVC max         303.55     38.58     0.13
DN              268.39     32.98     0.12
PR              344.76     30.83     0.09

END of AbsoluteTPEviewer output
Press any key to close. This will close all figures.
```

Figure 41: Console output of ART.LAB waveform viewer

Also it shows the following 3 graphic plots on screen

- One that closely resembles the ART.LAB waveform viewer, to evaluate all signals

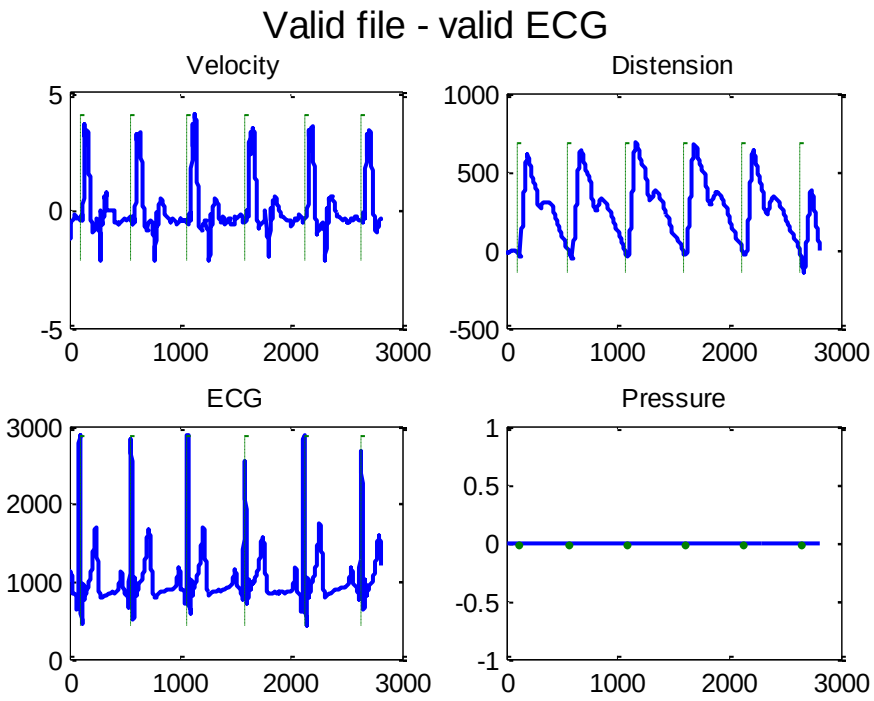


Figure 42: First view

- Another showing the entire Velocity and Distension waveforms, with all extracted time points

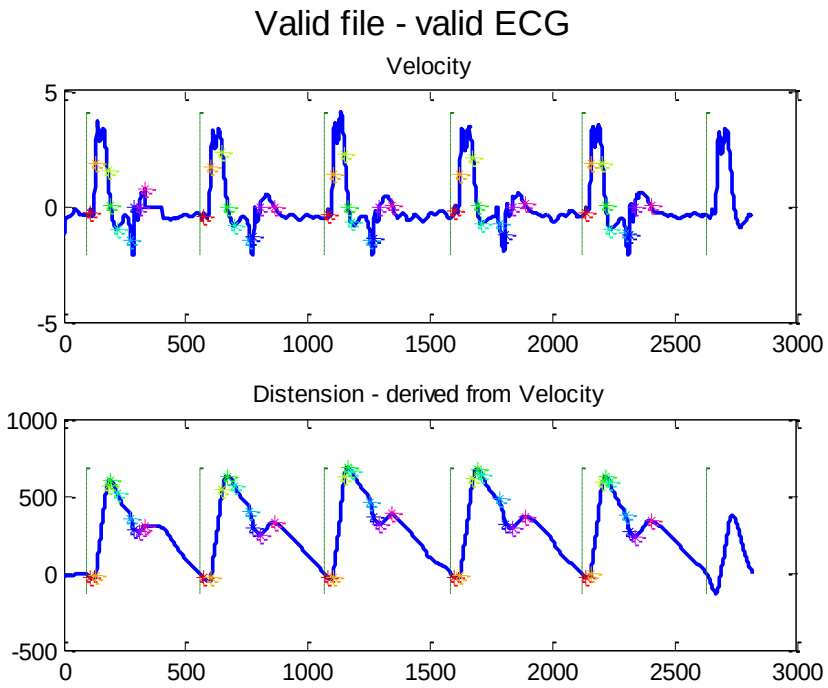


Figure 43: Second view

- And another showing the average Velocity beat and Distension beat, with the average time points

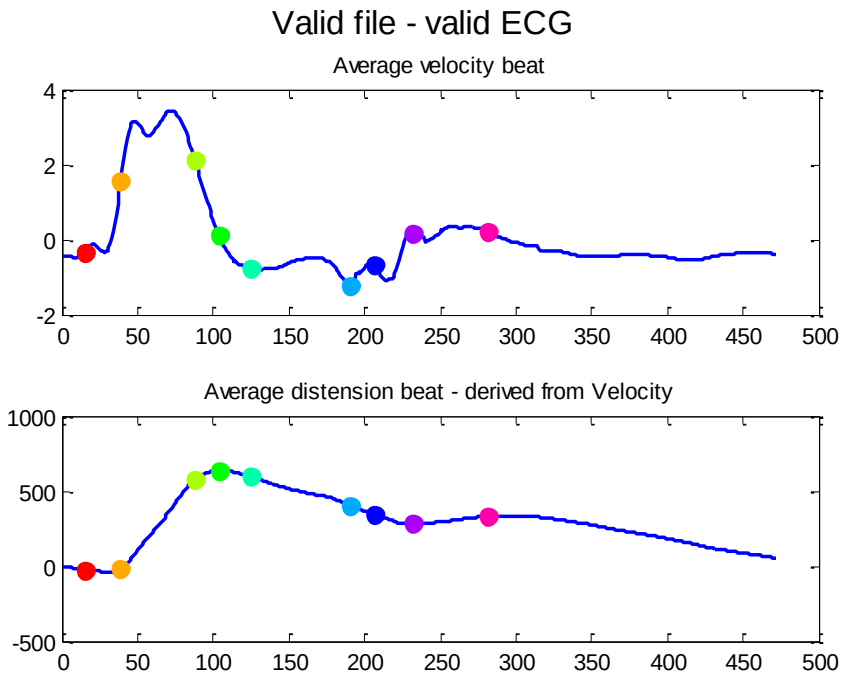


Figure 44: Third view

With these outputs, any file and the analysis on it can be easily verified and by viewing the different waveforms the results can be explained.

Programmes as executable

The above programmes have been developed in MATLAB, specifically in version 2007b. Everyone with this version or a newer version of MATLAB can run the programmes. Users without MATLAB were unable to run any of these programmes. I researched the possibility of compiling the programmes into an executable format and stumbled across the MATLAB Compiler. This allowed the compilation of MATLAB programmes (in .m-files) into executables. To run these programmes on systems without MATLAB, users have to install the MATLAB Client Runtime version 7.1. Fortunately, this version was included with MATLAB 2007b, and may freely be redistributed. After a day of research, I was able to compile my applications into an executable format. I tested them on a freshly installed computer without MATLAB. I installed the runtime, and my programs functioned like they originally did in MATLAB, just without the MATLAB GUI around it.

Summary

My programmes made the following workflow possible:

Step Program name

1. ART.LAB Waveform viewer
   Lets user sort through large data set to establish which measurement files are useful (have a valid wall velocity and ECG signal). This is useful in creating an entirely useful data set

2. AbsoluteTPE batch processor
   The actual processing. Automatically extracts time points from an entire folder of measurement files and use ECG where possible to make time points absolute. Also does global statistics over all measurement files in the folder, and puts all results in text files.

3. AbsoluteTPE file viewer
   Does the processing of the second programme, but only on one file, and outputs to the screen instead of output files. Lets the user check the results of the batch processor.

Conclusion

The programming of the algorithms in chapter 4 and 5 took a lot of time, but putting them to use in the Waveform viewer tool, AbsoluteTPE batch processor and AbsoluteTPE single file viewer also took quite some time.

Eventually I got everything to work as meant, in three programmes: the waveform viewer for data set evaluation, the batch processor for the raw processing and the single file viewer for checking afterwards. Being able to make these programmes MATLAB-independent made this part of my work a total success.

Chapter 7 - Other activities

During my internship I did not spend my time solely developing MATLAB routines. I was involved in several extra activities, all related to ART.LAB research. All of these activities seemed just as useful to me as the development itself, because they were a good example of company work, other than just sitting at my workspace and developing solutions.

The most significant of these activities were:

- Running latency tests on ART.LAB, between the RF data and both reference channel
- Visiting Maastricht's hospital (MUMC+) for service, updates and feedback
- Visiting l'Hôpital Européen Georges Pompidou (HEGP) in Paris to install and demonstrate my software

I will shed some light on the above activities later on in this chapter, as I deem them quite interesting. Next to these activities, I also performed the following tasks, that I will explain more shortly:

Documenting products, procedures and programs made by others

When developing my own products, I also made appropriate documentation for them, but sometimes other documents were needed or pending. I was then asked to document these items. The nature of the documentation was anything from assembly manuals for the production department to theoretical backgrounds of certain algorithms.

Auditing the industrialisation process of ART.LAB scanner setups

In one instance, I had to watch while production employees assembled an ART.LAB scanner using the existing documentation. My job was to see if the instructions were clear, and if not, change the documentation to clear up the misunderstandings.

Assembly and service of ART.LAB systems

During my time a client of Esaote ordered an ART.LAB scanner setup. I was asked to modify the MyLab scanner for this client to accommodate the ART.LAB-pc, and test the setup.

I was also asked to perform service or upgrades on existing ART.LAB systems, for example in the MUMC. More on my activities at MUMC can be found further down.

Figure 45: An ART.LAB ultrasound system assembled by me

Testing new releases of the ART.LAB software

As I went along the development of the ART.LAB software went along too. Whenever a new version was released, I was asked to test the software from the perspective of a user, to find bugs, situations where the software would crash or other problems. I would report these findings in writing, using a semi-standardized test form. Based on these results, Peter Brands would decide if this version would be released to the MUMC+ for further testing and feedback, or to the customers as a commercial release.

Latency tests of ART.LAB system

This part of my other work is very relevant to my education at Hogeschool Zuyd. The ART.LAB system for MyLab 70 scanners has several different input channels that enter the system through different paths. Each of these signal paths has a different latency range. That means that data sets recorded through these systems are asynchronous to a certain degree.

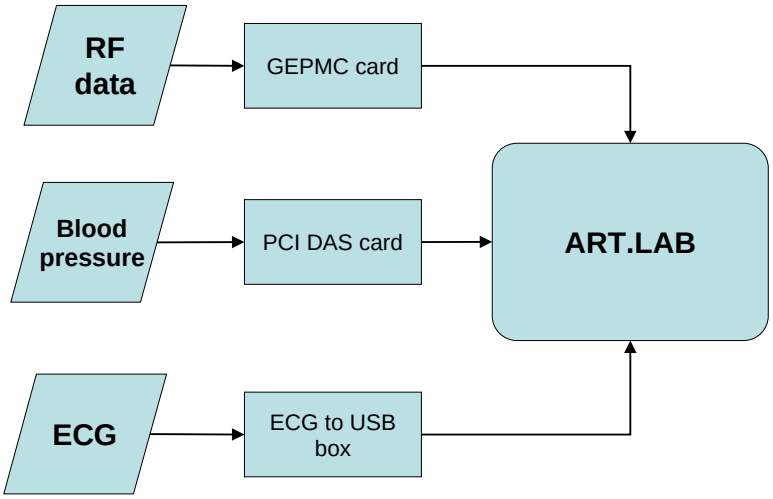


Figure 46: ART.LAB signal paths

To make the data reliable and usable, Esaote needs to know the differences in latency between these channels.

A test was in order, and we realised the following setup. We used a dual channel function generator to create a sharp transient on all three channels listed.

At the hardware department I was able to get hold of a coil that able to transmit 5 MHz radio signals, and a dual channel function generator. I then connected them in the following setup.

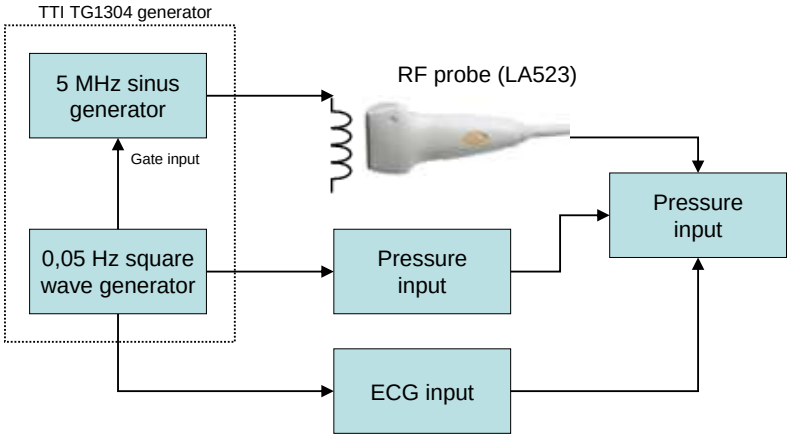


Figure 47: Test setup

This accomplishes the following:

When the square generator has a rising edge, the RF signal has a noticeable increase in amplitude, especially at lower depths. These events occur at exactly the same time. Using this, I could determine the delay between the RF and the reference channels, like shown in Figure 3.

For each scan mode we used I made 5 measurements, and read in the raw data using MATLAB. Then I manually determined the transient times of all signals and put them into a spreadsheet. This spreadsheet then calculated the delays and statistics on them.

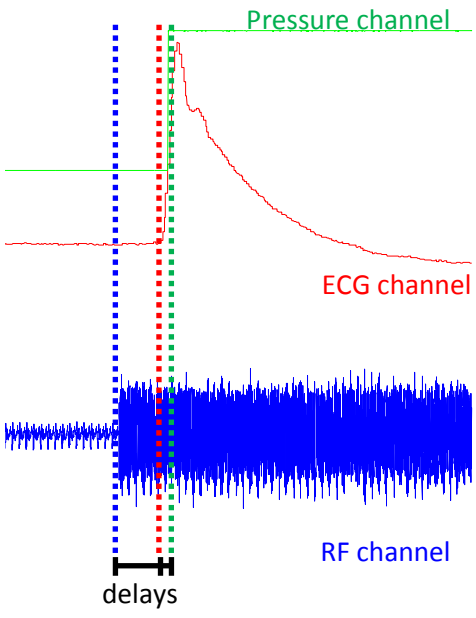


Figure 48: The three signals and their relative delays

Latency test result spreadsheet

B-mode 129 lines per frame

Line times of transient

File	RF	ECG	Pressure	RF to ECG	frames	RF to Pressure	frames
1	8417	9783	9723	1366	10.59	1306	10.12
2	9703	11331	11336	1628	12.62	1633	12.66
3	9809	11482	11477	1673	12.97	1668	12.93
4	10037	11611	11443	1574	12.20	1406	10.90
5	10593	12299	12105	1706	13.22	1512	11.72

Statistics	Avg lines	StDev lines	Avg frames	StDev frames	Relative
RF to ECG	1589.40	134.37	12.32	1.04	0.08
RF to Pressure	1505.00	151.99	11.67	1.18	0.10

Fast B-mode 152 lines per frame

Line times of transient

File	RF	ECG	Pressure	RF to ECG	frames	RF to Pressure	frames
1	5718	7867	8231	2149	14.14	2513	16.53
2	1552	3814	3740	2262	14.88	2188	14.39
3	6557	8779	8607	2222	14.62	2050	13.49
4	6189	8349	8559	2160	14.21	2370	15.59
5	5947	8146	8345	2199	14.47	2398	15.78

Statistics	Avg lines	StDev lines	Avg frames	StDev frames	Relative
RF to ECG	2198.40	46.14	14.46	0.30	0.02
RF to Pressure	2303.80	183.61	15.16	1.21	0.08

Figure 49: The latency test results

The above test results show the delay in transients, between RF and ECG on one side, and RF and Pressure on the other side. The delays are displayed in RF lines (the time it takes the scanner to scan one element on the probe) and frames (the time it takes to scan all elements on the probe). The most important fact is that the standard deviation of this delay is low (around 1 frame). If this deviation is low, the delay is quite stable and predictable. In this case, the stability of the delay is indeed satisfactory.

Visits to MUMC+

In the Maastricht University Medical Centre (MUMC+, a combination of the azM and Maastricht University) a total of three ART.LAB systems can be found. These systems are used in different branches of research, by different research institutes.

As soon as Peter Brands said a version of ART.LAB could be released to the researches at the MUMC+, I would contact their ultrasound researcher Jos op ‘t Roodt. I arranged to meet him at the hospital, in a room where he made sure all three systems would be.

Figure 50: MUMC+ researcher Jos op 't Roodt using an ART.LAB system to scan a volunteer

Depending on whatever Peter wanted to hand over to the researchers, I would perform one or more of these tasks on the systems:

Install ECG- or blood pressure inputs

All three systems were meant to eventually have ECG and blood pressure inputs, so these inputs could be used in measurement and research contexts. Not all systems were fitted with these inputs from the start. I was asked to install and test the missing interfaces.

Update the MyLab scanner software

Sometimes Esaote Italy would release new software for the MyLab 70XVG scanners present at the hospital. I would first install this on our own testing scanner. If I could not find any strange issues or unexpected changes, Peter Brands asked me to install this version at the scanners of MUMC+ too.

Update ART.LAB and relay feedback

Depending on my own testing results, Peter Brands would sometimes ask me to go and install new versions of ART.LAB on the ART.LAB PCs at MUMC+. Mostly these updates would accommodate the wishes of the researchers, or fix the problems of previous versions.
Regularly, the researchers had comments or questions on the new version I delivered. If this was the case, I wrote down the information and passed it on to Peter. He was then able to get back to the researchers on these topics, and perhaps order changes in the programme.

Visit to l'Hôpital Européen Georges-Pompidou, Paris

My presence was requested in this Parisian hospital at 23 April 2010. The purpose of this visit was to hand over the batch processing software I had created during my internship. It would have been possible to send them over the internet or the mail. Each programme even had its own user manual, but my presence there was useful. By being there, I could more easily demonstrate how to use the software and clear up any misunderstandings. Even if the manual was comprehensive and correct, misunderstandings could still rise. And of course, it is always pleasant to get to know people you work with face to face. Especially as there was a second visit planned.

I travelled there by high speed train, while Peter was already there. During this visit I received some feedback on my batch processing program. Later on I adjusted my programme in accordance with the requests and comments I received, and I sent the changed programme to them via email, including changed documents.

Figure 51: La Gare du Nord in Paris

Conclusion

All in all, I feel I have been left with quite some responsibility, even to the outside of the company. This seems like a good thing to me. The experience will definitely be useful in my career later on. Thanks to these activities, I'm familiar with far more than only MATLAB. I have increased my people skills, and have created a feeling for the way things are done within medium sized companies.

Chapter 8 – Conclusions

First of all, let's rephrase my main assignment: extend the existing MATLAB-based Time Point Extraction programme to use the ECG R-peak of every heart beat as a time base. To facilitate this objective I had to accomplish the following sub-objectives:

- Develop an automatic ECG detection and validation routine, to discard ART.LAB measurement files without valid ECG signal
- Make a programme that corrects the time base of the extracted time points, so that they use the ECG R-peak as a starting point
- Shape this into a programme that can automatically process an entire folder of ART.LAB measurement files

In all the above, I believe I have succeeded, by creating the set of three programmes: the ART.LAB Waveform Viewer, AbsoluteTPE batch processor and AbsoluteTPE viewer. With these three programmes, all ART.LAB arterial wall measurement files can be analysed and coupled together for research. This part of my work really made sense as a part of to my education, especially in digital signal processing and programming.

In my time at Esaote I was also involved in many other activities around ART.LAB software, hardware, documentation and support. These activities were quite a responsibility to me. I believe Peter Brands trusted me to do them well. I in turn gained self-confidence by doing them. The activities did take quite some time away from developing, so unfortunately I did not have time for any further assignments.

The next set of objectives will be passed on to my successor, whom I will acquaint with my work, so he or she can pick up right where I left off.

By performing the other activities I believe I have learned a great deal about all the different aspects to a technical platform, ranging from production and maintenance to servicing and people skills. I believe the successful development I did, coupled with the other activities, have greatly amounted to my personal development. All in all, I am quite happy after this internship.

Used literature

Printed literature:

Commissie van Opleidingen van de NVUBG (1991). Handboek Echografie. 1e dr. Leiderdorp: Leidse Onderwijs Instellingen (LOI).

Elling R., Andeweg B., de Jong J., Swankhuisen S (2004). Rapportagetechniek. 3e dr. Groningen/Houten: Wolters-Noordhoff

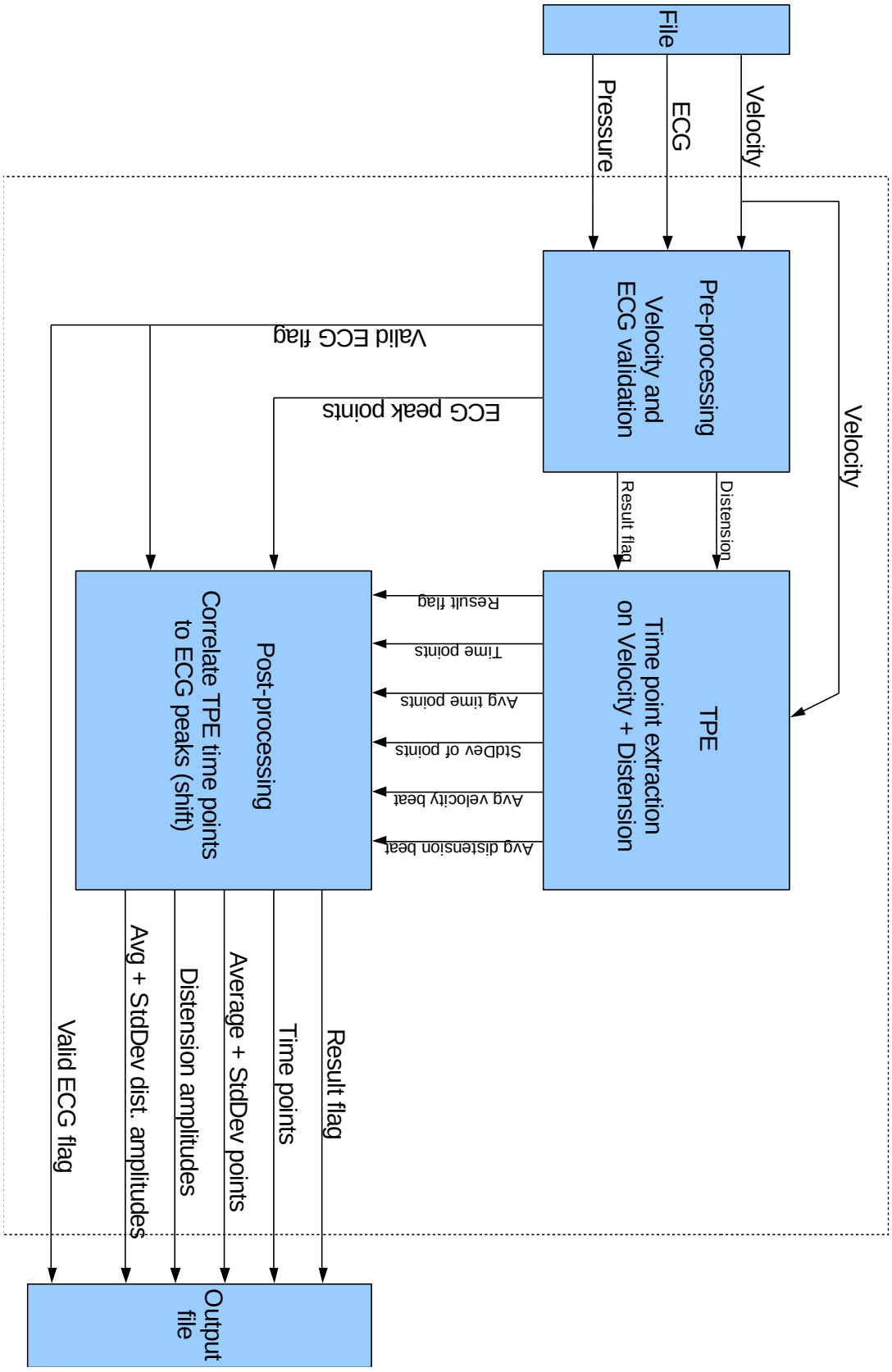
Electronic literature:

MathWorks MATLAB 2007b Online Help and online forums

PCI-DAS6023/PCI-DAS6025 Userguide
http://www.mccdaq.com/pdfs/manuals/pci-das6025-23.pdf

Various documents from internal ART.LAB design dossier

Appendix 1 – TPE process schematic



Appendix 2 - AbsoluteTPE batch processor - example of verbose output file

AbsoluteTPE-verbose.csv

BEGIN OF AbsoluteTPE VERBOSE OUTPUT
START OF HEADER

Folder	D:\Testbatch	
No of files	5	

END OF HEADER

START OF FILE

File	1	of	5
Filename	.\AS_fbmCCA2009_3_05___4.log		
File valid	YES		
ECG valid	YES	Time points counting from R-tops	
Framerate	688.07	fps	
TPE	SUCCEEDED		
Beats	3		

TIME POINTS

Beats	1	2	3
SIC	14	15	11
AVO max	40	43	35
AVO min	89	100	84
Max dist	100	111	98
IP	119	130	116
AVC min	196	219	199
AVC max	211	236	215
DN	226	252	232
PR	275	308	289
Start	571	1153	1766
End	589	589	589

AVERAGE TIME POINTS

	Avg sample	StdDev	Relative	Avg ms	StdDev	Relative
SIC	13	2.08	0.16	18.89	3.03	0.16
AVO max	39	4.04	0.1	56.68	5.87	0.1
AVO min	91	8.19	0.09	132.25	11.9	0.09
Max dist	103	7	0.07	149.69	10.17	0.07
IP	122	7.37	0.06	177.31	10.71	0.06
AVC min	205	12.5	0.06	297.93	18.17	0.06
AVC max	221	13.43	0.06	321.19	19.52	0.06
DN	237	13.61	0.06	344.44	19.79	0.06
PR	291	16.56	0.06	422.92	24.07	0.06

DISTENSION AMPLITUDES

Beats	1	2	3
SIC	-14.15	-13.82	-13.95
AVO max	-7.85	-8.52	-18.11
AVO min	390.39	398.72	371.45
Max dist	409.36	412.28	403.27
IP	382.07	390.5	379.37
AVC min	322.43	332.63	317.47
AVC max	256.55	260.83	254.04
DN	242.45	247.14	236
PR	281.81	279.65	269.93
Start	-0.83	-0.82	-1.29
End	-7.34	27.16	32.06

AVERAGE DISTENSION AMPLITUDES

	Avg value	StdDev	Relative
SIC	-13.97	0.16	-0.01
AVO max	-11.49	5.74	-0.5
AVO min	386.86	13.97	0.04
Max dist	408.3	4.59	0.01
IP	383.98	5.8	0.02
AVC min	324.17	7.73	0.02
AVC max	257.14	3.44	0.01
DN	241.86	5.59	0.02
PR	277.13	6.33	0.02

AVERAGE BEATS

Length	590					
Velocity		Read from left to right then downward				
-0.94397	-0.96974	-0.99751	-102.437	-104.451	-105.452	[CUT]
0.778209	0.811056	0.835777	0.849654	0.856501	0.856108	[CUT]
-0.75847	-0.74561	-0.73054	-0.71705	-0.70324	-0.69146	[CUT]
Distension		Read from left to right then downward				
-0.98105	-198.787	-302.245	-408.391	-51.655	-62.571	[CUT]
252.942	253.716	254.514	255.327	256.146	256.965	[CUT]
75.577	747.943	740.267	732.726	725.322	718.037	[CUT]

END OF FILE

[FILES 2 THROUGH 5 CUT]

BEGIN OF GLOBAL STATISTICS

GLOBAL FILE COUNTS

Files read successfully	5
Valid files	5

Files with valid ECG	5
Successful TPE files	5
Valid stat files	5

GLOBAL AVERAGE TIME POINTS

	Avg ms	StdDev	Relative
SIC	32.18	3.96	0.12
AVO max	79.55	8.76	0.11
AVO min	170.25	10.23	0.06
Max dist	216	25.49	0.12
IP	235.94	25.37	0.11
AVC min	393.86	25.23	0.06
AVC max	425.08	30.7	0.07
DN	467.5	33.44	0.07
PR	560.48	53.4	0.1

GLOBAL AVERAGE DISTENSION AMPLITUDES

	Avg value	StdDev	Relative
SIC	-16.96	1.76	-0.1
AVO max	-12.07	4.59	-0.38
AVO min	299.68	11.83	0.04
Max dist	329.22	9.31	0.03
IP	311.51	10.84	0.03
AVC min	256.73	17.46	0.07
AVC max	201.54	19.79	0.1
DN	183.34	19.07	0.1
PR	210.08	12.94	0.06

END OF GLOBAL STATISTICS

END OF AbsoluteTPE VERBOSE OUTPUT

Note: some output data was cut to save space, this is indicated in red.

Appendix 3 - AbsoluteTPE batch processor - example of physiological output file

AbsoluteTPE-physiological.csv – Time points in milliseconds only

BEGIN OF AbsoluteTPE PHYSIOLOGICAL OUTPUT
START OF HEADER

Folder	D:\Testbatch	
No of files	5	

END OF HEADER

START OF FILE

File	1	of	5
Filename	.\AS_fbmCCA2009_3_05___4.log		
File valid	YES		
ECG valid	YES	Time points counting from R-tops	
Framerate	688.07	fps	
TPE	SUCCEEDED		
Beats	3		

AVERAGE TIME POINTS

	Avg ms	StdDev	Relative
SIC	18.89	3.03	0.16
AVO max	56.68	5.87	0.1
AVO min	132.25	11.9	0.09
Max dist	149.69	10.17	0.07
IP	177.31	10.71	0.06
AVC min	297.93	18.17	0.06
AVC max	321.19	19.52	0.06
DN	344.44	19.79	0.06
PR	422.92	24.07	0.06

AVERAGE DISTENSION AMPLITUDES

	Avg value	StdDev	Relative
SIC	-13.97	0.16	-0.01
AVO max	-11.49	5.74	-0.5
AVO min	386.86	13.97	0.04
Max dist	408.3	4.59	0.01
IP	383.98	5.8	0.02
AVC min	324.17	7.73	0.02
AVC max	257.14	3.44	0.01
DN	241.86	5.59	0.02
PR	277.13	6.33	0.02

END OF FILE

[FILES 2 THROUGH 5 CUT]

BEGIN OF GLOBAL STATISTICS

GLOBAL FILE COUNTS

Files read successfully	5
Valid files	5
Files with valid ECG	5
Successful TPE files	5
Valid stat files	5

GLOBAL AVERAGE TIME POINTS

	Avg ms	StdDev	Relative
SIC	32.18	3.96	0.12
AVO max	79.55	8.76	0.11
AVO min	170.25	10.23	0.06
Max dist	216	25.49	0.12
IP	235.94	25.37	0.11
AVC min	393.86	25.23	0.06
AVC max	425.08	30.7	0.07
DN	467.5	33.44	0.07
PR	560.48	53.4	0.1

GLOBAL AVERAGE DISTENSION AMPLITUDES

	Avg value	StdDev	Relative
SIC	-16.96	1.76	-0.1
AVO max	-12.07	4.59	-0.38
AVO min	299.68	11.83	0.04
Max dist	329.22	9.31	0.03
IP	311.51	10.84	0.03
AVC min	256.73	17.46	0.07
AVC max	201.54	19.79	0.1
DN	183.34	19.07	0.1
PR	210.08	12.94	0.06

END OF GLOBAL STATISTICS

END OF AbsoluteTPE PHYSIOLOGICAL OUTPUT

Note: some output data was cut to save space, this is indicated in red.

Appendix 4 - AbsoluteTPE batch processor - example of rows output file

AbsoluteTPE-rows.csv – Time points in milliseconds only

Filename	fps	ECG valid	Avg pts	SIC	AV0 max	AV0 min	Max dist	IP	AVC min	AVC max	DN	PR	StdDev pt	SIC	AV0 max	.....
.\1019004	687.64	YES		18.91	49.44	91.62	216.68	158.51	301.03	319.93	0.00	0.00		1.45	3.22	.....
.\AS_fbm(	688.07	YES		18.89	56.68	132.25	149.69	177.31	297.93	321.19	344.44	422.92		3.03	5.87	.....
.\CL_fbm(	688.07	YES		0.00	72.67	132.25	236.89	177.31	319.73	342.99	376.41	424.37		0.00	5.09	.....
.\EB_fbm(	242.57	YES		61.84	160.78	362.78	428.74	515.31	787.40	853.36	956.42	1162.54		6.77	17.34	.....
.\JG_fbm(	687.64	YES		26.18	62.53	116.34	136.70	157.06	296.67	319.93	337.38	407.19		2.22	4.28	.....
.\SL_fbm(	687.64	YES		21.81	45.08	107.61	127.97	152.70	267.58	287.94	322.84	385.37		1.66	1.95	.....
.\1019000	687.64	YES		0.00	49.44	84.35	263.22	138.15	330.11	349.02	0.00	0.00		0.00	2.52	.....

Note:

This file format should normally be viewed horizontally, as in the above picture, but this was impossible to print properly . The different sections of the example below are normally joined at … .

Filename	fps	ECG valid	Avg pts	SIC	AV0 max	...
.\AS_fbmCCA2009_3_05___4.log	688.07	YES		18.89	56.68	...
.\CL_fbmCCA2009_3_04___2.log	688.07	YES		0	72.67	...
.\EB_fbmCCA2009_3_04___3.log	242.57	YES		61.84	160.78	...
.\JG_fbmCCA2009_3_06___1.log	687.64	YES		26.18	62.53	...
.\SL_fbmCCA2009_3_05___2.log	687.64	YES		21.81	45.08	...

...	AV0 min	Max dist	IP	AVC min	AVC max	...
...	132.25	149.69	177.31	297.93	321.19	...
...	132.25	236.89	177.31	319.73	342.99	...
...	362.78	428.74	515.31	787.4	853.36	...
...	116.34	136.7	157.06	296.67	319.93	...
...	107.61	127.97	152.7	267.58	287.94	...

...	DN	PR	StdDev pts	SIC	AV0 max	...
...	344.44	422.92		3.03	5.87	...
...	376.41	424.37		0	5.09	...
...	956.42	1162.54		6.77	17.34	...
...	337.38	407.19		2.22	4.28	...
...	322.84	385.37		1.66	1.95	...

...	AV0 min	Max dist	IP	AVC min	AVC max	...
...	11.9	10.17	10.71	18.17	19.52	...
...	8.08	47.96	12.94	20.99	21.85	...
...	15.26	27.9	53.32	45.2	59.13	...
...	6.96	6.65	7.06	16.47	16.56	...
...	5.94	4.76	6.45	9.88	9.14	...

...	DN	PR	Avg Dist ampl	SIC	AV0 max	...
...	19.79	24.07		-13.97	-11.49	...
...	24.86	25.37		-21.54	-24.88	...
...	64.17	112.03		-7.07	-5.48	...
...	17.08	19.65		-24.28	-4.47	...
...	13.15	9.99		-22.52	-14.04	...

...	AV0 min	Max dist	IP	AVC min	AVC max	...
...	386.86	408.3	383.98	324.17	257.14	...
...	254.49	303.9	285.06	266.69	216.3	...
...	142.5	156.6	143.93	100.68	73.63	...
...	425.15	465.57	445.91	378.18	294.36	...
...	289.42	311.74	298.65	213.95	166.26	...

...	DN	PR	StdDev DistAmp	SIC	AV0 max	...
...	241.86	277.13		0.16	5.74	...
...	195.2	202.78		3.59	1.56	...
...	65.1	83.63		0.98	2.8	...
...	273.86	324.34		2.89	6.34	...
...	140.68	162.54		1.77	4.69	...

...	AV0 min	Max dist	IP	AVC min	AVC max	...
...	13.97	4.59	5.8	7.73	3.44	...
...	7.95	9.15	2.04	8.68	9.96	...
...	8.86	8.67	13.61	12.77	9.36	...
...	16.85	12.58	15.34	28.69	35.98	...
...	8.91	9.74	11.34	20.08	21.56	...

...	DN	PR
...	5.59	6.33
...	9.93	8.77
...	8	7.48
...	34.97	19.85
...	20.05	16.45

Appendix 5: Internship assignment plan (Dutch)

Plan van aanpak Afstudeerstage - Hubert Bessems, 0620343

Inhoudsopgave

1. Aanleiding

Al gedurende enige jaren werkt Esaote Europe B.V. (vóór 1998 bekend als Pie Medical Group) samen met de Universiteit Maastricht en het academisch ziekenhuis Maastricht in onderzoek naar echografie door middel van ultrasound en toepassingen daarvan. Onderdeel van deze samenwerking is de opstelling van zogenaamde research-scanners op de universiteit. Deze ultrasound-scanners zijn gelijk aan de systemen die in de zorg gebruikt worden, maar hebben één belangrijke toevoeging. Deze systemen hebben een digitale data-uitgang waarmee de ruwe scandata van bloedvaten realtime naar een research-PC worden overgezet. Op deze research-PC draait ART.LAB, een programma dat deze data weergeeft, analyseert en archiveert. Aan dit programma worden telkens verbeteringen en nieuwe functies toegevoegd.

In de gearchiveerde databestanden zijn naast ultrasound-gegevens en daaruit afgeleide signalen (bijvoorbeeld uitzetting en wandsnelheid van het bloedvat), ook vaker bloeddruk- en ECG-signalen aanwezig. Esaote zoekt een manier om automatisch te bepalen of de aanwezige bloeddruk- en ECG-signalen geldig zijn, om vervolgens deze signalen te correleren aan de uitzetting en wandsnelheid van het bloedvat. De hieruit verworven data kunnen vervolgens voor onderzoek gebruikt worden, en later wellicht opgenomen worden in ART.LAB.

2. De afstudeeropdracht

1. **Projectomgeving**
   De opdracht heeft plaats op de afdeling Advanced Projects van Esaote Europe B.V., in het gezelschap van twee afstudeerders van de faculteit Technische Informatica, daarnaast de ontwikkelaar van ART.LAB (David Sontrop), onder leiding van het hoofd Advanced Projects, Peter Brands. Mijn werkkamer ligt centraal in het bedrijfspand, en mensen lopen er vaker en uit. Ook komt ik in aanraking met verschillende bedrijfsprocessen (logistiek en dergelijke), voornamelijk vanwege de nevenactiviteiten naast mijn hoofdopdracht.

2. **Opdrachtformulering**
   Mijn opdracht is uiteen te zetten in vier fasen:
   1. Maak een programma dat automatisch bepaalt of er een geldige ECG-signaal aanwezig is. Controleer dit op basis van de relatie met het wandsnelheid-signaal. Als het ECG-signaal geldig is, correleer dan de TPE[8]-punten van het distensiesignaal[9] aan de tijdbasis van het ECG-signaal. Geef de TPE-punten van het distensiesignaal dus een absolute verschuiving t.o.v. de R-top van het ECG-signaal.

[8] TPE: Time Point Extraction. Voorganger Marc van Dijk heeft tijdens zijn afstudeerstage een programma gemaakt dat de buigpunten uit het uitzettingssignaal (Distension) haalt.
[9] Distensiesignaal: signaal dat de verandering in de diameter van een bloedvat doorgeeft

2. Maak een computerprogramma dat de verwerking van punt 1 automatisch uitvoert op mappen met logbestanden.
3. Maak een aanvulling op bovenstaand programma dat controleert of er een geldig bloeddruksignaal aanwezig is. Dit kan weer gecontroleerd worden op basis van het ECG-signaal of het Velocity-signaal.
4. Zoek de buigpunten in het Velocity-signaal op een vergelijkbare manier met de bestaande TPE. Als er daarnaast een geldig bloeddruksignaal is, zoek hierin dan ook de buigpunten. Relateer beide sets punten aan de R-top van het ECG-signaal, mits aanwezig.

Naast deze concrete opdracht zijn er ook enkele nevenactiviteiten, waaronder het testen van nieuwe versies van ART.LAB-software op MyLab 70 ultrasound-scanners, het updaten van scanners, het uitvoeren van installatiewerkzaamheden omtrent scanners en accessoires, het documenteren van ontwikkelde producten en ondersteuning bieden aan de gebruikers van de research-scanners in het Maastricht Universitair Medisch Centrum (MUMC+).

3. **Op te leveren product**
   Een computerprogramma voor het automatisch valideren van bloeddruk- en ECG-signalen in opgeslagen metingsbestanden, dat daarna deze metingen correleert aan metingen van de distensie en wandsnelheid van het gescande bloedvat.

4. **Eisen en beperkingen**
   1. De verwerking van het uiteindelijke programma moet geheel automatisch verlopen
   2. Onbruikbare signalen moeten automatisch worden genegeerd
   3. Op termijn wordt er een vast uitgangsformaat gespecificeerd, waaraan het ontwikkelde programma zich dient te houden

3. Aanpak

In eerste instantie maak ik kennis met het bedrijf en de mensen waarmee ik samenwerk. Verder lees ik mijzelf in op de literatuur die de afstudeerders tot hun beschikking hebben over ultrasound.

Daarna begin ik aan het programma in Matlab, een makkelijke omgeving om de gespecificeerde gewenste functionaliteit in te ontwikkelen. Ik neem mijn volgorde over van de volgorde die staat genoemd onder *Eisen en beperkingen*. Hetgeen daar als eerste vernoemd staat ontwikkel ik ook als eerste. Bij elke deelopdracht zal ik ook een tijd moeten testen, om te verifiëren of alles echt zo werkt als gewenst. Dit testen zal ik doen met voorbeelden van reële patiëntgegevens.
Wanneer deze functionaliteit naar tevredenheid van Peter Brands werkt, kan de Matlab-code eventueel vertaald worden naar C++. Ook tijdens en na afloop van de vertaalslag zal ik blijven testen.

Tijdens de ontwikkelfases en testfases zal ik naar alle waarschijnlijkheid ook vaker dingen moeten heroverwegen, en aan de hand daarvan het ontwerp bijstellen. Verder zal ik gedurende de ontwikkeling ook duidelijke documentatie aanleggen over de werking van het programma, zowel voor mijn eindverslag als om te zorgen dat het later gemakkelijk overgedragen kan worden.

Aanvullende plannen

- **Activiteitenplan**

Activiteit	wk	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
Kennismaken met bedrijf																							
Inwerken																							
Bekend raken met Matlab																							
Ontwikkelen validatie ECG																							
Ontwikkelen correlatie ECG en TPE																							
TPE toepassen op Velocity-signaal																							
Batchverwerking																							
Ontwikkelen validatie bloeddruksignaal																							
Ontwikkelen correlatie ECG en bloeddruksignaal																							
TPE toepassen op bloeddruksignaal																							
Testen Matlab-programma																							
Omzetten functionaliteit naar C++ (eventueel)																							
Testen C++ (eventueel)																							
Documenteren																							
Maken eindverslag																							
Voorbereiden presentatie																							

Wat betreft overige plannen:

Een communicatieplan lijkt mij niet nodig. Communicatie geschiedt vrij direct, vaak mondeling met Peter Brands en David Sontrop, en belangrijke zaken worden via mail uitgewisseld. Ik zal eens per enkele dagen verslag uitbrengen aan Peter, en bij vragen liefst gelijk contact opneem met iemand van wie ik denk dat diegene mij kan helpen. Dit valt lastig te plannen. Af en toe zal ik ook het initiatief nemen om een gesprek te onderhouden met Peter over waar ik precies sta, en waar eventuele knelpunten zitten.

Een documentatieplan lijkt me ook niet nodig. Ik documenteer de zaken die ik ontwikkel, ten eerste door commentaar in mijn code, ten tweede door flow-diagrammen en andere aantekeningen, en ten slotte nog door uitgewerkte documenten, waarin ik de eerste twee items opneem om er één geheel van te maken. Dit gebeurt tijdens het ontwikkelen en die inhoud kan ik later dan weer opnemen in mijn eindverslag. Ondertussen houd ik dagelijks nog een logboek bij.