Adaptive Parameter Determination for Record Processing

Bachelor Thesis

TIC

Author :

Michael Heinzer
heinzerm.ch@gmail.com

Expert :

Noé Lutz
noe.lutz@google.com

Professors :

Ottar Johnsen
ottar.johnsen@hefr.ch
Frédéric Bapst
frederic.bapst@hefr.ch

Supervisors :

Carl Haber
chaber@lbl.gov
Earl Cornell
ewcornell@lbl.gov

14.08.2012

Class I-3

Abstract

At the Lawrence Berkeley National Laboratory old records are reconstructed using state of the art imaging technology. In an effort to improve sound quality, a study which includes multidimensional minimization on the edge detection parameters was carried out. Despite the use of different minimization techniques as well as various approaches on edge detection this method fails to deliver significantly better results.

Le Lawrence Berkeley National Laboratory dispose d'une technologie de pointe basée sur le traitement d'image pour la reconstruction de vieux enregistrements audio. Pour améliorer la qualité du son généré, un projet utilisant des algorithmes de minimisation pour la détection des contours a été réalisé. Malgré l'application de plusieurs approches pour les deux sujets, la méthode générale ne parvient pas à apporter de grandes améliorations de la qualité du son.

Am Lawrence Berkeley National Laboratory werden alte Schallplatten mit neuster Technologie wiederhergestellt. Um die Qualität zu verbessern wurde ein Projekt welches die multidimensionale Minimisierung der Kantendetektion beinhaltet durchgeführt. Trotz der Nutzung von verschiedenen Techniken für beide Teile ist die Methode nicht geeignet die Qualität signifikant zu verbessern.

Contents

List of Figures

1 Introduction

At the Lawrence Berkeley National Laboratory, the reconstruction of old records is undertaken with great effort. Carl Haber and his team extract sound with optical methods from old records. Some of them could not be read otherwise without destroying the disk in the process. Students from the College of Engineering and Architecture, Part of the University of Applied Sciences Western Switzerland have been supporting Mr. Haber in this undertaking for the last six years.

The acquisition is done with two different systems called IRENE and 3D probe. Both of them retrieve the information with optical methods. IRENE takes pictures of the surface, which results in 2D data with different intensities. The 3D probe uses a laser to generate a 3D view of disks. The recorded information is then processed and analyzed by two programs, PRISM for 3D and RENE for 2D information. The structure of the entire sound extraction project can be summarized as in the following image [Fig: 1] .

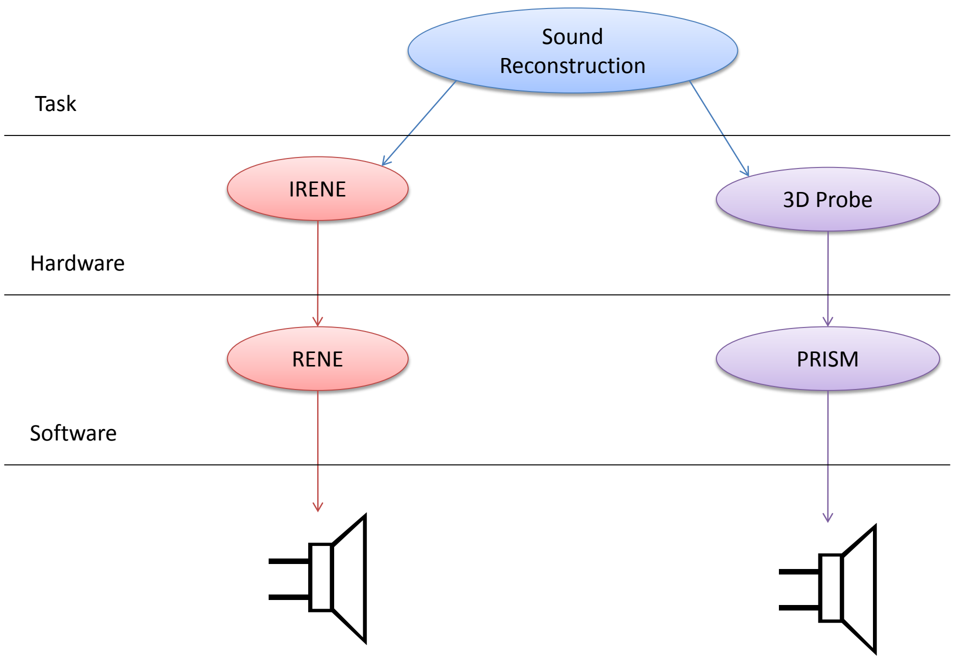


Figure 1: Overview of the sound extraction project

RENE and PRISM are capable of recalculating the sound from the acquired data. This bachelor thesis will focus primarily on the left part in the image [Fig: 1] , the processing of 2D data.

1.1 Context

The sound extraction is done in various steps. The first of them is an edge detection algorithm which will recognize horizontal changes in intensity. This helps identify the grooves in the image. Once the grooves are known to the program, it is able to translate the movement of the grooves into sound. The sound will then be written to the disk in the form of a wave file. The whole process can be illustrated as in the following image [Fig: 2] .

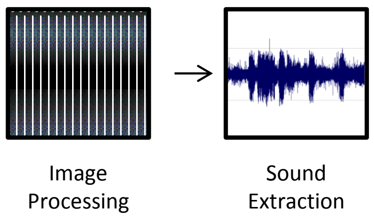


Figure 2: Basic functions of RENE

The image processing part contains the edge detection and smoothing algorithms. The sound extraction part portrays the conversion of optical to acoustical data. This view of the process is important because it displays the state of the project before this thesis. In order to perform an automated parameter adaption, several steps have to be added.

One might ask why the adaption of parameters is necessary more than once. One reason is that disks made of different material exhibit distinct groove movement for similar sound. This leads to noise in the final result. The case is best illustrated by the image [Fig: 3] of two grooves.

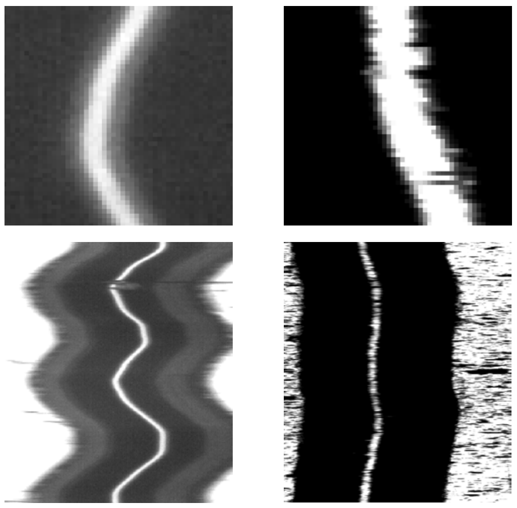


Figure 3: Left: lacquer. Right: shellac

The image [Fig: 3] above shows how the material influences the border of the grooves. The upper images are a close-up view of the bottom of the groove; the lower images display how the entire groove looks. In the case of the shellac, the borders are frayed. This will lead to noise component added at the 1 kHz frequency.

Another important fact is that all of the disks which were encountered during this thesis start with a short period of silence. It usually lasts between one and three seconds. If one listens to the audio file, it is not silent at the beginning but contains noise. This knowledge will be used to determine how the parameter adaption performs. Put simply, the goal is to change the parameters to reduce the noise in this first part without impairing the sound quality.

This leads to the insertion of two additional steps into the sound extraction process. Firstly the detection of the beginning of the sound, herein called the silence sound transition and secondly the parameter adaption process. Once they have been optimized, the whole extraction process starts over with the new settings. The improved software therefore has six steps as depicted in the next image [Fig: 4] .

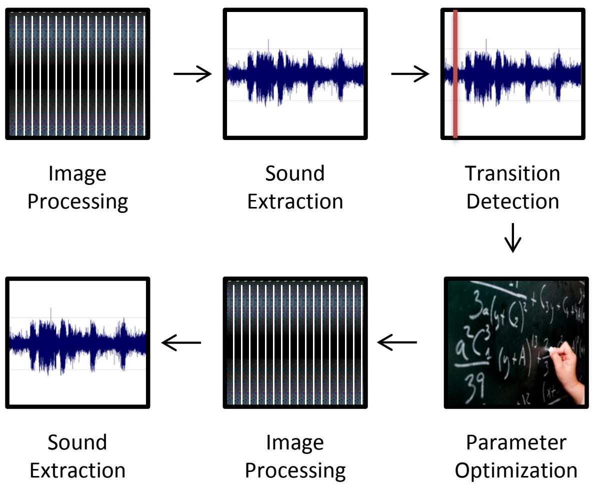


Figure 4: Basic functions of RENE afterwards

These two steps contain numerous subtasks and the activities to complete them will be outlined in the next two sub chapters.

1.2 Targets

Early in the project, the student and the supervisors have to agree on a functional specification which contains goals for the project upon which the result will be measured. The following 13 targets contain the general workflow. These goals apply for 2D data, if time allows the study will be expanded to 3D data as well.

1. Survey and study of prior work
2. Understand the IRENE code package used for 2D analysis
3. Selection of test disc sample, around 5-6 discs will be chosen as the initial test group for this study
4. Development of code to find and isolate quiet portions of the audio
5. Development of code to measure noise and spectral properties on these portions
6. Plots and summaries to monitor the above mentioned process
7. Study of edge detection algorithms
8. Selection of analysis parameters to vary in the optimization
9. Study of optimization processes and multi-dimensional minimization search algorithms
10. Development of code to perform and display the results of the optimization process
11. Study and comparison on the disc test sample
12. Final data analysis and conclusions
13. Documentation and Finalization

1.3 Activities

This section contains details of the targets which were specified in the previous subsection. A few short sentences will describe how each goal will be attained and if possible which technique will be used.

Survey and study of prior work
Read any previous Bachelor thesis which is related with the current project. Understand the concept of the image acquisition and the different characteristics of the image processing.

Understand the IRENE code package used for 2D analysis
The program which analyses the data retrieved by IRENE is called RENE and contains thousands of lines of code. The principal functions and the parts which are crucial for the modification have to be read and understood.

Selection of test disc sample, around 5-6 discs will be chosen as the initial test group for this study
Select records which contain different types of records but the same noise and sound structure. This means the records must start with a silent part and continue with sound. A lead in groove must occur in the image. Most records have this structure but not all the recorded files start from the beginning.

Development of code to find and isolate quiet portions of the audio
Introduce a code fragment in RENE which is capable of deciding when the quiet part ends and the music or voice starts. It is important that this task is achieved with a certain precision to do the minimization on the correct part of the sound file.

Development of code to measure noise and spectral properties on these portions
This task is included in the previous task but the methods which were used should be accessible by any other part of the program. The standard procedure to measure spectral properties is a Fourier Transformation which allows measuring the intensity for a certain interval of frequencies. Moreover the noise reduction which is supposed to be achieved by the parameter adaption has to be quantified; this requires grading the audio output.

Plots and summaries to monitor the above mentioned process
To present the results acquired in the noise detection phase they have to be properly visualized. A proper way to visualize the results is graphs.

Study of edge detection algorithms
The analysis of images requires a deeper understanding of the different edge detection algorithms and their parameters. This includes the currently implemented algorithms.

Selection of analysis parameters to vary in the optimization
In order to avoid wasting calculation time, the range of the parameters has to be shrunk to a handful of values; these will be the boundaries of the minimization process.

Study of optimization processes and multi-dimensional minimization search algorithms
Finding minima in n-dimensional space requires multi-dimensional minimization algorithms. Fortunately they have been developed before and efficient implementations exist. These will be reviewed in this section.

Development of code to perform and display the results of the optimization process
The extensive analysis has then to be integrated in RENE; the results will be visualized to inform the user of the progress and provide a feedback of the improvements.

Study and comparison on the disc test sample
The previously selected samples will be tested and the results have to be interpreted. If necessary the code has to be adapted in the next step to ensure proper results under various circumstances

Final data analysis and conclusions
A final conclusion will be drawn and the results will be integrated into RENE.

Documentation and Finalization
The whole report has to be written, adapted to the most recent events and insights. Moreover it has to be read by an external person and adapted to the remarks. The source code has to be documented and structured properly to be readable for the next developer.

1.4 Definitions

In order to understand all the activities described in this report, a certain vocabulary and knowledge about the acquisition process has to be known by the reader. This will be established in this section. The following explanations are of course a simplified version of reality and do not take into account the difficulties which might arise during these processes, such as damaged, deformed or unclean disks. If we take a record and make cut a tiny piece out of it, from the center to the border. It would have the following shape [Fig: 5] :

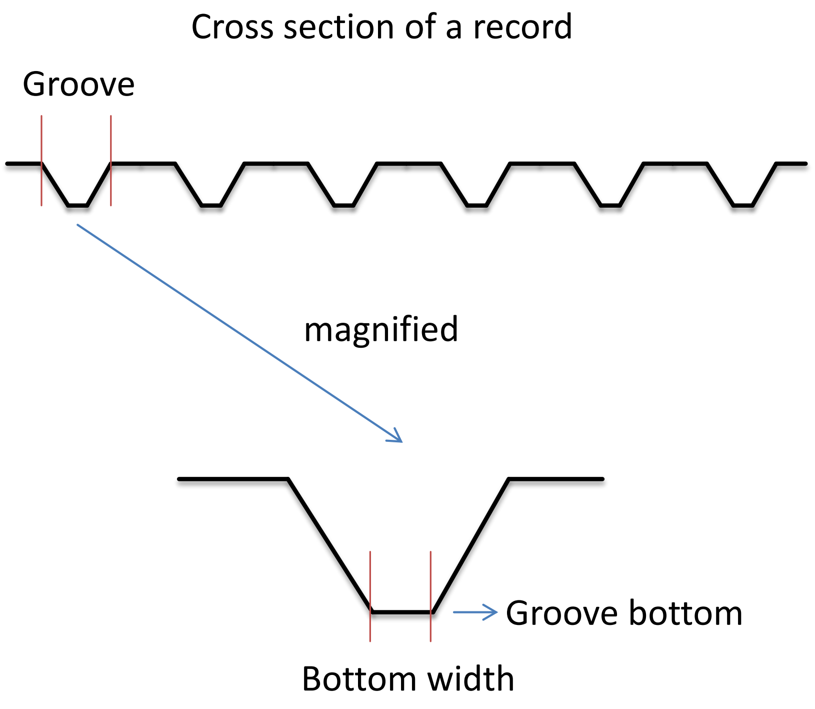


Figure 5: Cross section of a record

The immersions will be called groove [Fig: 5] . One record has only one groove which was continuously embossed. If we would watch this groove from above it would have the shape of a spiral. In the digitalization process, light is sent uniformly distributed to the disk from above. But the reflection differs inside a groove, while the bottom is as bright as the untouched parts; the descents appear darker due to the skewed reflection. The process is visualized in the next image [Fig: 6] :

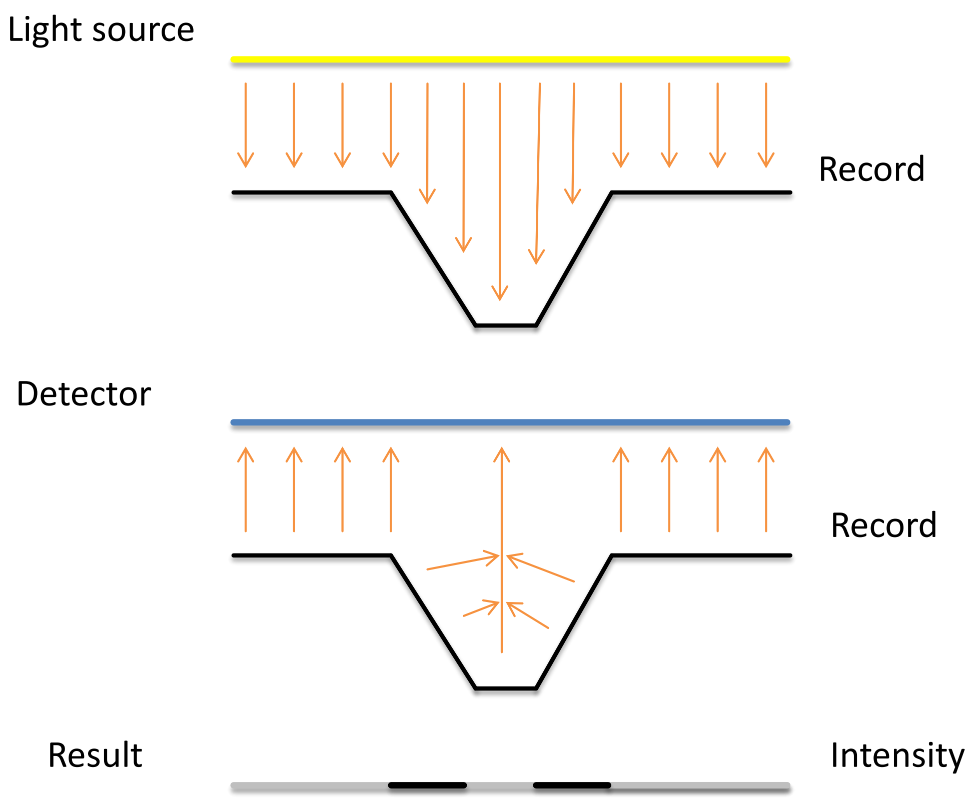


Figure 6: Scanning process

The result is a line which contains two small black or darker parts which represent the grooves. This result can be represented as a function of width and intensity [Fig: 7] . The digitalization consists of attributing values to the intensity and stores them as grayscale pixels.

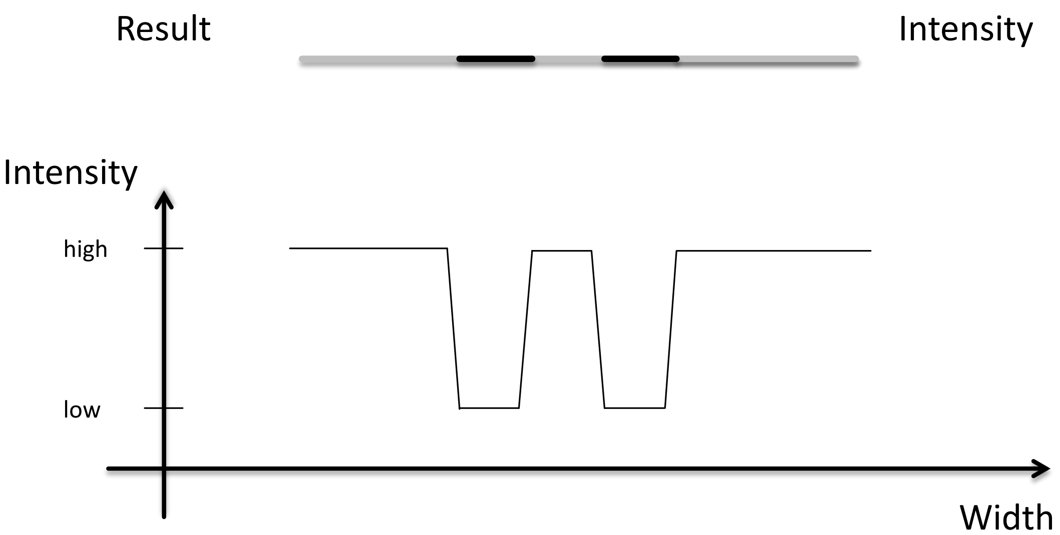


Figure 7: The result as a function

This process is repeated over and over in small intervals until a picture of the whole record exists. In reality the difference between the bottom and the slope has often less contrast than the image above and the transition happens gradually. A real word example of one groove is given below [Fig: 8] :

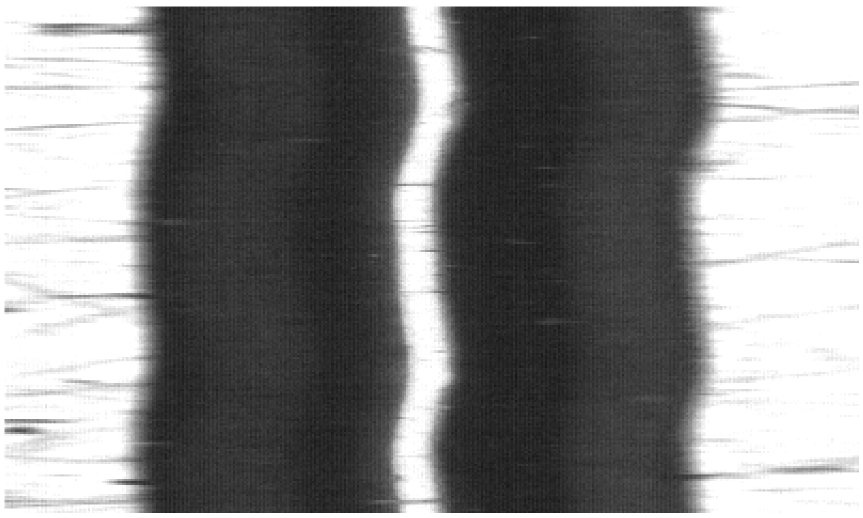


Figure 8: Extract of DCS record at the original size of the image

The extract [Fig: 8] is from a real scan where the record is in good shape. The contrast is high and the groove bottom is visible. Although it is not clear now how the groove bottom width will be measured, different threshold values will change the result considerably. The parameters which lead to the detection of these properties are described in the analysis section of this report.

1.5 Physical Characteristics of Records

When analyzing these records, one has to be aware of the physical properties of them. If we were to visualize only the groove, it would have the shape of a spiral. This effect has to be calculated and subtracted from the data. This is also true for the characteristics of the white noise. In the data it is uniformly distributed in the spectrum. But because for the wav file the derivative has to be taken, the noise will rise constantly from the left to the right in the spectrogram, from the lower to the higher frequencies with a gain of 6 dB per Octave.

1.6 Software

This chapter contains a brief overview over the software which was used during the course of this project. This also includes programs which were used for testing purposes but not in the final workflow.

1.6.1 Microsoft Visual Studio 2008

RENE is written in C# and a Visual Studio project. Visual Studio is an integrated development environment, similar to eclipse. It provides numerous tools which facilitate the development of .NET projects. One of them is an integrated debugger.

1.6.2 NI Vision Assistant 2010

The National Instruments Vision Assistant provides numerous tools for machine vision, one of them being an edge detector with various parameters. This was most useful for testing the response of edge detection algorithms to different parameter adaptations.

1.6.3 Sony Sound Forge Pro 10.0

This tool was used to analyze wave files. It provides among others a spectral analyzer which was used to compare the properties of numerous sound files. Moreover it contains multiple filters which are capable of reducing clicks and crackles.

1.6.4 LabView 2010

LabView is a program which allows writing software simply by drag and drop. No code has to be written; all elements are inserted and connected with the mouse. In the course of this project, it was used to test a number of techniques to visualize properties sound. The functionalities which were used were part of the Signal Processing toolkit which is an extension to the main program and has to be bought additionally.

1.6.5 R

R is a program for the statistical analysis of data. It has been used to draw histograms of different kinds of data and estimating the similarity between the normal distribution and the data.

1.7 Structure of the report

The first chapter will introduce the surroundings of the project; it includes also the definitions of certain terms which will be frequently used in the thesis. Moreover the goals of the project are being specified. The second chapter contains the analysis which was carried out to cover the areas of interest which were outlined in the chapter before. The theoretical foundations will be laid out for the implementation. The ideas which were evoked in the first two chapters will be put to practice in the third. It contains explanations how the concepts have been applied and obstacles overcome. If any of these pitfalls will be to any interest of for future projects they will be explained in detail in the fourth chapter which is dedicated entirely to this task. The verifications which have been made to ensure the proper functionality of the newly implemented functions are being described in the fifth chapter called tests. In the subsequent chapter a conclusion is drawn and recommendations are given concerning the future of the project. At last the appendix contains a glossary and user manuals for the particular curious reader.

This chapter contains mainly general information about the subject. An introduction as well as the definition of the goals for the projects was made.

2 Analysis

This chapter will describe the current status of the RENE program as well as the specifics which are important to this thesis. Specifically the current tracking program, including the edge detection algorithm will be examined in detail.

Theoretical concepts of minimization, spectrum analysis and edge detection will be reviewed in this part to give the reader an introduction to the subject and provide an insight into the work carried out during the analysis phase.

2.1 Overview of RENE

RENE is the software used to analyze data acquired by IRENE, the 2D probe. It provides a GUI with numerous options and is capable of extracting sound of images with various qualities. The principle is simple, an image is given as input and the program produces a sound file. Different aspects of this process will be highlighted in this sub chapter.

2.1.1 Input

The input is usually stored in multiple bitmap files. The name of the file contains useful information such as the name of the record, image acquisition settings, the width of the recording and the position. It has the following format: "name of the record_settings_tour number and letter indicating the part.bmp". Where tour number is the number indicates the number of rotations for this scan. An example of a filename would be "DCS_E100-I126-R118_0a.bmp" [Fig: 9] .In the cases which have been encountered in this project, there were eight parts for one tour.

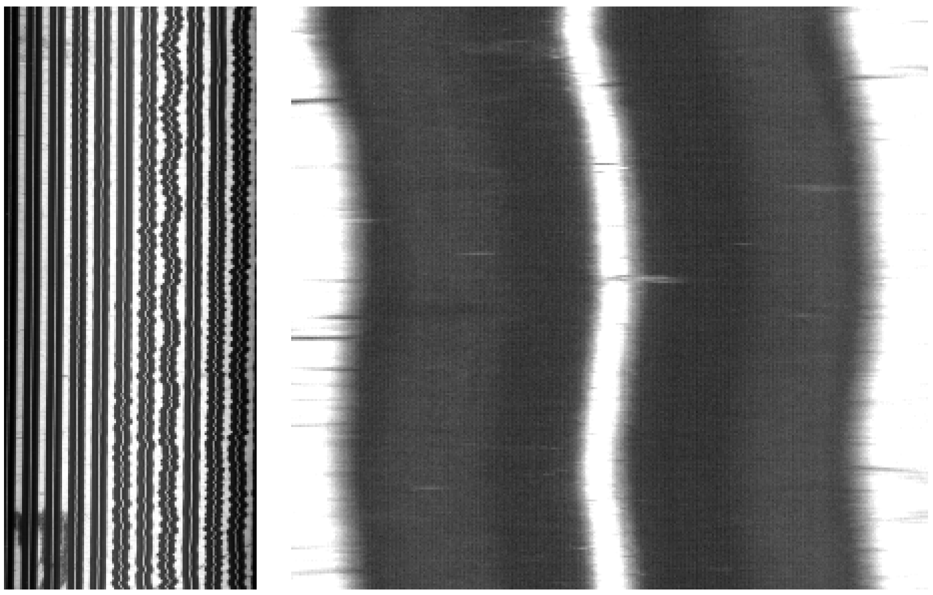
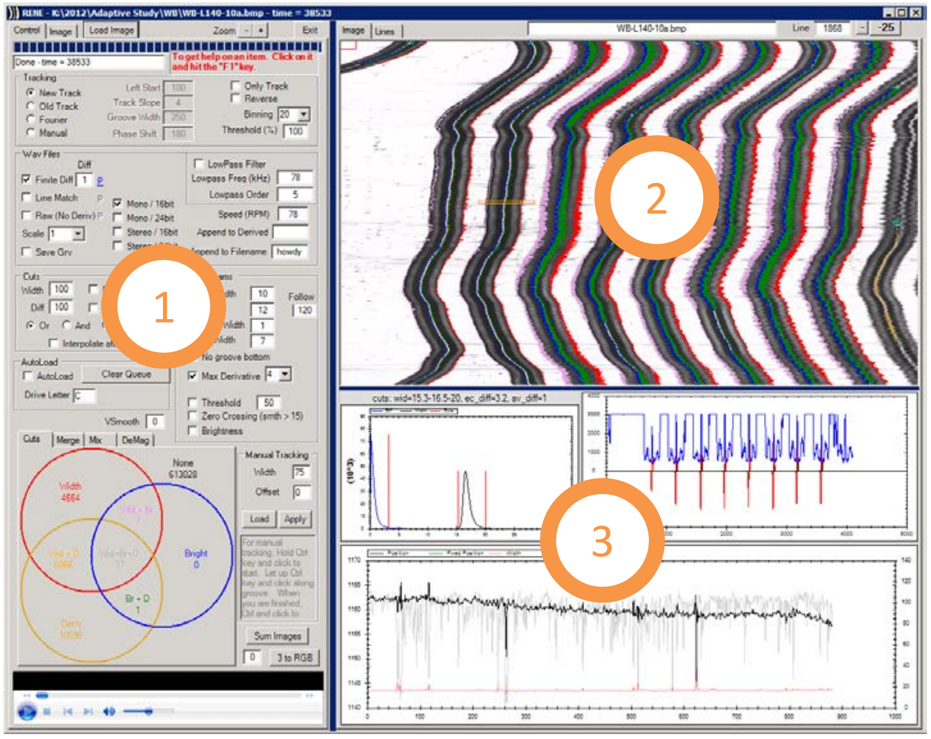


Figure 9: Left: 3% Zoom, Right: 100% Zoom

Eight of these images [Fig: 9] are then being put together by RENE. At the same time a reduced image is created upon which basic tasks will be executed. This includes a rough version of the edge detection. The software discovers where the grooves are located.

2.1.2 Interface

The interface is straightforward and contains all options on one single tab. The interface [Fig: 10] can be divided in three basic parts:


RENE - K:\2012\Adaptive Study\WB\WB-L140-10a.bmp - time = 38533

Done - time = 38533

To get help on an item. Click on it and hit the "F1" key.

Tracking: [✓ New Track] Old Track Fourier Manual; Left Start 100; Track Slope 4; Groove Width 250; Phase Shift 180; Only Track ☐; Reverse ☐; Binning 20; Threshold (%) 100

Wav Files: Finite Diff ✓ 1; Line Match ☐; Raw (No Deriv) ☐; Scale 1; Save Grv ☐; Mono / 16bit ✓; Mono / 24bit ☐; Stereo / 16bit ☐; LowPass Filter ☐; Lowpass Freq (kHz) 78; Lowpass Order 5; Speed (RPM) 78; Append to Filename howdy

Cuts: Width 100; Diff 100; [✓ Or] And; Follow 120; Max Derivative ✓ 4; Threshold ☐ 50; Zero Crossing (smth > 15) ☐; Brightness ☐

AutoLoad ☐; Drive Letter C; VSmooth 0

Manual Tracking: Width 75; Offset 0

None 613028; Width 4664

WB-L140-10a.bmp; Line 1868; -25

cuts: wid=15.3-16.5-20, ec_diff=3.2, av_diff=1

Figure 10: The three basic parts of RENE

Each of these parts has an essential function:

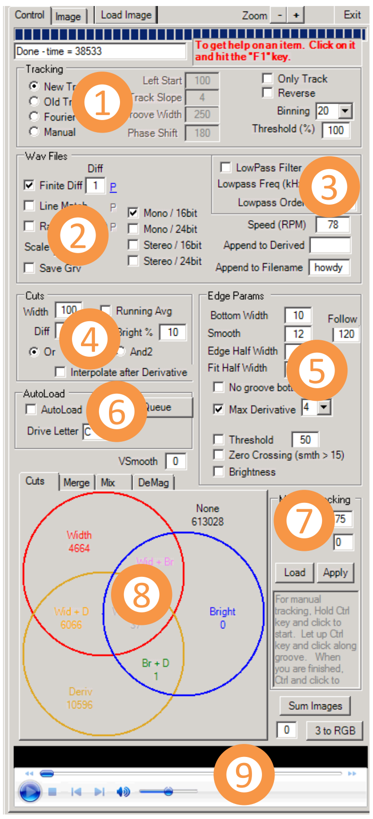
1. The control panel. Parameters for the extraction process.

2. Currently loaded image. Shows the progress.

3. Statistics about the selected section.

The extraction process is immediately started once an image has been chosen with the ”Load image” button which is placed at the top left part of the control section. Once the program is running, the options should not be changed anymore. Although some of them are being loaded only when the program starts the subroutine, this might lead to unexpected behavior. There is no button to stop the execution once it has been started.

The most important part of the program is the control panel, it has nine principal subsections [Fig: 11] .



Done - time = 38533

To get help on an item. Click on it and hit the "F1" key.

For manual tracking, Hold Ctrl key and click to start. Let up Ctrl key and click along groove. When you are finished, Ctrl and click to

Left Start 100, Track Slope 4, Groove Width 250, Phase Shift 180, Binning 20, Threshold (%) 100, Diff 1, Speed (RPM) 78, Append to Filename howdy, Width 100, Bright % 10, Bottom Width 10, Smooth 12, Follow 120, Max Derivative 4, Threshold 50, VSmooth 0

Figure 11: The nine sections of the control panel

Explaining every single subsection and the parameters in detail would be too time-consuming, for this reason only a brief is given in the following list:

1. The four available tracking algorithms, for this thesis only the ”new track” option will be used. The others are either dated, or only useful for disks recorded under special circumstances such as side lightening.

2. Decides which algorithm will be used to transform the coordinates into sound. The output sound quality as well as the channels can be chosen too. For this thesis only the 16 Bit Mono output will be used to create comparable results.

3. A low pass filter can be applied to the result. Not selected by default.

4. Determines when an edge will be cut and where. There are three possibilities to cut sound off: Width, Diff and Brightness. Width serves as an indicator when a groove bottom has a too large or small width. Diff is the maximal difference between two successive edges

in the time domain. Brightness will cut of edges where the difference between the two of them is too low. Depending on the " or " or " and " option edges which fulfill one or two criteria will be cut.

5. An edge detection algorithm can be chosen. Moreover the parameters can be varied.

6. Allows automated processing of multiple input files.

7. Options for the manual tracking.

8. Displays how many cuts have been made.

9. Allows for the final result to be played inside the application.

The parameters for the fifth subsection [Fig: 11] will be described in detail in another part of the report, part of this project will be the determination of the optimal values of these fields. The second section [Fig: 10] displays the currently loaded image [Fig: 12] in a reduced version. Once the program is running, the lines will be updated automatically to indicate the progress.

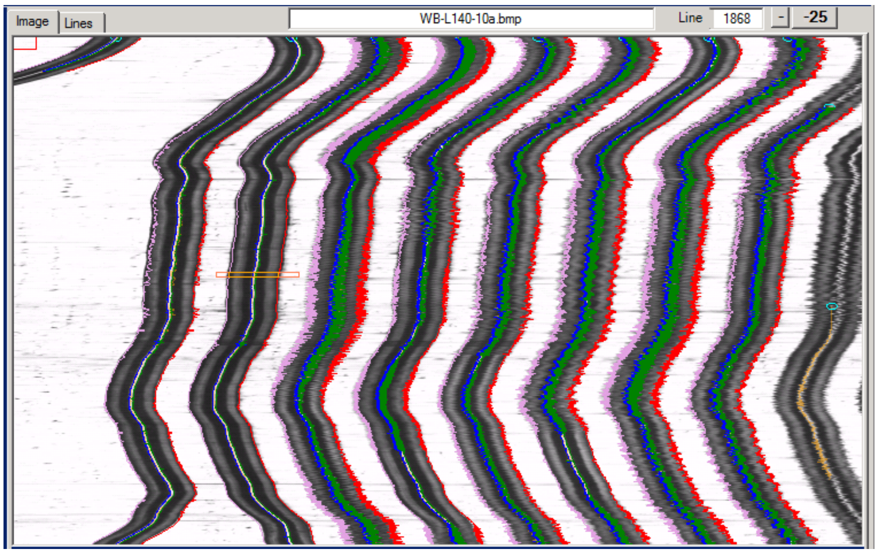


Figure 12: Close up of the currently loaded image section

The colored lines indicate where an edge has been detected. There are five lines drawn for each groove. Pink is the left border of a groove, red indicates the same at the right side. Orange shows when a groove has been detected by the same algorithm which detects the pink and red line, not every part of a groove will be analyzed in detail. Blue indicates the left and green the right side of the groove bottom. A cursor can be placed anywhere on the image [Fig: 12] , it will automatically update the three graphs below [Fig: 13] .

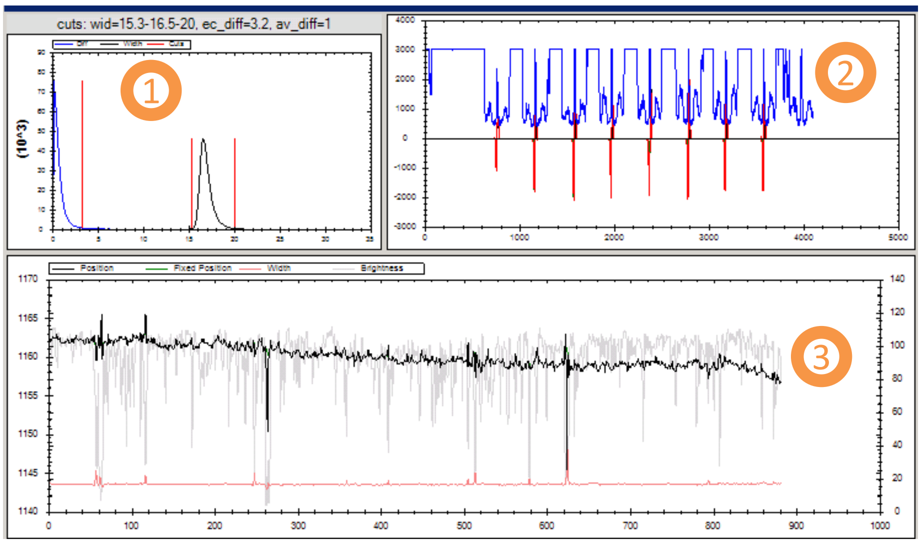


Figure 13: Three graphs below the image

1. Histogram displaying the width distribution of the groove bottom in black color. The blue line is a histogram of the differences between two successive edges. The red lines indicate where the cuts will be made.
2. Displays the cross-section of the entire image, where the red line indicates where the edges have been detected.
3. The red line is a graph of the bottom width in the displayed section of the image, the scale is on the right side. The red line is the position of the edge on the image; the scale is on the left side. A green line indicates that the position of the edge has been fixed by the cut algorithm. The brightness has been scaled down and has is not directly related to any scale.

2.1.3 Workflow

The workflow describes the steps RENE performs when of it is instructed to extract sound from an image. The different steps could of course be more detailed. But the eight steps presented in this sub chapter sum up the most important phases of the transformation. This flowchart [Fig: 14] does not include the steps necessary for the minimization process.

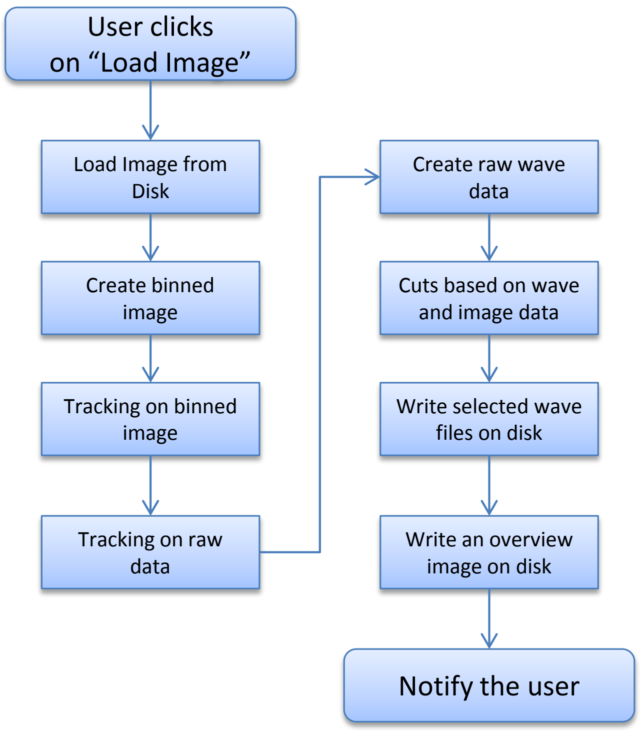

User clicks on "Load Image"
Load Image from Disk
Create binned image
Tracking on binned image
Tracking on raw data
Create raw wave data
Cuts based on wave and image data
Write selected wave files on disk
Write an overview image on disk
Notify the user

Figure 14: Visualization of the RENE workflow

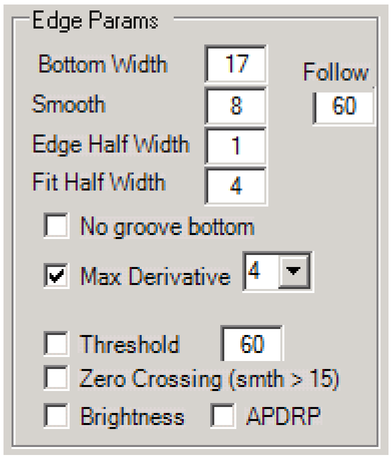
It is important to keep in mind the order of these steps, if not taken into account they can severely distort test result and other outputs. For example if one is testing a different edge detection algorithm, the output should be tested before the cuts are made. The cuts would remove the most obvious errors; this would lead to similar final data for different intermediate results.

2.1.4 Output

As described on the previous pages, the format of the output can be selected manually by the user. Whatever option the user will choose, mono, stereo, 16 or 24 bit, the output will be a wave file. It contains a customized header, which contains the parameters used during the extraction. Similar information is also being stored in the filename.

2.2 Edge Detection Algorithms

This is a short summary of the currently available edge detection algorithms in the user interface. A deeper understanding of the algorithms and their parameters is necessary to define constraints for the minimization process. Moreover not every algorithm takes into account each parameter of the interface. The section of the interface which contains these parameters is called "Edge Params" [Fig: 15] .


Edge Params

Bottom Width 17
Smooth 8
Edge Half Width 1
Fit Half Width 4
Follow 60

☐ No groove bottom
☑ Max Derivative 4
☐ Threshold 60
☐ Zero Crossing (smth > 15)
☐ Brightness ☐ APDRP

Figure 15: Section containing the edge detection parameters

There are two categories of parameters. The first category is the edge detection algorithms which can simply be selected with check box. The second are the fields which contain numbers and represent parameters of one or more algorithms. Five algorithms are available:

- No groove bottom
- Max Derivative
- Zero Crossing
- Threshold
- Brightness

Those which are of interest to this project will be described in a sub chapter.

2.2.1 Max Derivative

This algorithm is available in four different versions; they can be selected next to the algorithm. The latest version which preceded this project is number four, it also the only one which delivers usable results. It is based on the derivative of the intensity function. For every groove, the point where the derivative takes the maximal or minimal value will be considered an edge. The technique is visualized in the next image [Fig: 16] .

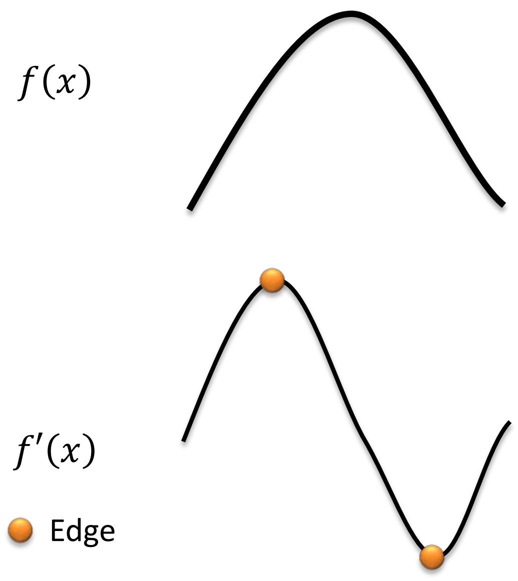


Figure 16: Max derivative edge detection

The function $f(x)$ depicts the intensity graph where the groove bottom is located, $f'(x)$ is the derivative of this function. It has a local minimum as well as a local maximum which will be considered edges. Both curves are only approximates of a real situation, the real derivative contains a saddle point between the two extremes. The figure does not yet contain the influence of the parameters. The max derivative algorithm will try to fit a parabola with the length of the bottom width parameter to each pixel; the most likely position will be taken as edge for the final step.

The first parameter is *bottom width*, it defines how wide the groove bottom is supposed to be. The unit of measurement is pixels.

The second parameter is *smooth*, it defines over how many pixels an average will be made. This will eliminate local spikes in intensity which could lead an edge to jump to a local peak, and introduce noise in the process. The average is done linearly.

The third parameter is called *follow*. When the software does the tracking, this parameter will weigh the current position compared to the new one.

The fourth parameters name is *fit half width* and fixes the size of the curve which will be fitted over the two extremes. The size of the curve can be calculated by taking the parameter, multiplying by two and adding one. [Fig: 17] .

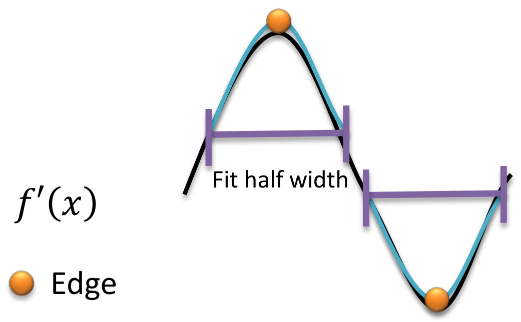


Figure 17: Visualization of the fit half width curve

Whereas *fit half width* is only half the length of the violet bar [Fig: 17] . The width of the curve is calculated in the following manner: $fithalfwidth * 2 + 1$. The unit of measurement is once again in pixels.

2.3 Spectrum Analysis

The analysis of spectral properties of a given sound file is necessary to distinguish silent from other parts in a wave file. This requires knowledge about the activity in a certain range of frequencies for arbitrary parts of a sound file. A sound file in the wav format contains only information in the time domain, this information has to be translated into the frequency domain. It has to be considered that these two domains have a dual nature comparable to those of the position and impulse of a particle. An interesting fact for spectral analysis is that the range which an average human is capable of hearing. Which is between 20-20,000 Hz[7]. But this capability decreases as the age progresses, which results in a range of 20 to 14,000 Hz for an average middle aged person[7]. The range of instruments is even narrower, most of them do not exceed 5,000 Hz[8]. The human voice is similar and ranges up to 7,000 Hz[9]. These facts allow us to make an assumption that most energy in parts which contain sound will be below 7,000 or 5,000 Hz, depending if an instrument or a voice is used.

2.3.1 Discrete Fourier Transformation

In order to transform a digital signal from the time into the frequency domain a Discrete Fourier Transformation has to be applied. In the case of this project, the signal is finite which implies further consequences. Those will be treated later on. The upper limit frequency of the Fourier transformation depends on the sampling rate, it is half of the maximum sampling rate s [10].

$$f_{max} = \frac{s}{2}$$

The resolution of the Fourier transformed signal then depends on the number of samples N which were taken from the original signal. Where the frequency increment δf is

$$\Delta f = \frac{f_{max}}{N}$$

This will be important when extracting the energy of a certain frequency range. Another application is locating the peak intensity. The Discrete Fourier Transformation $X_d(n)$ of a signal $x(k)$ with length N and offset of k_0 can then be calculated by the following formula:

$$X_d(n) = X(f = n * \Delta k) = \sum_{k=k_0}^{k_0+N-1} x(k) * e^{-j2\pi \frac{nk}{K}}$$

Often we do not want to transform the entire signal at once, especially if the subsections have to be analyzed for specific properties. In this case a windowing function has to be applied to the signal in order to restrict side effects which will impair the quality of the transformation. The case where N samples are taken and the rest is set to zero corresponds to a rectangular window. This might introduce side lobes, depending on the frequencies and the window size. The standard approach is to multiply the values with a windowing function before they are sent to the Fourier transformation. A few short tests have shown that for this project the Hann(Hanning) window shows the best results. For a sample $x(k)$ each sample has to be multiplied by the value of the function $w(k)$:

$$w(k) = 0.5 * (1 - \cos(\frac{2\pi k}{N - 1}))$$

The sides of this function approximate zero; this reduces the side lobes but requires that the samples overlap. An optimal overlap percentage is hard to define; therefore it will be tested in the designated section. Sound Forge Pro uses 75% as a default value.

2.3.2 Wavelets

The Fourier Transformation is a useful tool for the analysis of a stationary signal, it will be decomposed into a series of sinusoids[5]. A Wavelet Transform decomposes a signal into a family of wavelets. Sinusoids are regular, smooth and symmetric. Wavelets exist theoretically in an infinite number of forms[12], they are not restricted to these three properties. Moreover Wavelets have a finite duration as opposed to an infinite one of sinusoidal signals. Signals or time series which have to be analyzed by a Wavelet Transformation must have a size N which is a multiple of two $N = 2^k$ where $k = 1, 2, 3....$

For each step of a Wavelet Transform with T elements (where $T = N$ at the beginning), it calculates $T/2$ averages and $T/2$ coefficients. The coefficients will be stored in the latter half of the array, and the averages will be used as input for the next step. These recursive iterations continue until a single average and coefficient are left. The averages a_i in the case of the Daubechies T4 Wavelet are calculated the following way:

$$a_i = h_0 s_{2i} + h_1 s_{2i+1} + h_2 s_{2i+2} + h_3 s_{2i+3}$$

Where s is the signal and the scaling function coefficients h are defined as follows:

$$\begin{aligned} h_0 &= \frac{1+\sqrt{3}}{4\sqrt{2}} \\ h_1 &= \frac{3+\sqrt{3}}{4\sqrt{2}} \\ h_2 &= \frac{3-\sqrt{3}}{4\sqrt{2}} \\ h_3 &= \frac{1-\sqrt{3}}{4\sqrt{2}} \end{aligned}$$

The new coefficients will be calculated similarly:

$$c_i = g_0 s_{2i} + g_1 s_{2i+1} + g_2 s_{2i+2} + g_3 s_{2i+3}$$

Where the wavelet function coefficients are defined as:

$$\begin{aligned} g_0 &= h_3 \\ g_1 &= -h_2 \\ g_2 &= h_1 \\ g_3 &= -h_0 \end{aligned}$$

The calculation will shift by two places for each step until the end of the dataset is reached.

2.4 Minimization

Minimization is the task of finding the smallest value taken on by a function with one or more parameters. This problem appears in numerous fields and industries such as Mechanics, Economics and Particle Physics. Even if the subject will be referred to as minimization, the same procedure applies to maximization, only the sign changes. A more general expression would be optimization, this term covers both cases. A formal definition of this task would be

For $f(x_1, x_2, \ldots, x_n)$ find $x_1, x_2, \ldots, x_n$ which minimize f

This is a fairly general definition and does not yet impose any restrictions on the parameters; this will be done later on. The minimum of a one dimensional function, a function with only one parameter, has according to the fundamental theorems of calculus one of the following properties (definitions from the Minuit tutorial[1]):

1. The derivative $\frac{\partial F}{\partial x} = 0$ (stationary point), or
2. the derivative $\frac{\partial F}{\partial x}$ do not exist (cusp), or
3. the point x is on the boundary of the allowed region (edge point).

The problem is that any number of points which fulfill one of these criteria may exist. If the function is unknown the problem becomes even larger. Therefore the task of localizing a global optimum is abandoned; a local minimum has to be sufficient. A local minimum around x_0 with the precision p may be defined to be

$$f(x_0) < f(x_0 + p)$$

and

$$f(x_0) < f(x_0 - p)$$

The definition can be generalized to N dimensions by simply applying the restriction to each variable. The approach to find a local variable is simple in one dimension, vary x in small steps until it decreases no further. The case in multiple dimensions is as usual more complicated. However for this project certain restrictions apply:

$$\text{The function } f(x_1, x_2, \ldots, x_n) \text{ is not known in analytical form} \tag{1}$$

$$\text{For some } x_i \text{ where } 1 \leq i \leq n \text{ and } x_i \in \mathbb{Z} \tag{2}$$

$$\text{For some } x_i \text{ where } 1 \leq i \leq n,\ x_i = \{n, \ldots, m\} \text{ and } n, m \in \mathbb{Z} \text{ and } n < m \tag{3}$$

From (1) it follows that the derivate and therefore the gradients have to be estimated using a set of points. This is time consuming and error prone, nonetheless a closer look will be taken in the section covering the gradient based optimizer. From (2) and (3) it follows that the function is not continuous. In other words, it is a discrete function because the input values are only integers and not real numbers. Where (3) is just a special case of (2). Due to the discrete nature of the function, the evaluation of all possible input parameters is in theory a possibility to locate minima. However in practice this depends on the restrictions which can be placed on some of the parameters. If the range for even one variable is too large, the calculation time will skyrocket and render this approach unusable. Another important factor is the necessary time for one function evaluation. Tests have shown that the evaluation of one set of parameters will easily take one second; this seriously limits the number of evaluations which can be done in a reasonable time. An example would be the following search space:

- Parameter A: 5 options
- Parameter B: 4 options
- Parameter C: 4 options
- Parameter D: 4 options
- Parameter E: 15 options

This would be a typical example for the Generic Kernel Edge detector which will be introduced later. Although none of the parameters has a wide range, the combination of all of them results in a staggering number of function evaluations: $5 * 4 * 4 * 4 * 15 = 4800$. And this is only an example with very restrictive parameter ranges.

2.4.1 Nelder–Mead Method

The Nelder-Mead method is also called downhill-simplex or amoeba method and is a non-linear optimization technique for N dimensional problems[6][2]. It is a heuristic method but requires relatively few function evaluations. However it can only progress linearly which might lead to problems if the seed point is far away from the minimum. It can be imagined as an animal with $N+1$ feet which looks for a cool spot, where the temperature is given by $f(x)$. It takes into consideration three feet at each step. The hottest (P), the second hottest (Q) and the coolest (R). For each turn the feet are reevaluated and the P, Q and R are reassigned to other feet if necessary. There are four possible moves which the amoeba can make: reflection, expansion, 1-dimensional and N-dimensional contraction. These cases are best illustrated by an example [Fig: 18] where $N = 2$, which results in an amoeba with three feet.

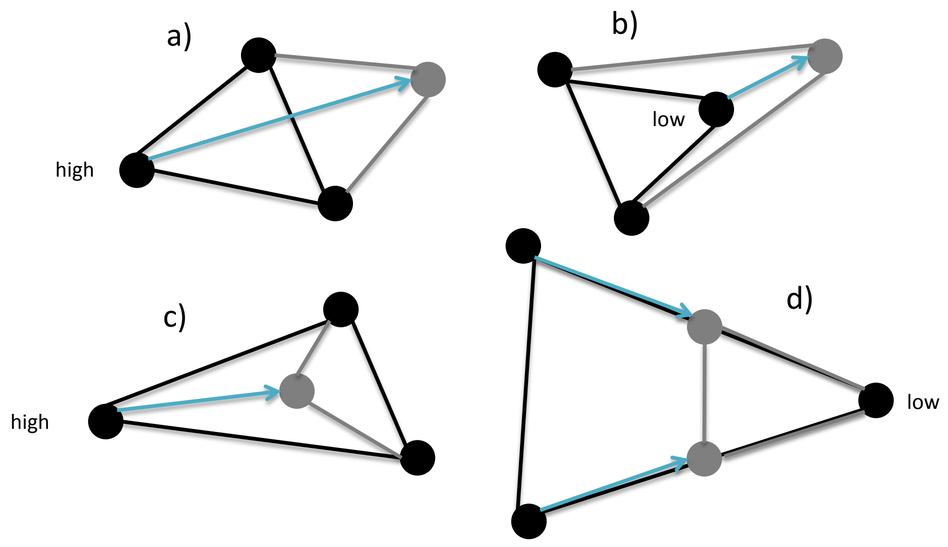


Figure 18: The four possible movements, example in 2 dimensions

Where *a)* describes a reflection away from a high point, *b)* is an expansion beyond a low point, *c)* depicts a 1-dimensional contraction away from a high point and *d)* is an N-dimensional contraction towards a low point. Now that the possible moves have been described, the question remains how they will be coordinated and when they stop. The latter question is simply answered; there are two possible criteria for a stop. Either the amoeba converges, which means the difference between the highest (P) and the lowest point (R) is below a given threshold. Or the maximal number of function evaluations has been reached.

The behavior for one turn can contain up to nine different steps. It is best visualized in a flowchart [Fig: 19] .

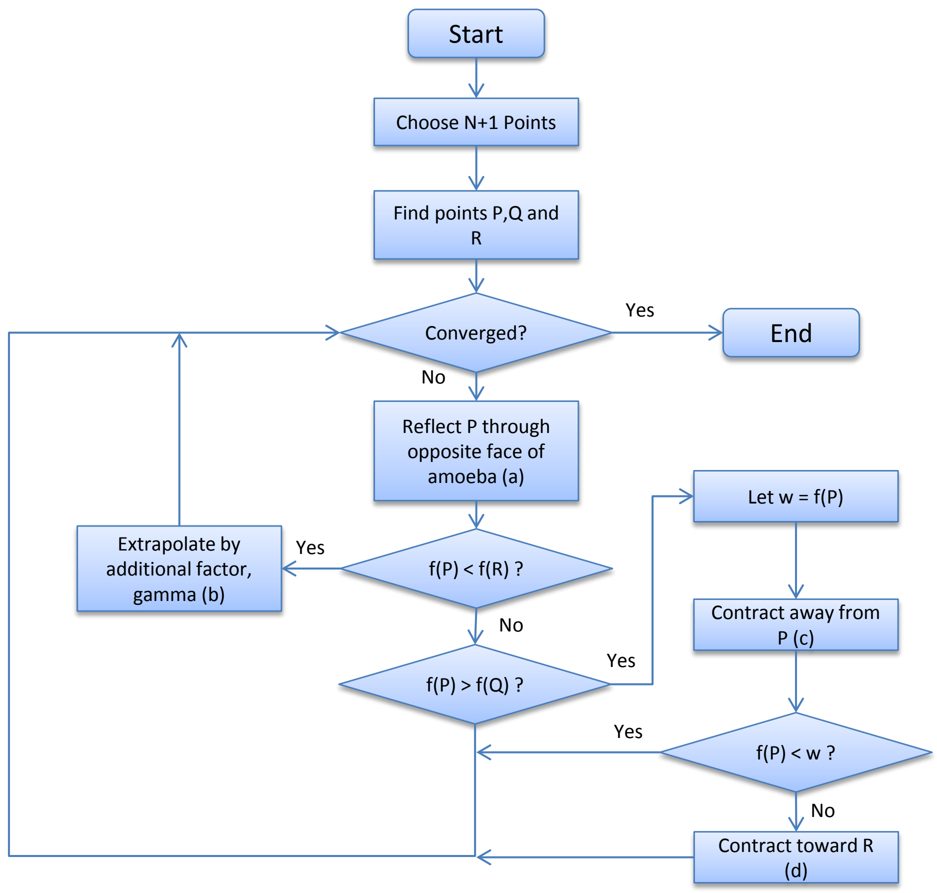


Figure 19: Behavior of the amoeba

The flowchart [Fig: 19] incorporates the four possible movements we have seen in the previous image [Fig: 18] . However the algorithm is not fool proof, there are certain cases where it might fail. For example if the $N + 1$ points become aligned, the amoeba will only move in an $N - 1$ dimensional hyper plane. Despite this problem, the algorithm has a relatively simple implementation and will be used in the course of this project. The simplicity allows easier changes for the restrictions which will be in place for our function $f(x)$.

2.4.2 Conjugate Gradient Method

The conjugate-gradient method is as the name indicates, a gradient based optimizer. But in order to calculate gradients, we need to calculate derivatives first. In the case of this project, the function is not analytically known. The derivatives have therefore to be approximated. The simplest approximation would be [1]

$$\left.\frac{\partial F}{\partial x}\right|_{x_0} \approx \frac{F(x_0+d)-F(x_0)}{d}$$

Where d is a small step. The error will be the lowest order in the Taylor expansion[1]

$$\delta \approx \frac{d}{2} * \left.\frac{\partial^2 F}{\partial x^2}\right|_{x_0}$$

This means it is favorable to choose d as small as possible, but not small as the rounding error in F becomes significant. A safer method would be to use symmetrically chosen points in the neighborhood of x_0

$$\left.\frac{\partial F}{\partial x}\right|_{x_0} \approx \frac{F(x_0+d)-F(x_0-d)}{2d}$$

This has the disadvantage that another function evaluation is required, but the error δ falls to the second order. As we are now able to calculate the gradient, the conjugate gradient method can be applied.

Explaining in detail the inner workings of this algorithm would go beyond the scope of this report. A detailed derivation can be found in one of the referenced documents[3]. Because we can reasonably assume linearity of the function, we have to use the nonlinear version of the algorithm. This translates to applying the linear version, but using the gradient as residual. The residual was used to indicate how far the current point was from the solution. Moreover the whole process has to be combined with a line search method[4].

2.5 Simple Random Sampling

Simple Random Sampling is the most elementary form of sampling a population called sample surveys. In statistics sample surveys are of tremendous importance when trying to obtain information of a large population by examining only a fraction of the total. The general ideas and formulas have been taken from the book "Mathematical Statistics and Data Analysis"[11]. This sampling technique is probabilistic in nature because every member of the population has a specific probability of being included in the sample. This guarantees the unbiasedness. Random Sampling has a number of advantages:

- A small sample costs less, in this case computation time.
- Random selection is a guard against investigator biases; one could be tempted to declare certain regions as more representative than others.
- Random sampling makes the calculation of an estimate of the error possible.

- A sample can be designed to have a predetermined error level.

For this project, the main use of simple random sampling will be the estimation of the mean μ of the bottom width. The sample size will be denoted by n, the population size as N and the value of the sample members by $X_1, X_2, ..., X_n$. Because X_i is a random variable, the sample mean is a random variable which distribution depends on those of the sample members. The sample mean is considered

$$\bar{N} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

The probability distribution of $\bar{N}$ will determine how accurately it estimates μ. In general, the more tight the distribution is centered on μ the better the estimate. Now we have a way of calculating an estimation of the mean, but we have no idea how accurate it is. The accuracy can be estimated by using the standard error. The standard error of a sample mean with replacement can be estimated in the following manner

$$\sigma_{\bar{X}} = \frac{\sigma}{\sqrt{n}}$$

The problem is that usually the standard deviation is unknown, this also holds true for this project. This means that the population variance has to be estimated too. The sample standard deviation is estimated the following way

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2$$

Which is almost equal to the calculation of the variance of a sample, with the difference that in order to ensure unbiasedness, the factor has to be adapted to $\frac{1}{n-1}$. To gauge the variance of our estimated mean, the estimated variance can be calculated with the following formula:

$$s_{\bar{X}}^2 = \frac{s^2}{n}$$

These equations are everything which is necessary to introduce simple random sampling for parameters of the edge detection algorithms.

2.6 Edge Detection

Edge detection is the process of locating discontinuities in an image. A discontinuity is a place where the intensity changes quickly in an image. This will highlight areas where which exhibit these characteristics. Usually the process of edge detection consists of applying a kernel to each pixel. Multiple edge detection algorithms exist, with various degrees of complexity. Most of them are discrete differentiation operators, which mean they calculate an approximation of the derivative of the image for each pixel. The derivatives are calculated in both directions and then combined to calculate the direction and the magnitude of the gradient. This approach is taken by the most common edge detectors such as the Sobel and Prewitt operator as well as Canny and Roberts Cross. In order to create a customized edge detector, the functionality of an existing one will be analyzed first.

2.6.1 Sobel Operator

The Sobel operator is one of the most commonly used edge detectors. The Canny edge detector is based upon it, and the Prewitt operator executes the same operations but with a different kernel. The goal is to calculate the gradient at each point of the image. The underlying function is $z = f(x, y)$ where x, y represent the coordinates of the image and z the corresponding intensity. The function is of course not continuous, which means the derivate can only be calculated approximately. The gradient consists of the two derivatives, their approximations are calculated by applying the kernels (4) and (5) to the image. Where (4) represents the horizontal and (5) the vertical approximation of the derivative.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 2 \end{bmatrix} * A \tag{4}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * A \tag{5}$$

The two results are then combined as in (6) to calculate the magnitude of the gradient at each point in the image.

$$G = \sqrt{{G_x}^2 + {G_y}^2} \tag{6}$$

The higher the magnitude of the gradient, the higher the intensity changes at the specific point in the image. This is usually a strong indicator for an edge. The information contained in G_x and G_y can also be used to calculate the direction of the gradient. But as this property will not be used in the project we will not look into it. The original Sobel operator is useful if we want to identify all edges and their absolute intensity. But both of this properties are not what will be needed to successfully detect the groove bottom, this is why the algorithm has to be customized.

2.6.2 Customized Edge Detection

The customized edge detector will take into account only the horizontal changes, as the groove bottoms happen to be nearly vertical in each image. Moreover the absolute intensity is not necessary and therefore it will not be calculated. This means the kernel is applied once and the output will be the derivative at each point of the image. In order to provide parameters for the optimization algorithm, the kernel size should be able to be adapted automatically as well as the values inside the kernel. This is achieved by using a multiplication factor for the horizontal and vertical values inside the kernel. They define how much weight will be given the values which are farther away from the center. Or put another way, these two values will define how quickly the values grow in each direction within the kernel. Summarized the kernel has the following four parameters:

1. Horizontal kernel size $h = (n * 2) + 1$
2. Vertical kernel size $v = (n * 2) + 1$
3. Horizontal multiplication factor $i > 0$
4. Vertical multiplication factor $w > 0$

For example, if we generate a kernel with the values $(3, 3, 2, 2)$ we will have the following matrix:

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 2 \end{bmatrix} \tag{7}$$

This is exactly the Sobel operator. But the principle is best shown in a larger kernel which was created with the values $(5, 5, 2, 3)$:

$$\begin{bmatrix} -1 & -2 & 0 & 2 & 1 \\ -3 & -6 & 0 & 6 & 3 \\ -9 & -18 & 0 & 18 & 9 \\ -3 & -6 & 0 & 6 & 3 \\ -1 & -2 & 0 & 2 & 1 \end{bmatrix} \tag{8}$$

The kernel size and the multiplication factors can be adapted freely as long as the values remain positive. However if the height of the kernel is higher than one, a vertical averaging will be done which acts as a low pass filter on the final sound file. Moreover does a pixel not represent the same distance on the horizontal and vertical axis on a real record.

2.7 Current State of IRENE

Currently the edge detection and the whole image processing part are executed with parameters which are given by the user. The output is not rated automatically nor do any of the settings adapt to the different types of the record. This means for each record the best settings have to be found manually by rerunning the whole sound extraction process over and over again. The results of the different runs have then to be analyzed manually in specialized software such as Sound Forge which can analyze the spectral properties of those sound files.

2.8 Desired State of IRENE

The goal of the project is to have the possibility to automate the detection of the optimal parameters according to different criteria. The criteria can be chosen by the user, an example would be a maximized Signal Noise Ratio between the quiet and the non-quiet part. The results will automatically be analyzed by the software; no human interaction is required to find the optimal parameters.

The analysis chapter had a lot of ground to cover; however the introduction of all this topics is necessary to gain an understanding of the entire project.

3 Implementation

This chapter covers the implementation; it describes how the theoretical concepts outlined in the previous chapter have been applied to tackle the specific problems of this project.

3.1 Discovery of quiet parts

The quiet parts at the beginning of a disk are called lead-in grooves. It is a part where the needle has been put on the record, but no sound is recorded yet. This part has a low energy in the frequencies which represent voice and music. Moreover the peak activity will be in a frequency bin which is higher than the audible range. This is due to the fact that white noise is uniformly distributed in the frequency domain before the derivative is taken. Once the derivative is calculated for the wav file, the noise will rise in the frequency domain steadily by 6dB per octave. Which means the peak intensity will be at the highest frequency. The following image [Fig: 20] visualizes the difference between the two parts. It contains a spectrum analysis of one second of sound and silence. Where the silent part is not really silent, but contains a broad noise spectrum. The graph is logarithmic in both directions.

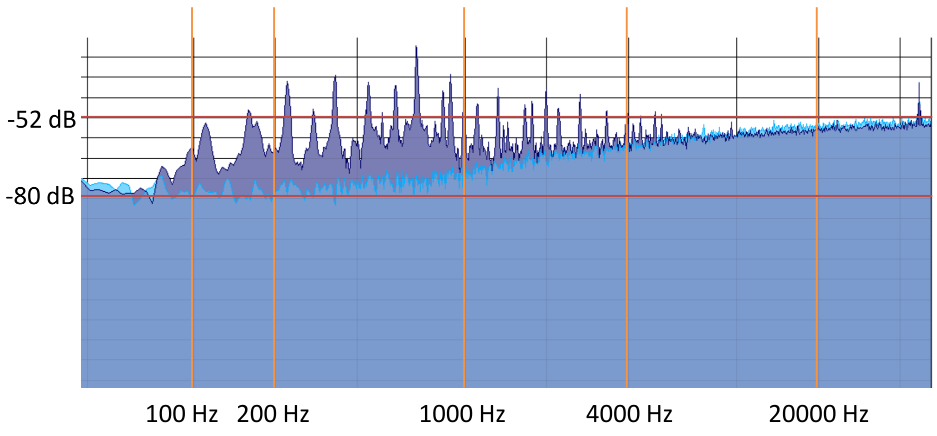


Figure 20: WB Record, one second of silence (blue) and sound (violet)

Not clearly visible are the frequencies with the most energy, in this case the intensity peaks at the following frequencies:

- Quiet part: -39 dB at 46,804 Hz
- Sound part: -30 dB at 663 Hz

This observation holds true for other records as well as for other samples in the same region of each record. This means we have now have two means of detecting the transition from silence to sound.

3.1.1 Energy Peak

The first criterion is the energy peak of a sound sample. The energy peak is found by transforming a sample of sound into the frequency domain with a Fourier transformation. This process is repeated for, partially overlapping, samples throughout the entire sound file. The sample size has to be sufficiently large to not lose precision. If the sample is too small, the frequency resolution becomes crude and the peak might be detected wrongly. Moreover the sample size has to be sufficiently small to detect the transition accurately. In the course of this project the window sizes used were between 4096 (2^12) and 65536 (2^16) where the sound files have sampling rate of 104,000 Hz. The process consists of detecting the peak frequency for each sample, calculate an average peak frequency for the entire file and run through the file from the beginning while the peak frequency is above average. Once the peak frequency drops, the transition has been detected. There are however certain pitfalls for this approach. There is always the possibility of an outlier which might contain a below average peak which will mislead the program into detecting the transition prematurely. The process of detecting the transition can be visualized as follows [Fig: 21] :

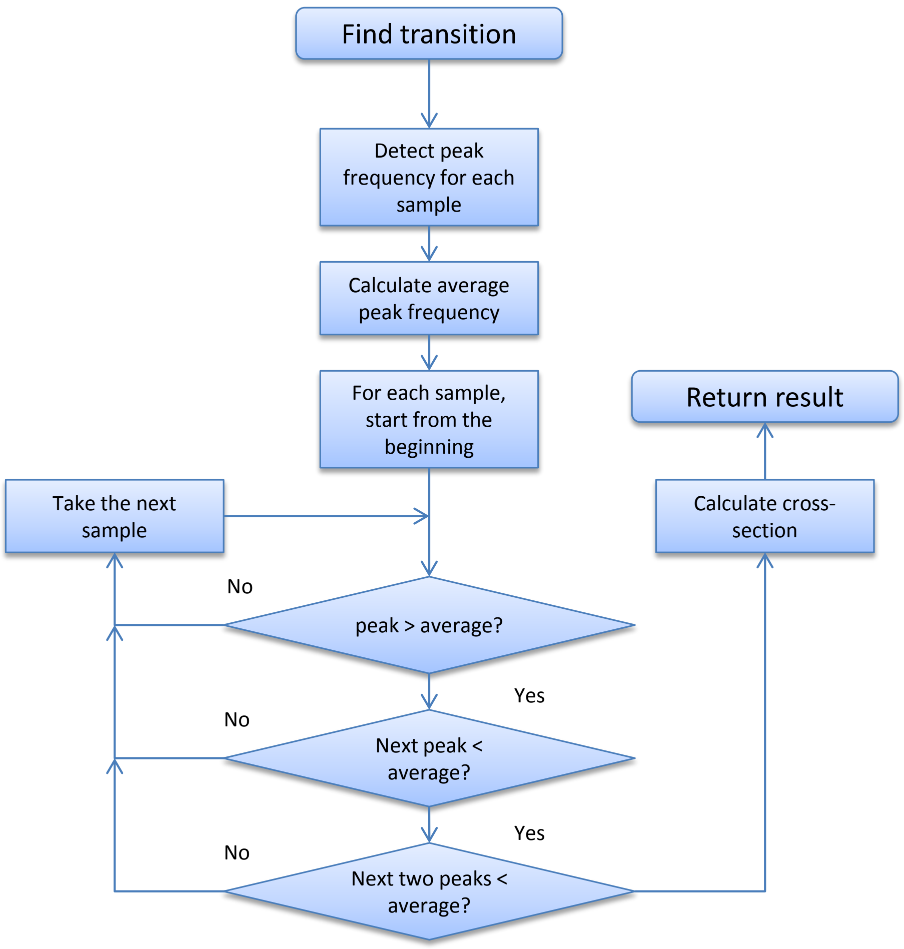

Find transition
Detect peak frequency for each sample
Calculate average peak frequency
For each sample, start from the beginning
Take the next sample
Return result
Calculate cross-section
No
peak > average?
Yes
No
Next peak < average?
Yes
No
Next two peaks < average?

Figure 21: Visualization of the transition detection

However if the sample size becomes smaller, even in reasonable steps, the probability for outliers rises. This leads to a significant error which grows quickly beyond an acceptable range. The countermeasure is to start at a sample size which is large enough to not contain outliers, and then using the result as a seed for the smaller sample sizes. This is done recursively until the intended sample size is achieved [Fig: 22] .

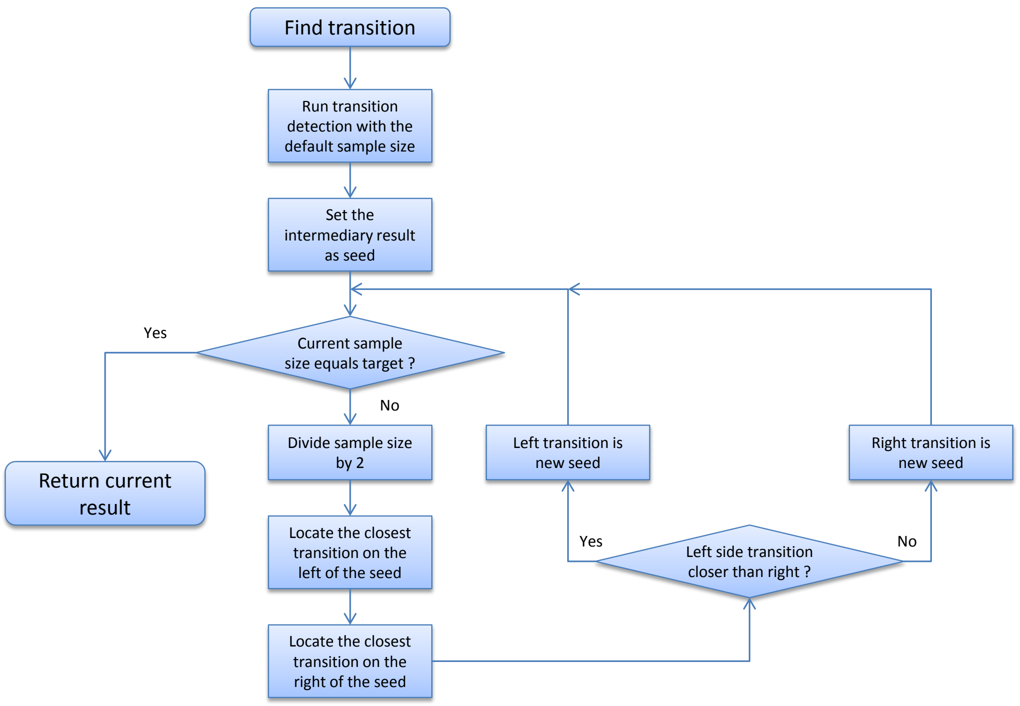

Find transition
Run transition detection with the default sample size
Set the intermediary result as seed
Current sample size equals target ?
Yes
No
Return current result
Divide sample size by 2
Locate the closest transition on the left of the seed
Locate the closest transition on the right of the seed
Left side transition closer than right ?
Yes
No
Left transition is new seed
Right transition is new seed

Figure 22: Visualization of the transition detection done recursively

The result of the transition detection is visualized in RENE itself. The following image shows a graph with the peak intensities for a large sample size for a sound file with a length of 7.5 seconds. The transition is represented by the red line [Fig: 23] .

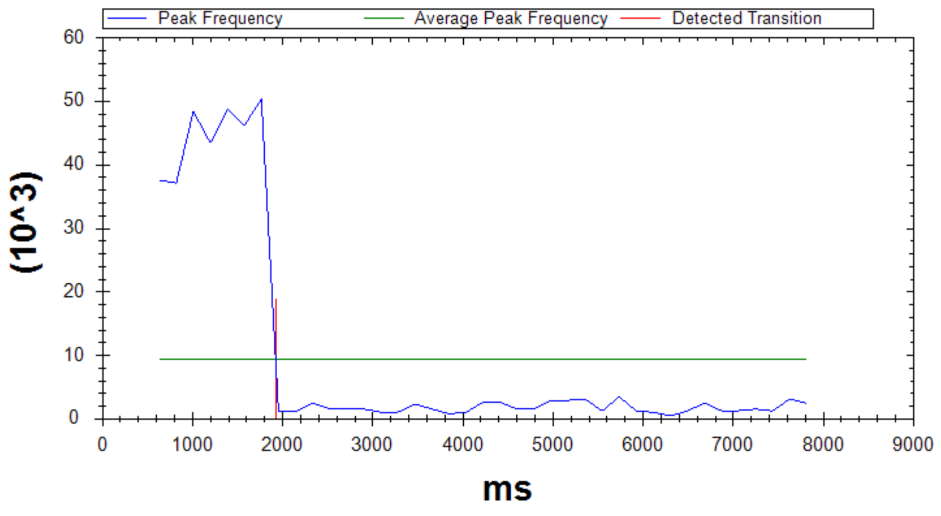


Figure 23: Sample size 64k, Sound Sample 3, 70% overlap

As we can see, the peak frequencies are well above the average for the quiet part and there are no outliers which could mislead the algorithm. However if the sample size decreases, it becomes more difficult to spot the transition [Fig: 24] .

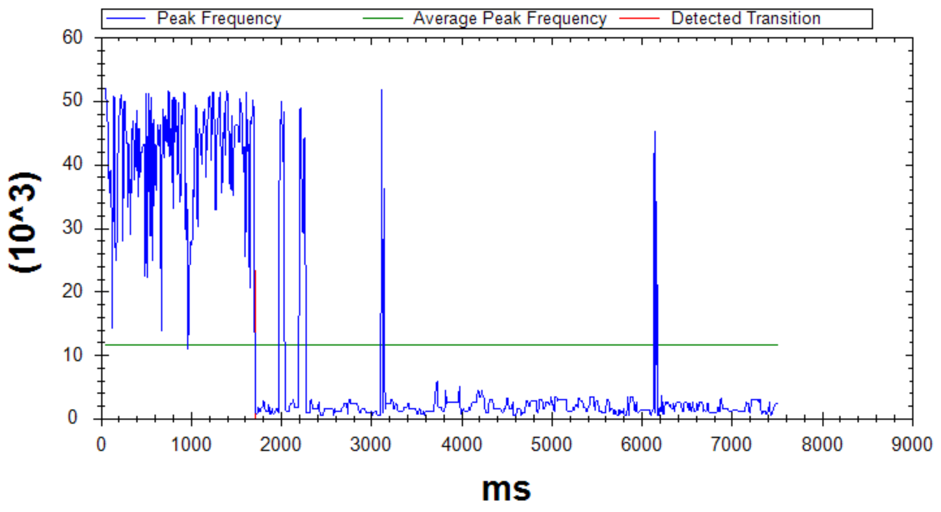


Figure 24: Sample size 12k, Sound Sample 3, 70% overlap

With the smaller sample size, the resolution increases, but so does the number of outliers. The record from which the sound was taken has a transition which is easy to detect. There are almost no outliers and the sound does not start gradually. This is the case in the following image which displays a different record with the same resolution. In this case the transition is not obvious at all [Fig: 25] .

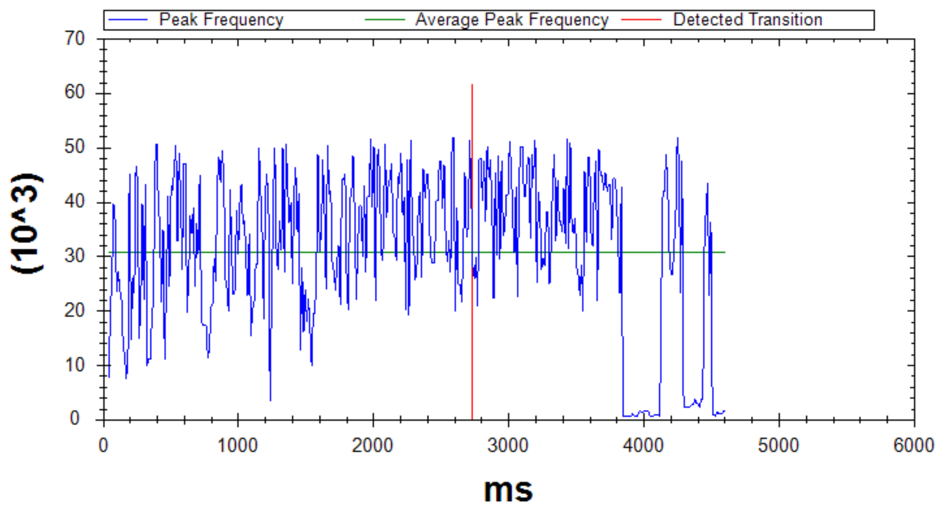


Figure 25: Sample size 12k, Sound Sample 7, 70% overlap

The only way the program was able to extract the position of the transition was by using the seed value given by the run with a large sample where the transition is more obvious [Fig: 26] .

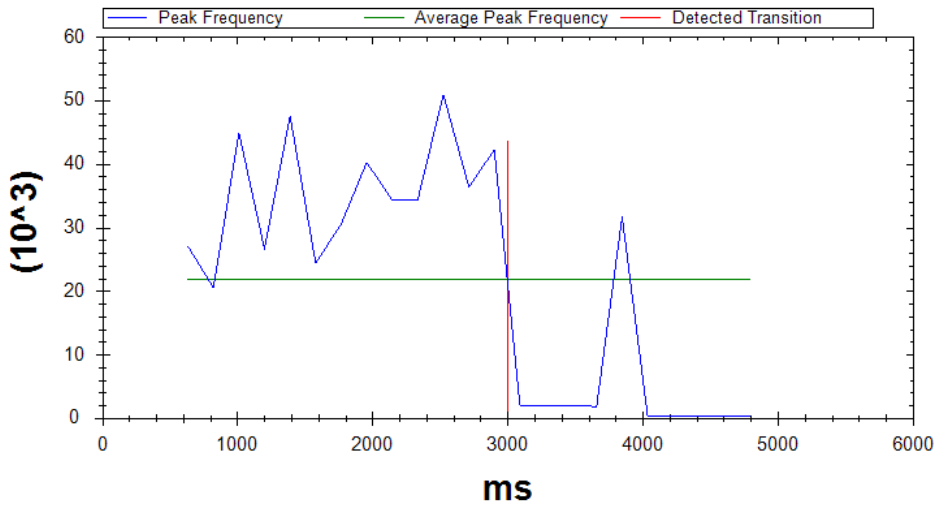


Figure 26: Sample size 64k, Sound Sample 7, 70% overlap

The result of the silence sound transition is then interpolated. A line is drawn from the last point above, and from the first point below the average. The point where this line crosses with the average is considered the transition.

3.1.2 Energy in the audible range

The second criterion is the energy contained in the audible spectrum. The first few steps of this criterion are the same, and they have the same limitations too. A Fourier transformation is applied to a sample and the energy in each frequency bin is calculated. But then instead of finding the frequency bin with the maximum intensity, the intensity of the audible range is summed up. Once this has been done to the entire file, an average is calculated and each sample will be classified according to the sum of its intensities [Fig: 27] .

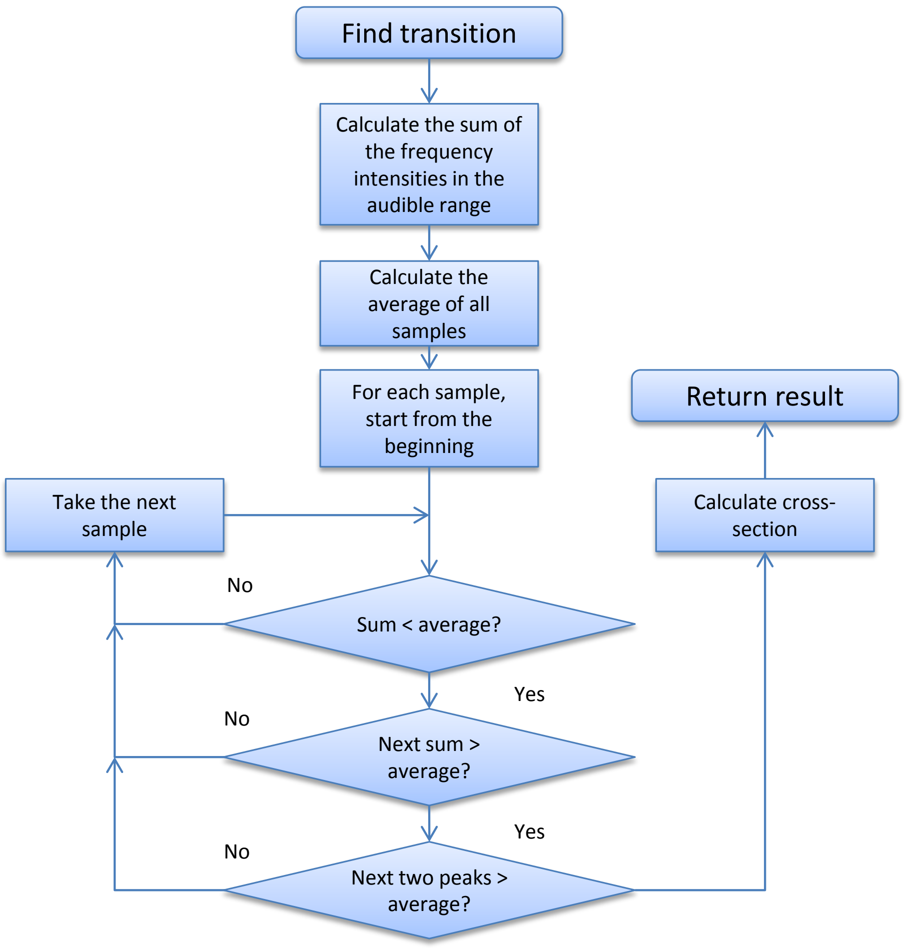

Find transition
Calculate the sum of the frequency intensities in the audible range
Calculate the average of all samples
For each sample, start from the beginning
Return result
Take the next sample
Calculate cross-section
No
Sum < average?
Yes
No
Next sum > average?
Yes
No
Next two peaks > average?

Figure 27: Visualization of the transition detection process

In theory this approach works well, but if tested against samples with a smooth transition, it performs badly. In the case of the third sample, where the transition is clearly visible it performs almost as well as the other algorithm [Fig: 28] :

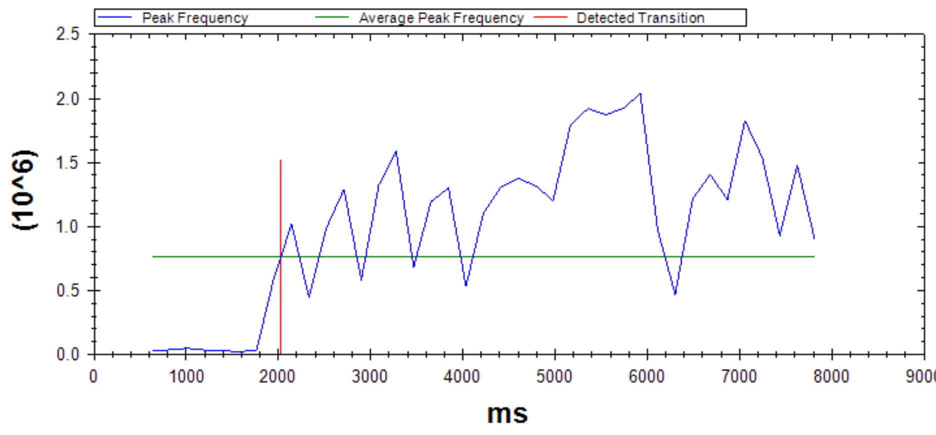


Figure 28: Sample size 64k, Sound Sample 3, 70% overlap

However if the transition is smooth, the detected transition is off more than two seconds compared to the previous approach [Fig: 29] . And this is in the simplest case where the sample sizes are large.

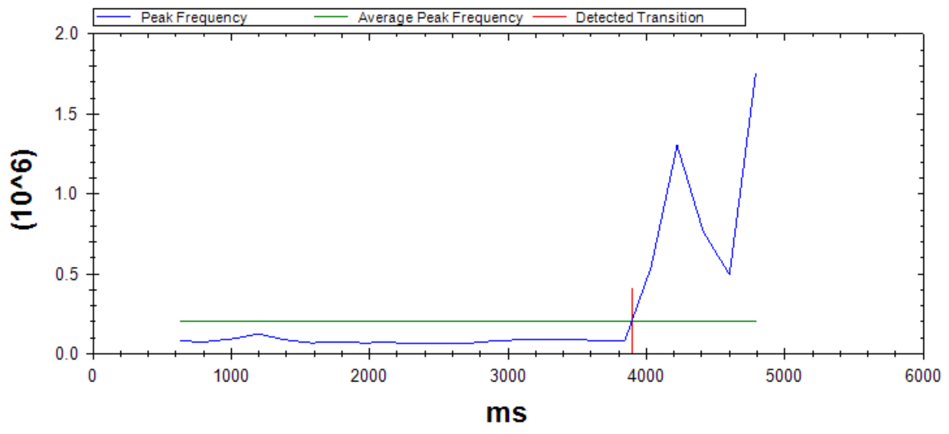


Figure 29: Sample size 64k, Sound Sample 7, 70% overlap

3.1.3 Denoising

After the transition has been roughly detected by the previous methods, the Wavelet denoising comes into play. The part of the sound file which contains the transition is denoised and then analyzed for a local minimum. The denoising is done by transforming the original sound sample into the wavelet domain. Then the Wavelet coefficients above a certain threshold are set to zero to eliminate the high frequencies. Now the transformation can be undone and the result is ready for further processing. The sample is approximately half a second long and is centered on the transition detected by the FFT. The resolution can be varied in order to avoid stumbling into a small local minimum. The goal is then to descend down the slope as long as the intensity decreases; this process is visualized in the next flowchart [Fig: 30] :

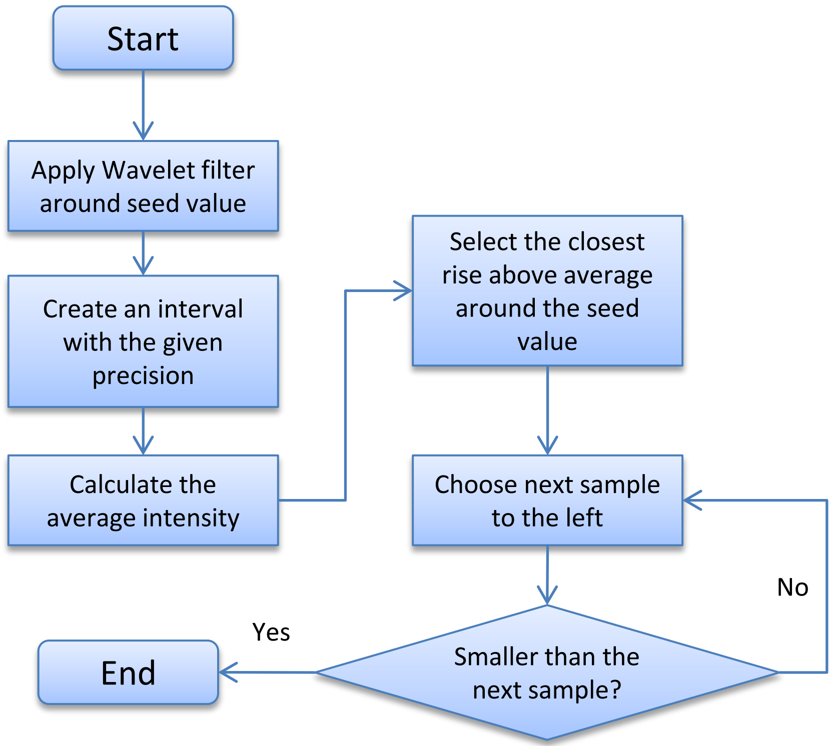

Start
Apply Wavelet filter around seed value
Create an interval with the given precision
Calculate the average intensity
Select the closest rise above average around the seed value
Choose next sample to the left
No
Yes
End
Smaller than the next sample?

Figure 30: Visualization of the Wavelet descent

The result is once again drawn into a graph in the application to visualize the progress. Using the same two input files as before, the results show once again that a smoother transition is a source of problems. The simpler case of a rough transition translates into a graph [Fig: 31] where the result is clearly visible.

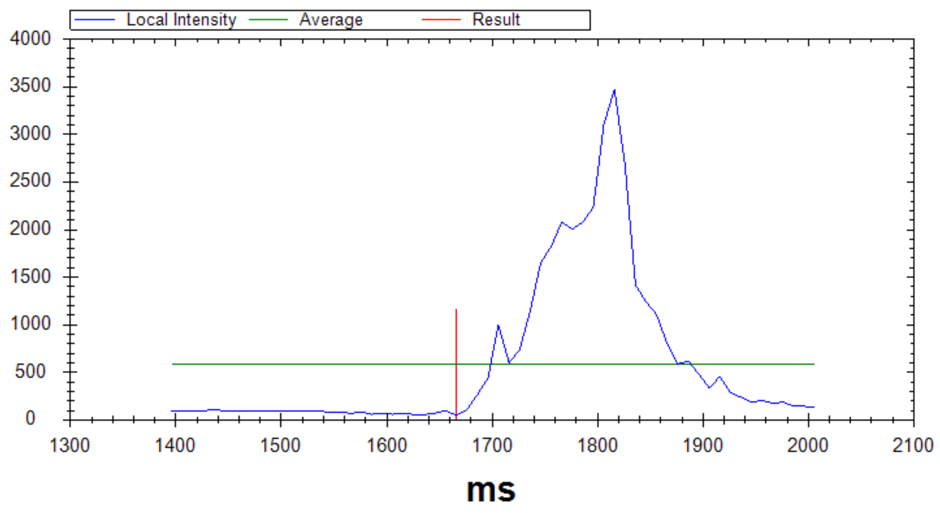


Figure 31: Sample size 64k, Sound Sample 3, 70% overlap

The algorithm started at the closest crossing point and descended to a local minimum which is represented by the red line. The closest crossing point is defined to be a line which has a positive slope and crosses the line which represents the average. In this case it is conceivable that the same result could have been achieved without the seed point from the previous steps. In the case of the second example, the result [Fig: 32] would have been completely different:

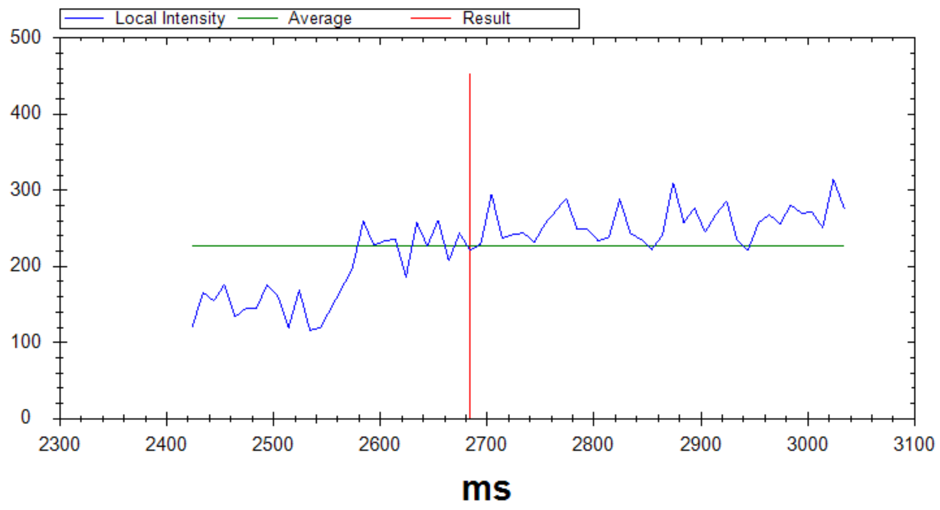


Figure 32: Sample size 64k, Sound Sample 7, 70% overlap

There are multiple crossing points and a broader test sample would bring to light even more difficulties.

3.1.4 Specifics of the Fast Fourier Transformation

The previous chapters mention the use of the Fourier transformation to translate the sound into the frequency domain. This process has certain pitfalls as documented in the analysis section. To reduce side lobes a Hann window is applied to each sample. The Hann window has been chosen because of the properties it showed in Sound Forge. It outlines the intensity differences in the audible frequencies best. Such a window attributes the weight to each sample as follows [Fig: 33] :

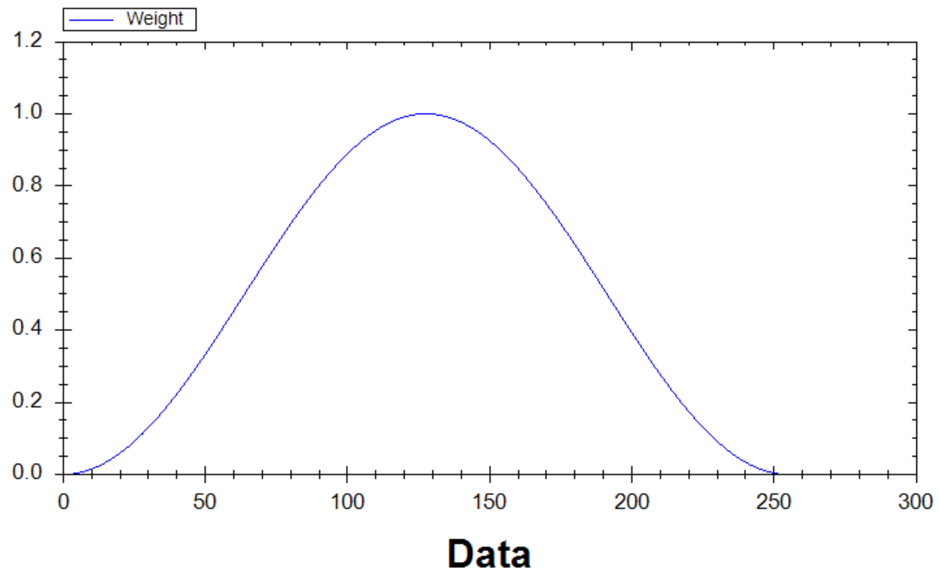


Figure 33: Simple Hann Window, width of 256

This window decreases sharply to the end; the values at the borders are zero. In order to avoid not taking into account some of the samples, each sample which is FFT transformed overlaps to a certain percent with the previous one. The tests will show that 70% delivers on average the best results. This process [Fig: 34] is best visualized in an image showing the overlapping Hann windows:

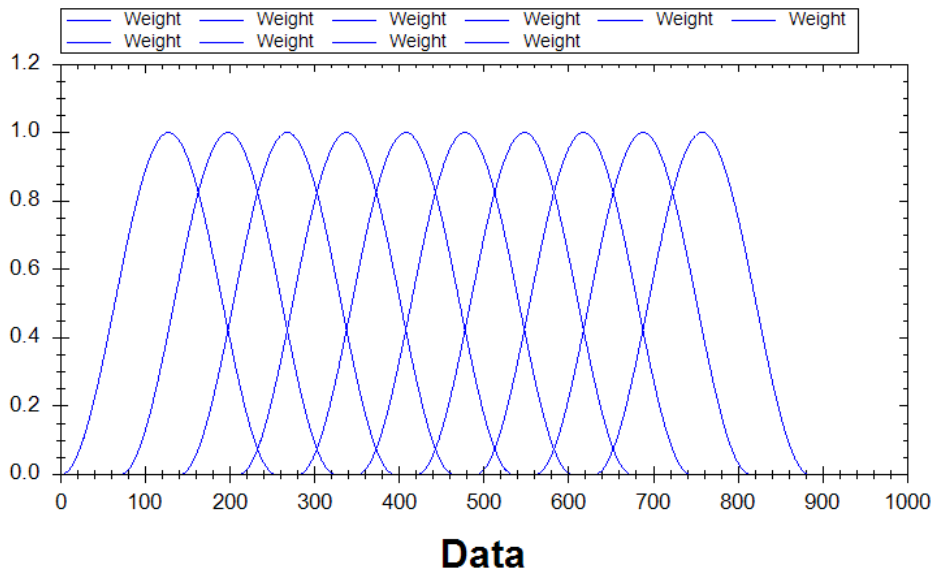
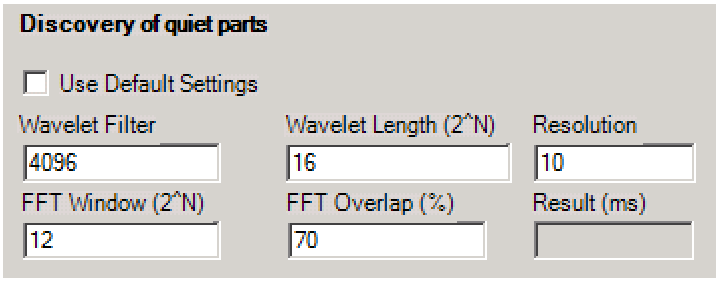


Figure 34: Ten overlapping Hann windows, width of 256, overlap of 70%

The only points which are now being discriminated are those at the very beginning and ending of the entire record. This is an acceptable result because we expect the points of interest to be after the first and before the last second.

3.1.5 Parameters

The whole process of the discovery of quiet parts is adaptable by certain parameters. They have been mentioned in the previous chapters, but will be summed up and explained in detail here. There are five parameters which are available to the user for customization [Fig: 35] . By default, the parameters which delivered the best results in the testing phase will be used:


Discovery of quiet parts

☐ Use Default Settings

Wavelet Filter: 4096
Wavelet Length (2^N): 16
Resolution: 10
FFT Window (2^N): 12
FFT Overlap (%): 70
Result (ms):

Figure 35: Parameters available to influence the transition

Wavelet Filter is the threshold for high frequency cut-offs in the Wavelet filter. It should be a multiple of 2. And it has to be lower than the total length of the array, which is defined by *Wavelet Length*. This parameter defines how wide the wavelet window will be, however the transition detected by the peak energy algorithm will be positioned at the center. If *Wavelet Length* is n, the window size will be 2^n. A reasonable value is between 12 and 16. The parameter *resolution* describes how many points in the noise suppressed result will be averaged. The unit of measurement is in ms. The value 10 means that the step size for the descent will be 10 ms. This parameter will limit the precision of the final result. *FFT Window* defines the size of the sample for each Fourier transformation. It is calculated in the same way as *Wavelet Length*. How far these samples will overlap is defined by *FFT Overlap*, the value is in percent and should therefore be between 1 and 99. None of the input values are being tested internally in the program, using wrong values will cause the program to malfunction. To facilitate the use of these parameters, a list with reasonable ranges of the values is given below:

- Wavelet Filter: 512 - 16384 and smaller than Wavelet Length
- Wavelet Length: 12-16
- Resolution: 5-50
- FFT Window: 12-16
- FFT Overlap: 50-90

3.2 Minimization by Parameter Adaption

The main goal of the project is the multidimensional minimization, but in order to execute such a task some preconditions have to be met. A minimization, or optimization, requires a function which can be evaluated. A function in this project has as input the parameters of the edge detection function and returns some measurement of the noise in the resulting sound. As described in the analysis chapter, only a part of the record will be evaluated. Which one depends on the result of the silence sound transition detection and which algorithm is used for the noise level measurement. For example the Signal Noise Ratio evaluator requires two samples; one has to the signal, the other not. The principal mode of operation for the minimization can be visualized as follows [Fig: 36] :

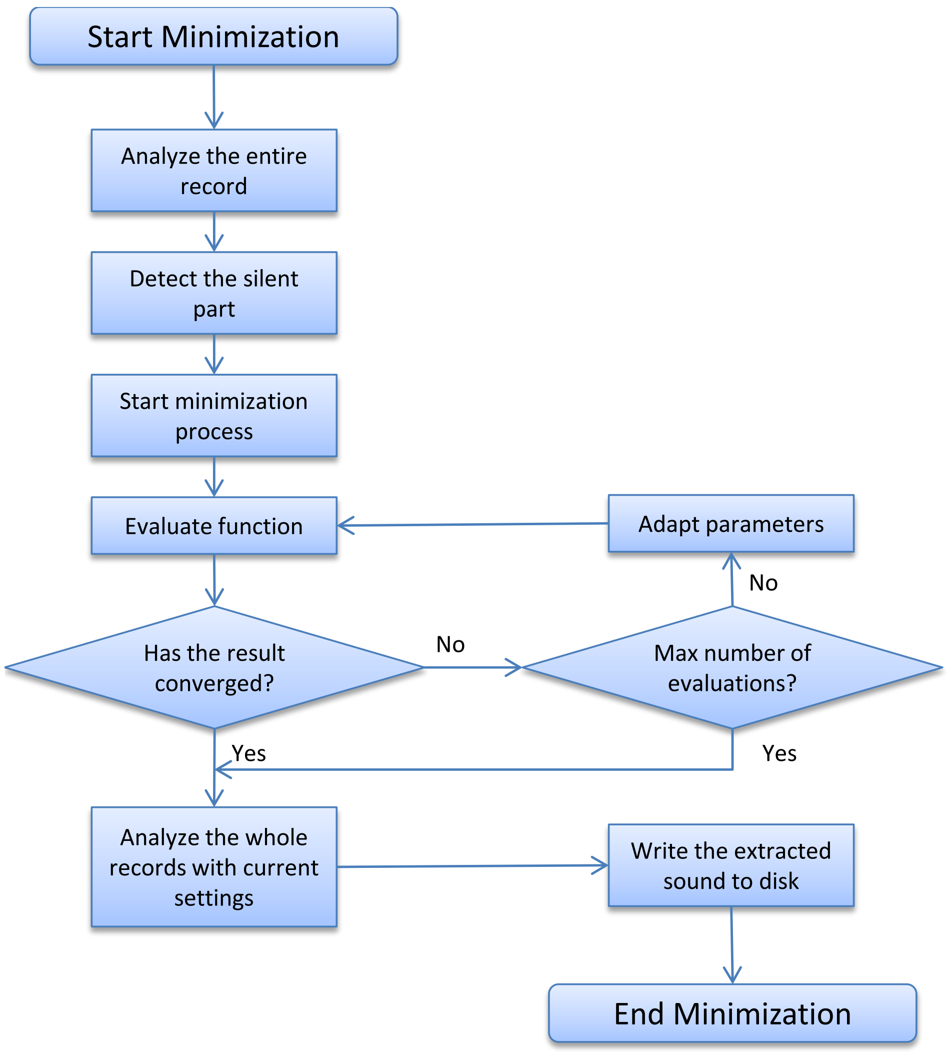
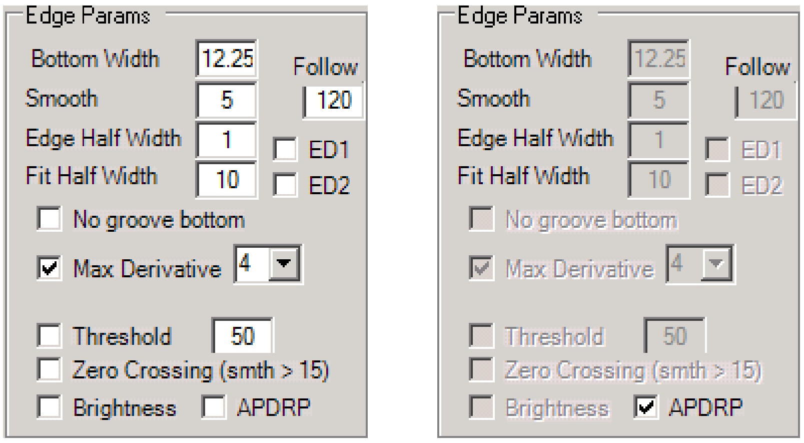

Start Minimization
Analyze the entire record
Detect the silent part
Start minimization process
Evaluate function
Adapt parameters
No
Has the result converged?
No
Max number of evaluations?
Yes
Yes
Analyze the whole records with current settings
Write the extracted sound to disk
End Minimization

Figure 36: Visualization of the basic steps in the minimization

The flowchart shown above contains the basic steps of the algorithm. At first the entire record has to be analyzed with the standard settings and algorithm. Afterwards the length of the silent part has to be measured, the existence of such a part is assumed. If no such part is at the beginning, the minimization will fail or produce strange results. It will then continue with the task of locating a minimum. The function will be evaluated up to a certain number of times; this depends on the restrictions imposed by the user as well as by the applied minimization algorithm. Once the optimization has extracted the optimal settings, they will be stored and used for sound extraction over the whole record. The process ends with the creation of the sound file. The optimal settings will automatically be exported to the Graphical User Interface. During the minimization process the part of the

GUI responsible for the edge detection parameters will be disabled [Fig: 37] .


Edge Params
Bottom Width 12.25
Smooth 5
Edge Half Width 1
Fit Half Width 10
Follow 120
☐ ED1
☐ ED2
☐ No groove bottom
☑ Max Derivative 4
☐ Threshold 50
☐ Zero Crossing (smth > 15)
☐ Brightness
☐ APDRP

Edge Params
Bottom Width 12.25
Smooth 5
Edge Half Width 1
Fit Half Width 10
Follow 120
☐ ED1
☐ ED2
☐ No groove bottom
☑ Max Derivative 4
☐ Threshold 50
☐ Zero Crossing (smth > 15)
☐ Brightness
☑ APDRP

Figure 37: Left: Minimization disabled. Right: Minimization activated

This is necessary due to design flaws in the RENE application; some attributes are internally stored as global variables and can be changed anytime by the user.

3.2.1 Evaluating the Noise Level

Once the function has been evaluated, the result will be reduced to a number. This process is crucial and if the evaluating function was not to function properly, the whole result will be distorted. The function should be able to minimize noise levels without reducing the amplitude of the sound in the process. Over the duration of the project, multiple ways of evaluating the noise levels inside the result have been tested:

- RMS of the bottom width
- Standard deviation of the bottom width
- RMS of a wave file data
- Standard deviation of a wave file data
- Energy in the wave file data
- Energy in the frequency domain in the audible range
- SNR between silence and sound

All of these methods have been regrouped in a separated class in the project called Evaluator. The idea behind the fluctuation in the bottom width as a quality criterion is that if the values vary too frequently, this will indicates noise. It has been observed that in low quality records, the distribution of the bottom width is wider than usual. Another idea was to analyze the fluctuations in the sound, the lower they are in the silent part, the lower is the activity. The only activity which is inside the quiet part is noise. Therefore minimizing this activity should also minimize noise. This is frequently the case, but this quality measure does not take into account detrimental effects on the part which contains sound. The same problem persists if only the activity in the frequency domain inside the silent part is measured, one might accidentally minimize the content too. The only way around this quandary is taking into account the effects on the sound. This is done using a Signal Noise Ratio between the two parts. The Signal Noise Ratio is normally measured by dividing the energy of the signal by the energy of the noise.

$$SNR = \frac{P_{signal}}{P_{noise}}$$

However the result of the equation represents better quality if the SNR becomes higher, which is incompatible with the minimization algorithm. Therefore the formula has to be inverted. This means the result will always be below zero because $P_{signal} > P_{noise}$. To facilitate the readability of the result is multiplied with a constant C_r, where C_r is set to 100 for this project.

$$SNR = \frac{P_{noise}}{P_{signal}} * C_r$$

Now the result of the SNR is always between 100 and 0, where 0 represents the best possible outcome.

3.2.2 Selecting Parameters for the Adaption Process

The parameters for the input functions are at the same time the parameters for the edge detection algorithms. The more parameters a function has, the more complicated becomes the detection of a minimum. If some parameters can be ruled out in a preliminary study, the calculation time will decrease significantly. Moreover for some parameters only a specific range of values makes sense. In this sub chapter we will introduce these constraints for the two main algorithms, Max Derivative and Generic Kernel Edge detection. The first algorithm to be analyzed is the currently used Max Derivative algorithm, he has two parameters which might be susceptible to this kind of optimization: *Fit Half Width* and *Smooth*.The other parameters have either no influence on the result(*Edge Half Width*), are not prone to optimization due to a binary operating principle (*Follow*) or will be determined otherwise (*Bottom Width*). The following values were retrieved using the SNR algorithm [Fig: 38] .

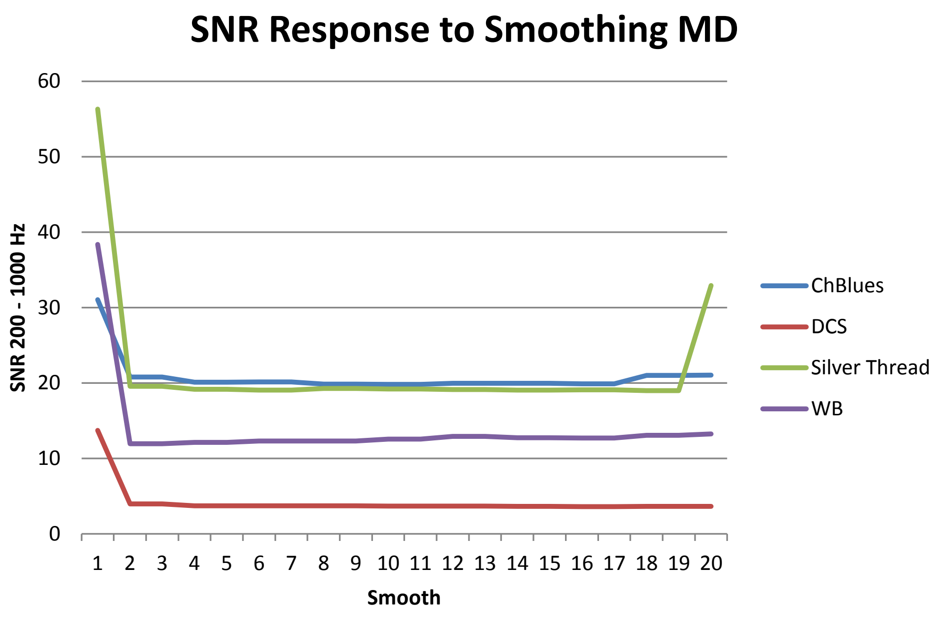


Figure 38: Four records and their behavior concerning smoothing

The graph shows how the different records react to changes in the smoothing [Fig: 38] . There seems to be a valley in the middle, but no minimum which promises an enormous gain compared to other values. It is important however to choose a value inside this valley, a value below 2 or above 19 has a detrimental effect on quality. There is no connection between the quality of a record and its response to smoothing.

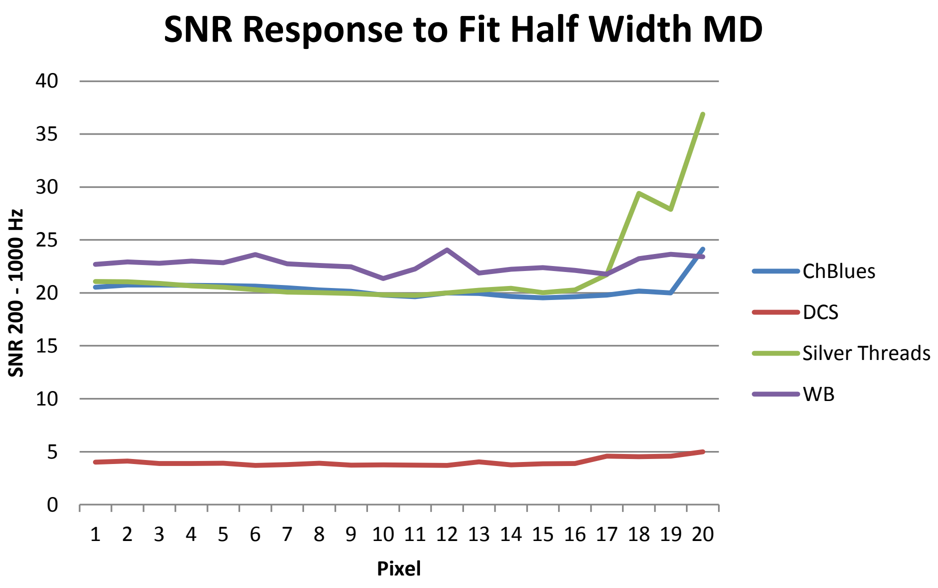


Figure 39: Four records and their behavior concerning Fit Half Width

The result for the *Fit Half Width* parameters is more ambiguous [Fig: 39] . While there is no obvious minimum for all of the records, some respond extremely to high values, whereas others have a relatively flat graph. The DCS record, which has the best quality, proves to be the most immune sample in regard to this parameter. The findings presented in the last two graphs [Fig: 38] [Fig: 39] were the main reason for implementing an edge detection algorithm which has more parameters. The Generic Edge Detection algorithm has a kernel of variable size. The five parameters have been explained in the previous chapter, but will be briefly named again for convenience:

1. Kernel Width
2. Kernel Height
3. Horizontal Multiplication Factor
4. Vertical Multiplication Factor
5. Fit Half Width

Whereas the last parameter is the same as in the previously tested edge detector. All of the above mentioned parameters exert influence on the result. As for the previous algorithm, the parameter *Bottom Width* is determined automatically. The first parameter to be tested is the width of the kernel:

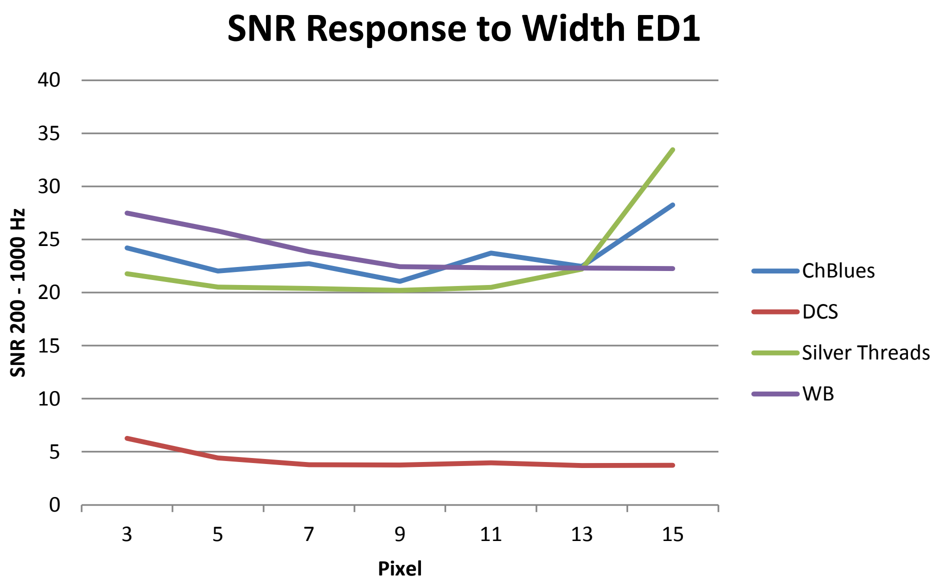


Figure 40: Four records and their behavior concerning the kernel width

The results are once again ambiguous, for some records there seems to be a valley [Fig: 40] , for example ChBlues or Silver Threads. For others the response is flat and the quality increases slightly with the size. No general rule can be deduced, but it looks as a width of 9 pixels delivers the best results for most of the records. Another important parameter is the height of the kernel; the manipulation of it introduces at the same time a low pass filter and should therefore be modified with caution:

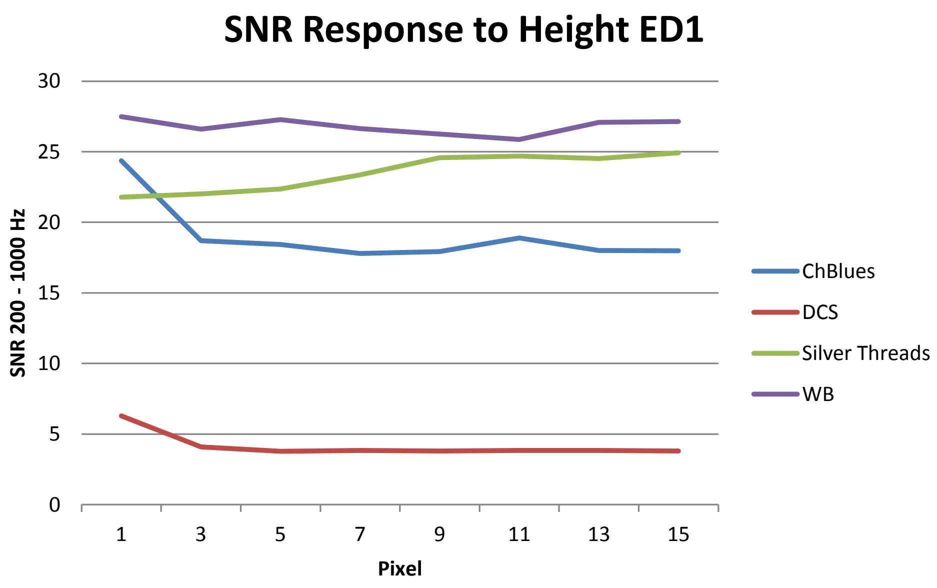


Figure 41: Four records and their behavior concerning the kernel width

The graph [Fig: 41] shows how most of the records react gently to the adaption, only low values may sometimes introduce a decrease in quality. In general the influence on the final result is once again small, as long as the kernel height is above one. However while some records display a slight increase in quality with an increase in height, such as DCS and ChBlues. For others the opposite is true. Now we have seen how the size of the kernel influences the result, but how does the weighting of the different points inside the kernel modify the sound quality?

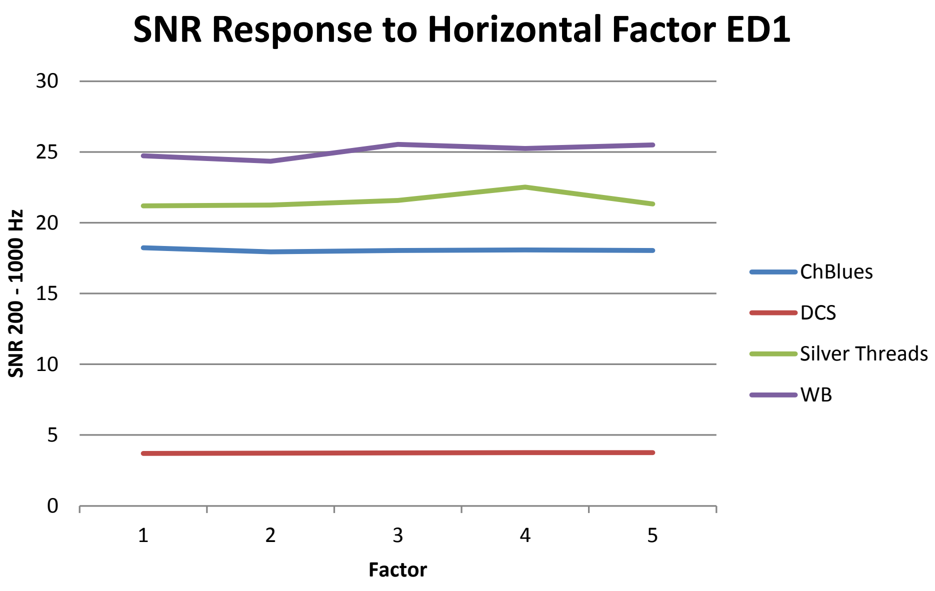


Figure 42: Four records and their behavior concerning the horizontal factor

Almost all of the lines are flat [Fig: 42] . The adaption of this parameter has no influence as long as the quality of the record is medium or better. Only WB and Silver Threads, which are the record containing the most noise, are showing minor changes.

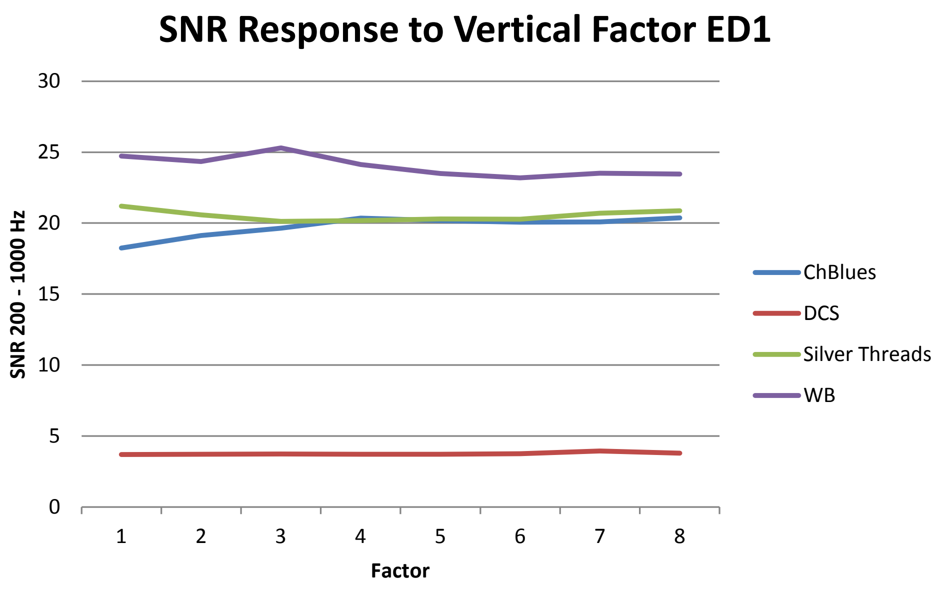


Figure 43: Four records and their behavior concerning the vertical factor

The *Vertical Multiplication Factor* delivers a similar performance [Fig: 43] . But this time medium quality records are affected also, such as ChBlues. Only DCS resists any manipulation by this parameter. However, for most of the result is no global minimum available. The impact on the final result is once again minimal.

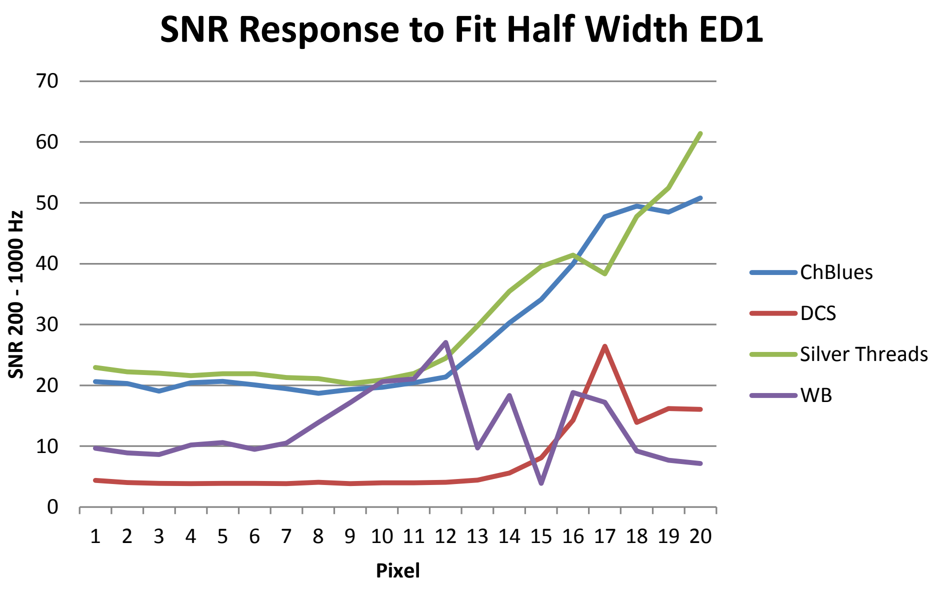


Figure 44: Four records and their behavior concerning the vertical factor

Fit Half Width delivers a more vivid image, all of the records are severely affected by this parameter [Fig: 44] . Some show highly irregular behavior, which will be hard to detect for a minimization algorithm. However most of the local minima are close to the previously used values, in a range between 3 and 9. But even in this area the results differ largely for every record.

All the results presented above represent only how the result reacts to manipulation of one of those parameters. However if the function is nonlinear, the results might be entirely different for other combinations of settings as presented above. An example of mutual influence would be the relationship between the width and the height of the kernel. In the previous images [Fig: 40] and [Fig: 41] the relation between the score and the both parameters is weak. If we now compare these results with the following graph [Fig: 45] :

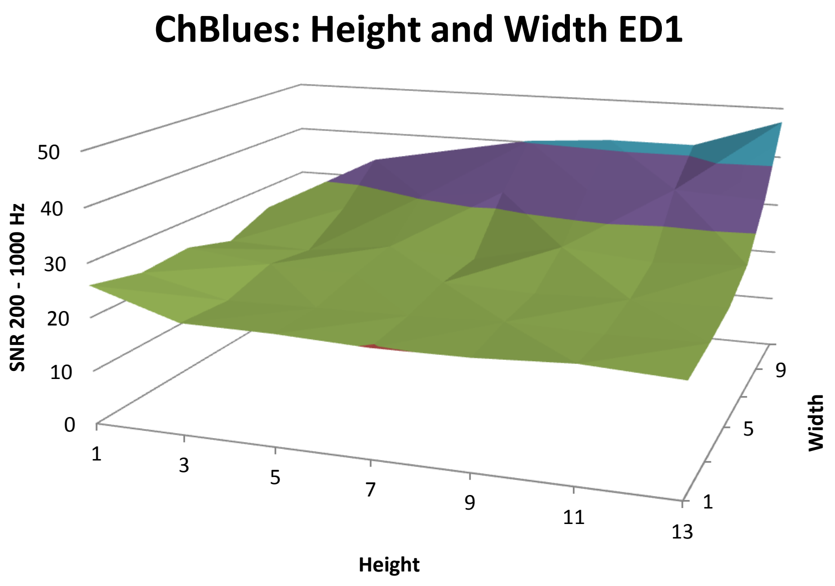


Figure 45: Four records and their behavior concerning the vertical factor

The relationship between the two parameters and the score is not that obvious anymore. For both parameters, the result changes quickly if the other parameter has a high value. For example the noise increases much faster with the height if width has a value above nine.

3.2.3 Nelder-Mead Algorithm

The minimization algorithm does not contain any specifics of the project. It will only receive a function and try to adapt the input parameter in a way which results in a minimum as output. The translation of the input parameter to the score has to be made inside the function itself. The function contains three basics steps: translation of the arguments, recalculation of the edge detection and the calculation of the score. To simplify the translation, the Nelder-Mead algorithm has been customized to take into account boundaries for each parameter. Whenever a new value for a parameter is being calculated, it will automatically be tested if it exceeds the boundaries.

3.2.4 Conjugate-Gradient Algorithm

This minimization algorithm includes the same basic steps as the Nelder-Mead algorithm: translation, recalculation and rating. Moreover it has to provide a method which is able to calculate the gradient numerically. The derivatives are calculated with the formula given in the analysis chapter.

Often the result of the derivative is not precise enough, so the algorithm will jump to points far outside of the scope of reasonable values. For this reason the result of the gradients will be multiplied with a factor which decreases the result. For the moment this value has been fixed to 50%.

3.3 Modifying the Edge Detection

As it became obvious that the currently implemented edge detection algorithms would not perform significantly better, a routine with more parameters had to be created. The basic procedure, using the first derivative will not be changed. It is robust and an established principle. The herein presented algorithm has been inspired by the Sobel operator, and a specific set of parameters is used it will behave the same way. But of course, the calculation of the derivative is only a small part of the complete edge detection algorithm. The interesting edges have to be extracted and tested for viability. But this is not sufficient to extract sound which is natural; the edges have to be detected with sub pixel accuracy to achieve a clear and crisp reconstruction of the original audio data.

3.3.1 Bottom Width Calculation

One parameter which was varied during the first phase was the bottom width. It quickly became clear that adapting this parameter randomly is a waste of calculation time. The only goal for it is to be as close as possible to the real value in order to deliver optimal results. The first impulse was to deduce it manually for each image and change it every time a different record is being analyzed. This is unnecessary and time consuming. Scanning the entire image and simply taking the average would simply be too time consuming, this is where random sampling comes into play. Before any edge detection on the original image is run, this parameter will be calculated with a given probability. As long as the bottom width samples are normally distributed, the number of samples which is necessary can be predicted with accuracy. If every line has the same probability of being selected, the probability distribution can be visualized using a histogram of the entire population. This is the case for each member of the sample $X_1, X_2, ..., X_n$.

The largest obstacle is the usage of the bottom width in each of the edge detection algorithms. It is itself an important parameter for the max derivative and the generic kernel edge detection. This means it has to be calculated before the two algorithms are run. The problem is how will the bottom width be measured without using one of the edge detection algorithms? This is a classical chicken or the egg problem. The solution which was chosen, is using the generic kernel edge detector but without any correction. This means any value between zero and the entire groove width is possible. The better the quality of a record, the closer the form fits a normal distribution. A very nice example would be the bottom width distribution [Fig: 46] of the DCS record, which has a moderate quality but very low noise.

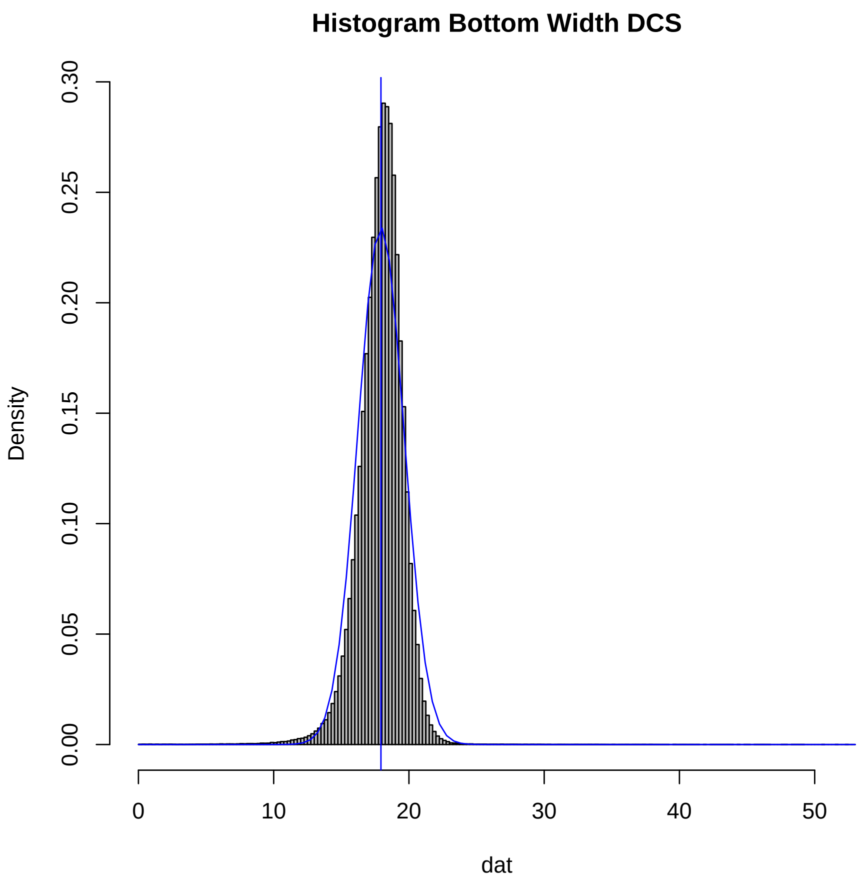


Figure 46: Histogram of the 795,883 bottom widths which were correct out of 802,800

An opposite example would be the low quality and low contrast record called WB. The distribution [Fig: 47] is massively skewed towards lower than average values, but the peak is still close to the absolute mean.

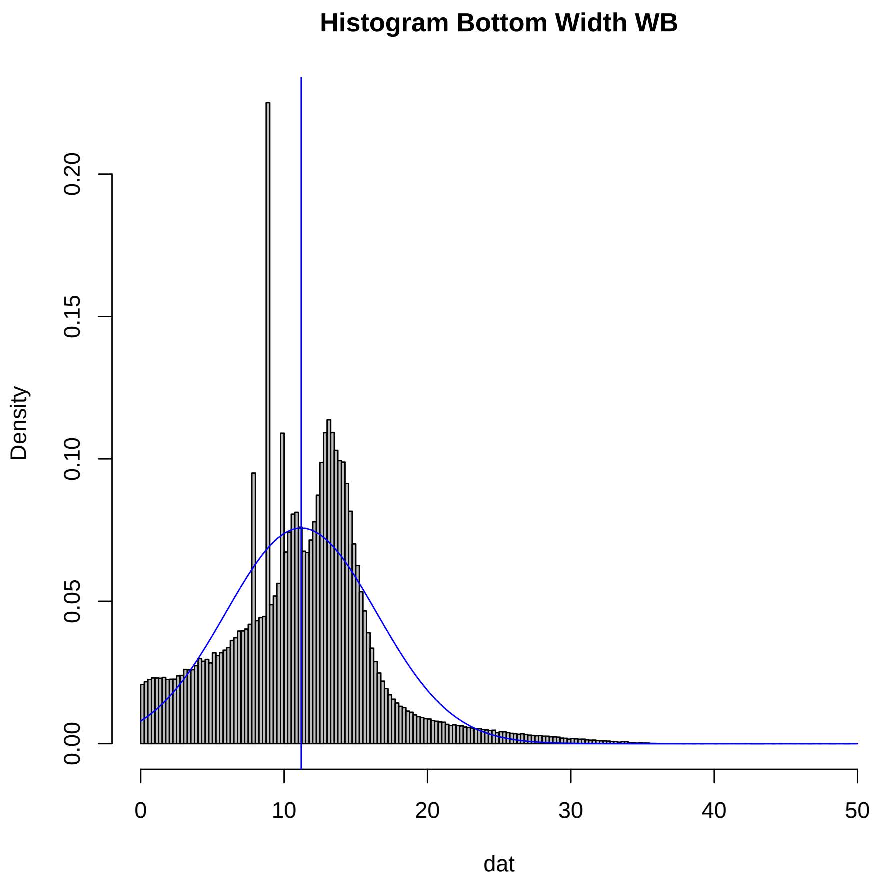


Figure 47: Histogram of the 559,339 bottom widths which were correct out of 634,760

Moreover massively more bottoms were incorrectly detected and thrown away. This could serve as an indicator for the record quality in a further project. The implementation of the random sampling process is a simple loop which doubles the sample size each time until the standard error is low enough. Or in other words the probability for the bottom width to be another value than the current sample average is sufficiently small. This procedure is visualized in the next flowchart [Fig: 48] :

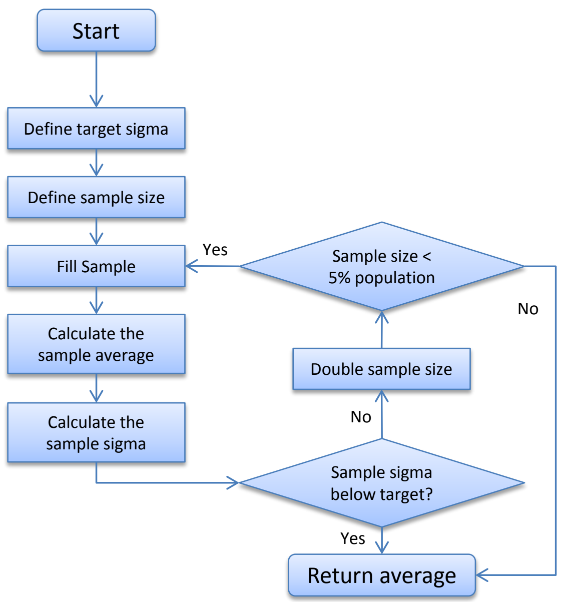

Start
Define target sigma
Define sample size
Fill Sample
Calculate the sample average
Calculate the sample sigma
Yes
Sample size < 5% population
No
Double sample size
No
Sample sigma below target?
Yes
Return average

Figure 48: Visualization of the random sampling process

3.3.2 Generic Kernel Edge Detection

As explained in the analysis chapter, the kernel which is used for this edge detection algorithm may vary in size as well as in the weighting of the different lines. The kernel will be applied for each line which was chosen by the tracking algorithm to be part of the groove. Each line will then be searched for a maximum and minimum value; these two maxima represent the highest slope in the original image and are the border of the groove bottom. An example how the result at this stage looks is the following graph [Fig: 49] :

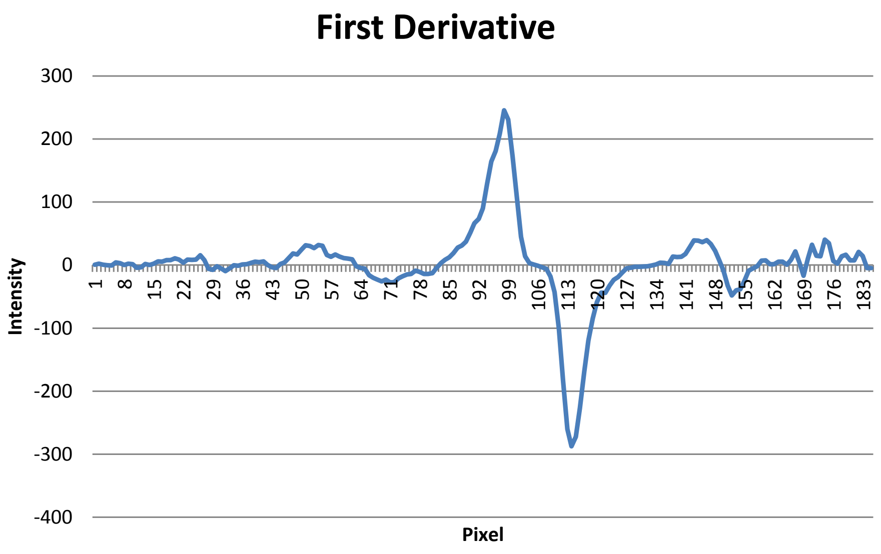


Figure 49: Average of the first derivative of the first 100 lines, ChBlues record

In order to eliminate outliers and give the groove center more weight, a windowing function usually applied to data before a Fourier transformation is multiplied with each line. This has proven helpful in disk with low contrast where the detected point might bounce from one line to the next [Fig: 50] .

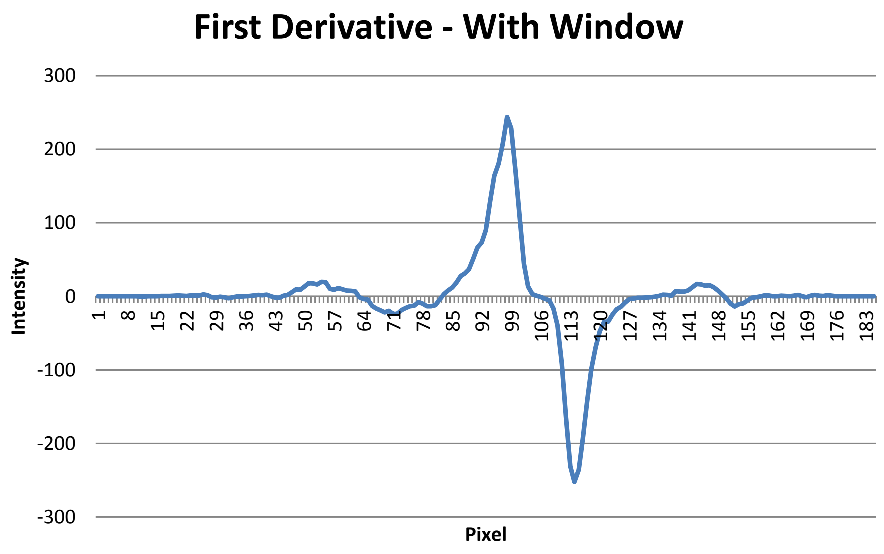


Figure 50: Average of the first derivative of the first 100 lines, windowed, ChBlues record

The data is now ready for the maximum and minimum detection. Once they have been found and the position is reasonable, a curve will be fitted over the top of both maxima to achieve sub pixel accuracy. If the result is unreasonable, this could be the case if the bottom width is negative or larger than the average bottom width multiplied by a certain factor. The algorithm will try to figure out which edges are at a reasonable position based on a likelihood distribution, and if both are highly unlikely it will simply interpolate. The process is described in the following flowchart [Fig: 51] :

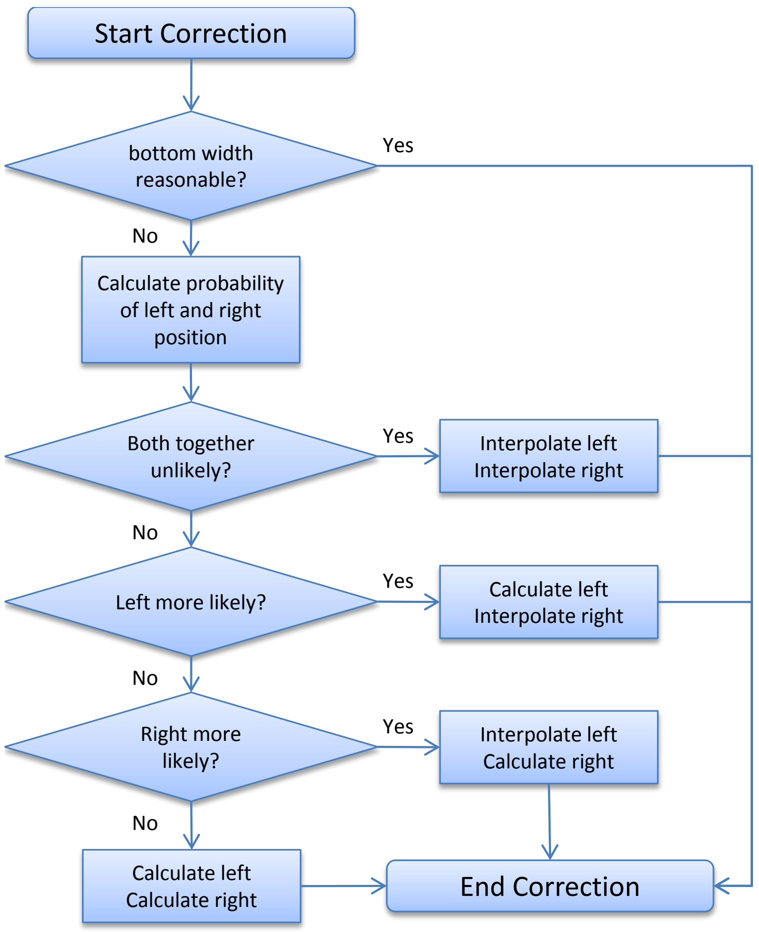

Start Correction
bottom width reasonable?
Yes
No
Calculate probability of left and right position
Both together unlikely?
Yes
Interpolate left Interpolate right
No
Left more likely?
Yes
Calculate left Interpolate right
No
Right more likely?
Yes
Interpolate left Calculate right
No
Calculate left Calculate right
End Correction

Figure 51: Visualization of the error correction process

This process will affect the sound quality because it removes clicks and crackle at an early stage. An edge is considered to be positioned reasonably if it is close to the position of the edge on the previous line. Because most of the time the incremental change in intensity from one line to another is small. The probability is calculated using a Hann window which is centered on the previous edge and has a width of two times the average bottom width. The probability will be indicated by retrieving the value contained in the windowing function at the position which corresponds to the relative horizontal distance.

3.3.3 Edge detection with the second derivative

In an attempt to reduce the noise by extracting more information out of the original image, the second derivative was integrated into a new edge detection algorithm. While the first derivative produces two peaks, a maximum and a minimum, the second produces four peaks. The procedure at the beginning of the algorithm is to apply the Sobel operator two times to the entire image; this produces lines with the following pattern [Fig: 52] :

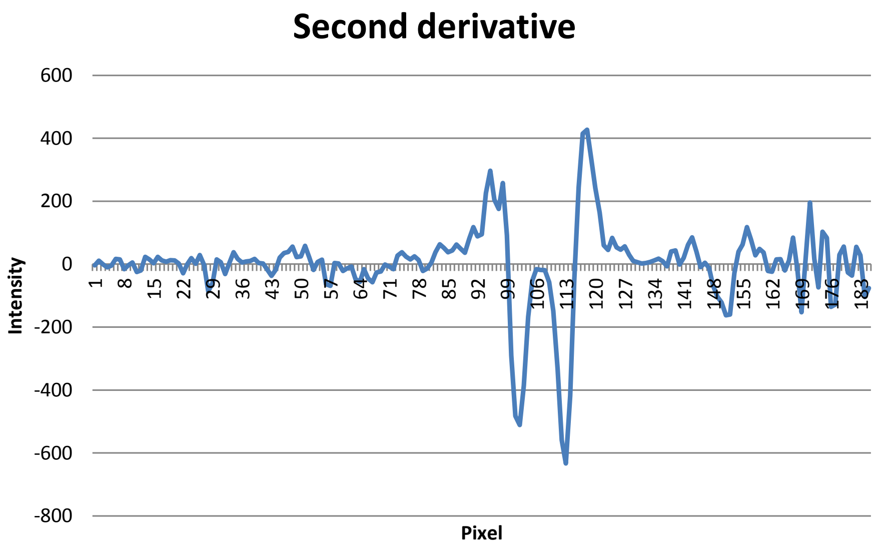


Figure 52: Average of the second derivative of the first 100 lines, ChBlues record

As for the first derivative, often there are points in the image which might interfere with the maximum and minimum detection. Especially in the case of a low contrast record, the groove sides contain significant activity [Fig: 53] .

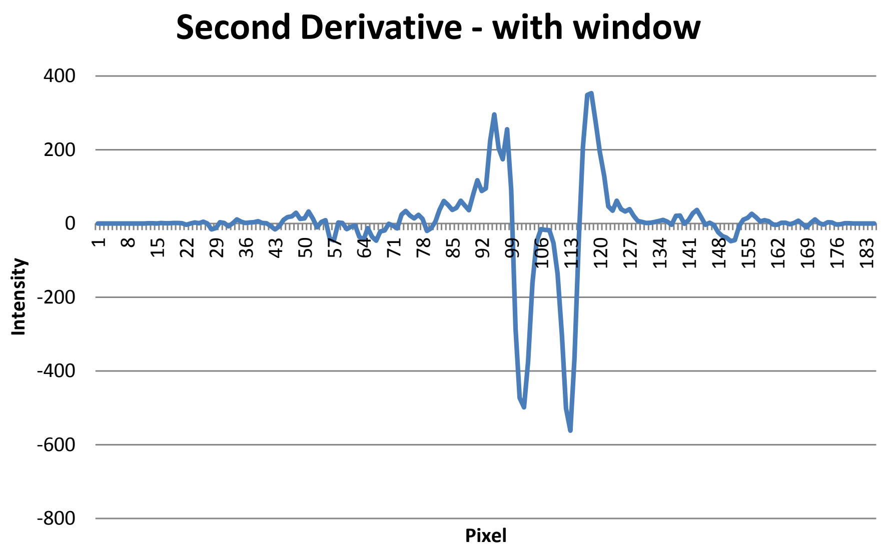


Figure 53: Average of the second derivative of the first 100 lines, windowed, ChBlues record

Occasionally this measure will not be sufficient to detect the four peaks; in this case it can be helpful to use the peaks from the first derivative as a fall back edge. If even the edge detection on the first derivative fails, the only remaining option is to interpolate using the last value. The error correction of this algorithm is based on the same principles as those of the generic kernel, but is more complex due to the possibility of a fall back to the first derivative. Once again the bottom width is a helpful indicator of the correctness of the measurement. The error correction process can be depicted in a flowchart as below [Fig: 54] :

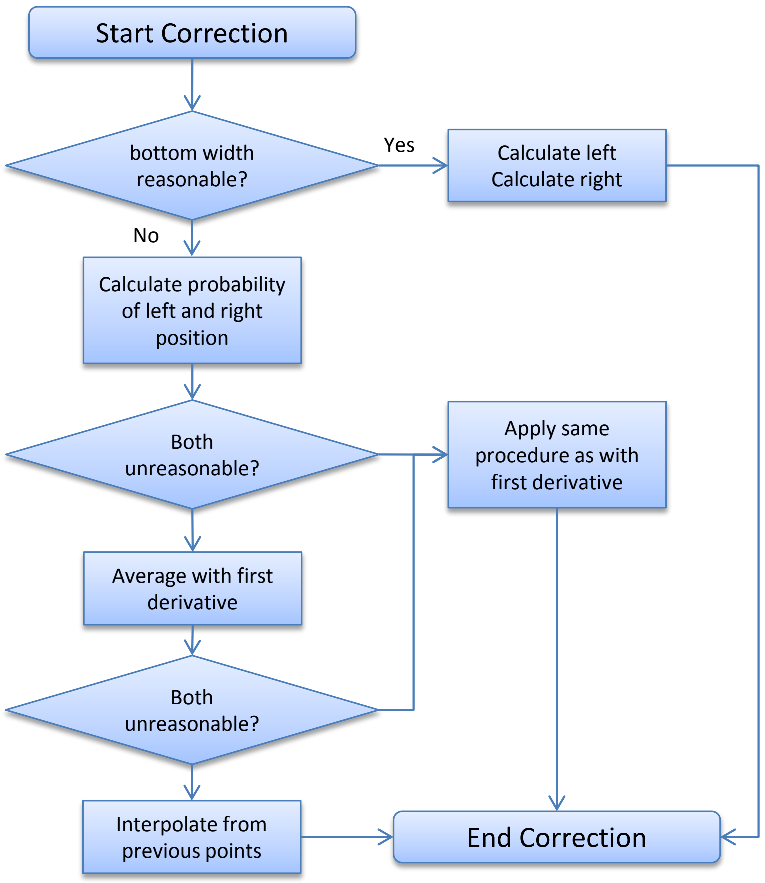

Start Correction
bottom width reasonable?
Yes
Calculate left
Calculate right
No
Calculate probability of left and right position
Both unreasonable?
Apply same procedure as with first derivative
Average with first derivative
Both unreasonable?
Interpolate from previous points
End Correction

Figure 54: Visualization of the error correction process

If the availability of more data is an advantage for the algorithm will be shown in the chapter which contains the tests. The probability of an edge position will be calculated in the same way as for the first derivative.

To sum up, the implementation had to overcome some obstacles but in general the techniques could be implemented as planned.

4 Problems

This chapter will shortly introduce some of the pitfalls which were encountered in the course of this project. If known the cause will be explained and the solution will be outlined.

4.1 Restricting the Edge Detection to a specific Timeframe

When minimizing the noise in the quiet part, it is not necessary to analyze the entire image when revaluating adapted parameters. This means the evaluated points have to be restricted to a certain time limit which corresponds to the transition. An additional difficulty is the measurement of the effect of the parameter adaption in a specific sector of the image. This is necessary when calculating the SNR because a part of the file which contains sound has to be taken into account too. An unresolved problem is the graphical user interface which seems to ignore the Y coordinates when scanning the middle of the image. The problem is visualized as follows [Fig: 55] :

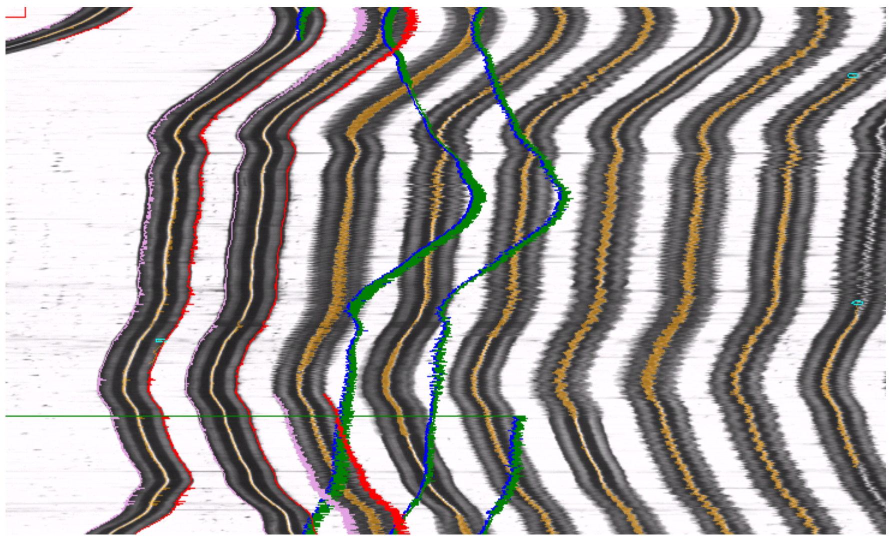


Figure 55: Problem with the visualization of the tracking in the middle of the record

The green and blue lines show where the edges supposedly have been tracked.

4.2 Avoiding the relative Intensities of Wav Files

When comparing two sound streams, one has to take into account relative scale in the wav file. The reconstruction of the source intensity is not possible once a wav file has been created. The only way to avoid this dilemma is to access the derivative of the movement directly. This approach is being taken when the rating algorithms are judging the result of a minimization.

In general there were no serious problems which could not be handled.

5 Tests

This chapter contains the tests which were made to compare the results of this project to the expectations as well as to previous results.

5.1 Discovery of quiet parts

Using the techniques described in the previous chapter, test files will be fed to the program to test its response to the various files and different types of records. To shorten the process, a method of processing a wave file directly has been introduced into the program.

5.1.1 The test samples

In order to ensure proper functioning with different kinds of records, a test sample of seven records has been taken. All of them contain a lead-in groove which means they have at least a second of silence in the beginning. They will be referenced to as V1 to V7. The exact position of the quiet part has been detected with the software Sound Forge Pro. Using a sonogram, the transition has been determined by spotting the patterns which emerge when sound is played with the eye. The following list describes the seven records:

Identifier	Record name	Length(ms)	Transition(ms)
V1	DCS	7714	1071
V2	Loc	7741	1285
V3	MC101	8349	1675
V4	MC101-3	7808	2212
V5	Video Shoot	7918	4008
V6	Quinc3	5014	1324
V7	SundarDiscs	5113	2540

These files have a varying degree of quality, which means some contain more noise than others. Moreover the transition is not always smooth. Especially the last file *V7* has a transition which is hard to detect. Even using a sonogram and as a human, the transition is not easily spotted. Although the unit of measurements is in milliseconds, it should be made clear that this is slightly misleading because it simulates an accuracy which is simply not present. More often than not, the beginning of the sound is ambiguous.

5.1.2 Results for the test sample

Using these seven sample files as input, the results are very close to those obtained by the eye. All but one case are detected with an error below 100ms. As described in the previous chapter, the examples with a smooth transition produce a bigger error. The following table contains the result for each record:

Identifier	Transition(ms)	Detected Transition(ms)	Error(ms)
V1	1071	1086	15
V2	1285	1300	15
V3	1675	1666	9
V4	2212	2216	4
V5	4008	4012	4
V6	1324	1421	97
V7	2540	2684	144

The average error is 41ms, which 50% below the results any other algorithm yielded when tested against the same samples. These results were obtained using the standard settings, how the standard settings were found is described in the next sub chapter.

5.1.3 Optimal Parameters for FFT

In contrary to the minimization goal for the edge detection algorithms, where we look for optimized settings for each record, in this case we are looking for the optimal settings for a set of records. There are five different parameters for the transition detection algorithm, trying to minimize them all would be too big a task. So the optimization was limited to two key values, the size of FFT array and the overlap percentage. The range of values was constrained to the following ranges:

- FFT Window: 12 - 16
- FFT Overlap: 10 - 90

The error function δ_s is the sum of all deviations $\left|t_{e1} - t_{a1}\right|$ between the two values for the transition:

$$\delta_s = \sum_{i=1}^{7} \left|t_{ei} - t_{ai}\right|$$

All the possible combinations have simply been calculated which resulted in the following table:

	12	13	14	15	16
0	1713	1649	1738	1430	3204
10	1691	1745	2504	2096	3287
20	1129	1207	1215	1172	3193
30	758	816	842	1151	2383
40	1368	1395	1204	1117	2433
50	613	566	887	1338	2022
60	1340	1396	1484	1411	1841
70	575	685	727	1113	2062
80	1130	1114	1198	1503	1114
90	725	833	783	1095	1717

It is important to mention that these error values represent the values before the Wavelet denoising process. It is assumed for simplicity that an optimal preliminary result will result in a improved overall result. The best and the worst result have been highlighted. They are also the extremes of a general trend, the smaller the window size, the more precise the result will be. The result for the overlap percentage is not that obvious, it seems as the extremes yield slightly less precise results. It can be concluded that in general an overlap of 70% and a window size of 2^{12} will return the best preliminary results.

5.2 Minimization of Max Derivative

At first we are testing which kind of optimization is possible for the current standard edge detection algorithm Max Derivative version four. As we have seen in the previous chapter, the records which possess a low quality have the most potential for optimization. This is why the tests will be run on the WB record which seems to be susceptible to parameter adaptations. The standard rating algorithm, based on SNR will be used if not indicated otherwise. As explained in the previous chapter, there are only two parameters which influence the result. *Fit Half Width* and *Smooth*, these two will be varied in order to improve the result. All of the following results were taken with the following settings for the cutting:

- Cuts Width: 100
- Cuts Difference: 100
- Cuts: Or
- Cuts Brightness: deactivated

The results are always compared with a file called "Before", it was extracted with the following settings:

- Smooth: 5
- Fit Half Width: 5
- Bottom Width: random sampling result
- Follow: 120

5.2.1 Results with Nelder-Mead

At first, the WB record was minimized with the Nelder-Mead algorithm which does not use any gradients. It requires seed values which indicate the departure coordinates:

- Smooth: 15
- Fit Half Width: 15

They were chosen far away from a point which could be called a reasonable in order to test the capability of the algorithm to descent to a minimum. However there is always the danger that the algorithm will get stuck in a relatively high local minimum. The initial perturbation was set to two for each parameter; this parameter determines how fast the legs of the amoeba will move initially. The test yielded the following result:

- Smooth: 8
- Fit Half Width: 20

If we compare the result graphically in a spectrogram with the default settings, the overlay will look as follows [Fig: 56] :

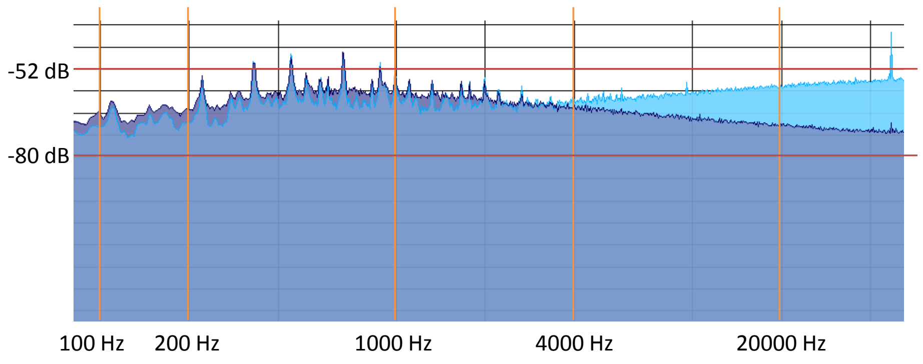


Figure 56: Record: WB, Minimization: Nelder-Mead, Before: Blue, After: Violet

As we can see the noise in the higher frequencies has dropped radically, however in the audible range, intensity has been added. The result will be a decrease in the high frequency domain but an increase in the audible range. However the result looks digital in the time domain [Fig: 57] :

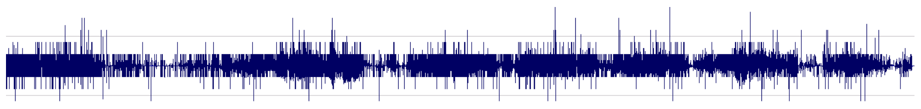


Figure 57: Audio file after the minimization with NM

The audio file confirms the conclusions which were drawn from the inspection of the frequency domain. There is less high frequency noise but the parts which contain sound have lost clarity.

5.2.2 Results with Conjugate-Gradient

The conjugate-gradient Optimizer is based on derivatives which have to be numerically approximated; the question is how much the error introduced by these calculations will influence the result. In order to simplify the task, the seed values were close to the range of reasonable values:

- Smooth: 4
- Fit Half Width: 4

After more than 150 iterations the algorithm came up with the following parameters as result:

- Smooth: 5
- Fit Half Width: 5

Apparently they are close to the seed values, by analyzing the log it becomes obvious that the algorithm has its problems with the discreteness of the function. Numerous values which differ by less than 10^{-3} are tested in a row. Not surprisingly the resulting spectrogram contains two identical frequency distributions [Fig: 58] :

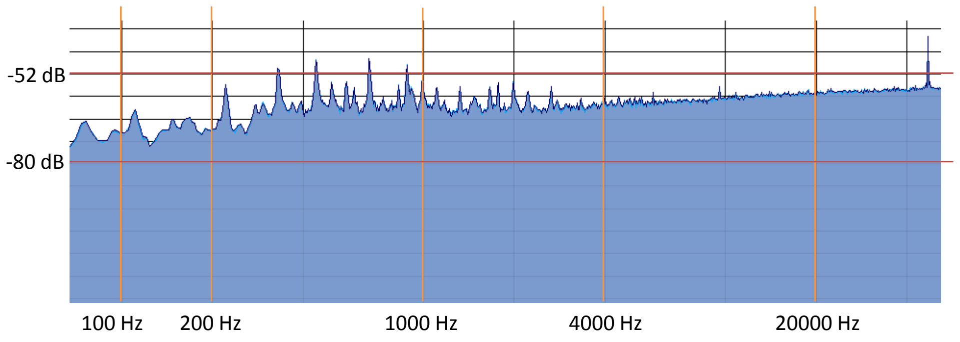


Figure 58: Record: WB, Minimization: Conjugate-Gradient, Before: Blue, After: Violet

The picture might be considered redundant. But it proves that with two different runs with the same settings, the result does not differ. This means that the simple random sampling of the bottom width does not introduce randomness into the final outcome.

5.2.3 Results with Brute Force

The brute force approach cannot really be considered a minimization algorithm because it merely tests all possible outcomes to locate the minimum within these boundaries. For this run, the following bounds were given to the algorithm:

- Smooth: From 2 to 15
- Fit Half Width: From 3 to 15

Due to the relatively few parameters which influence the result the algorithm was done after 20 minutes, it came up with the following result:

- Smooth: 8
- Fit Half Width: 9

The result is close to the values which have been used in previous setups, however the spectrogram does not show significant variations in the frequency domain [Fig: 59] :

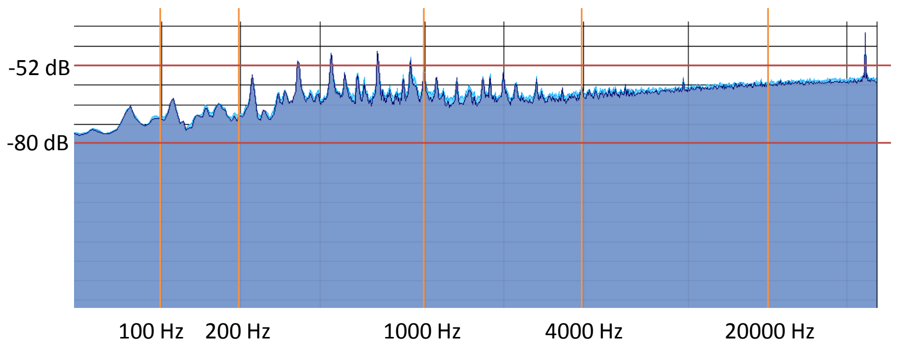


Figure 59: Record: WB, Minimization: Brute Force, Before: Blue, After: Violet

When taking a really close look at the image, a slow decrease in overall intensity can be recognized in the minimized audio file. However when playing the audio file, the difference is not audible.

5.2.4 Conclusion for the Max Derivative Algorithm

Apart from the Nelder-Mead minimization, none of the algorithms was able to create a result which was audibly different from the input. However this difference was bought by an increase in noise in the audible range. This also means that the minimization algorithm is flawed; a result which decreased the quality should be punished heavily. Which could also mean that the Max Derivative algorithm is already close to the optimum and there is almost no room for improvement.

5.3 Minimization of Generic Kernel Edge Detector

After the conclusion was reached that the MD algorithm cannot be optimized any further, the next step was to introduce an edge detector which is more flexible and provides more parameters for the optimization algorithms. However the increase in flexibility leads to a higher dimensional function, in this case a five dimensional space has to be explored. This will cause additional strain for the minimization algorithms. The other parts of RENE remain unchanged:

- Cuts Width: 100
- Cuts Difference: 100
- Cuts: Or
- Cuts Brightness: deactivated

The sound file which is labeled ”Before” was obtained with the following settings:

- Kernel Width: 3
- Kernel Height: 3
- Horizontal Multiplication Factor: 1
- Vertical Multiplication Factor: 1
- Fit Half Width: 5

This kernel is equal to one which the Prewitt edge detector uses.

5.3.1 Results with Nelder-Mead

The first test was once again the Nelder-Mead algorithm, which will execute a constrained minimization in the solution space. The seed point is considered reasonable and is at the following coordinates:

- Kernel Width: 5
- Kernel Height: 5
- Horizontal Multiplication Factor: 1
- Vertical Multiplication Factor: 2
- Fit Half Width: 2

After a search with roughly 100 function evaluations, the following point was returned as minimum:

- Kernel Width: 11
- Kernel Height: 9
- Horizontal Multiplication Factor: 4
- Vertical Multiplication Factor: 1
- Fit Half Width: 4

This means a larger kernel size will improve the result for this record, the specifics can be viewed in the following spectrogram [Fig: 60] :

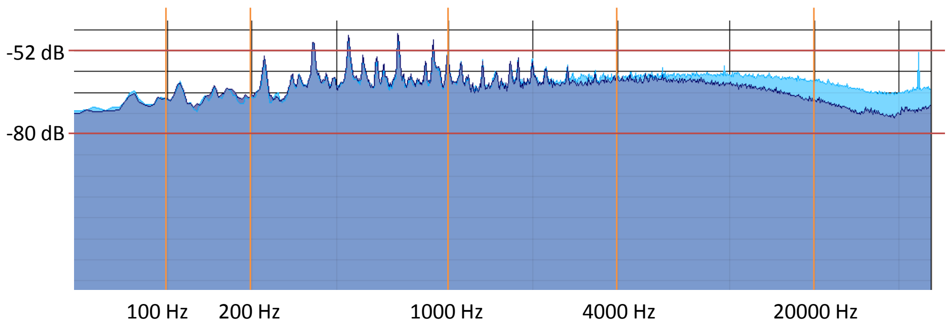


Figure 60: Record: WB, Minimization: Nelder-Mead, Before: Blue, After: Violet

The intensity drop off for frequencies higher than 10,000 Hz is obvious and does not highlight any new patterns. The difference in the audible range is meager. However the peak at the 45,000 Hz frequency is gone.

5.3.2 Results with Conjugate-Gradient

The last time the conjugate-gradient algorithm failed to deliver any significant results, nevertheless the seed point has been kept close to the settings which define the measurements which were taken for the "Before" file:

- Kernel Width: 3
- Kernel Height: 3
- Horizontal Multiplication Factor: 2
- Vertical Multiplication Factor: 2
- Fit Half Width: 10

Once the multidimensional search was done the following result was returned:

- Kernel Width: 21
- Kernel Height: 21
- Horizontal Multiplication Factor: 2
- Vertical Multiplication Factor: 2
- Fit Half Width: 10

This result is highly surprising, because such a large kernel will cut the range which is evaluated at each side of the groove by half of the kernel size. If the groove bottom strays to close to the borders, it will be off the grid. Moreover these values mean that a large number of lines have influence on the current result and it corresponds to a low pass filter which will reach far into the low frequencies. The result however does not exhibit the characteristics of a low pass filter at all [Fig: 61] :

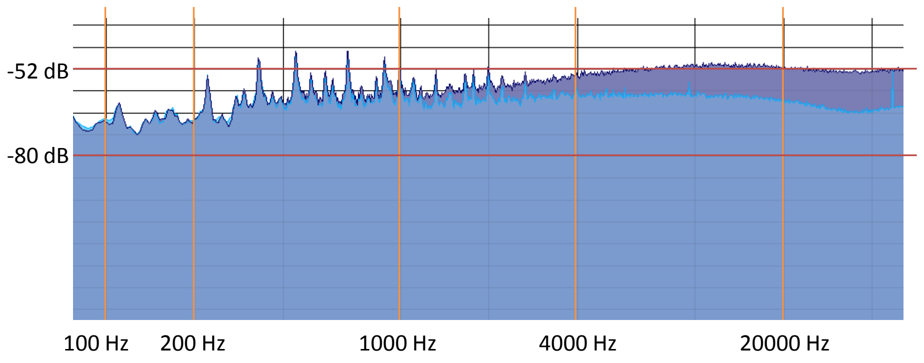


Figure 61: Record: WB, Minimization: Conjugate-Gradient, Before: Blue, After: Violet

In the range which is analyzed by the SNR rating algorithm, the results show almost no difference. But shortly afterwards the noise seems to increase rapidly. The minimized sound file has considerably lower quality than the original.

5.3.3 Results with Brute Force

This time the brute force approach took significantly longer, because the complexity grows exponential the calculation time explodes. This requires a narrower search space which includes the most likely points but will not be able to yield large surprises. The parameters were limited to the following ranges:

- Kernel Width: from 3 to 11
- Kernel Height: from 1 to 11
- Horizontal Multiplication Factor: from 1 to 4
- Vertical Multiplication Factor: from 1 to 4
- Fit Half Width: from 5 to 18

The exploration of this space took more than seven hours and was run overnight. It returned the following parameters for a minimum:

- Kernel Width: 3
- Kernel Height: 3
- Horizontal Multiplication Factor: 1
- Vertical Multiplication Factor: 3
- Fit Half Width: 15

It is remarkable that the resulting values are almost the same as the default values, only the choice of the *Fit Half Width* parameter is questionable. However the output is unexpectedly weak: [Fig: 62]

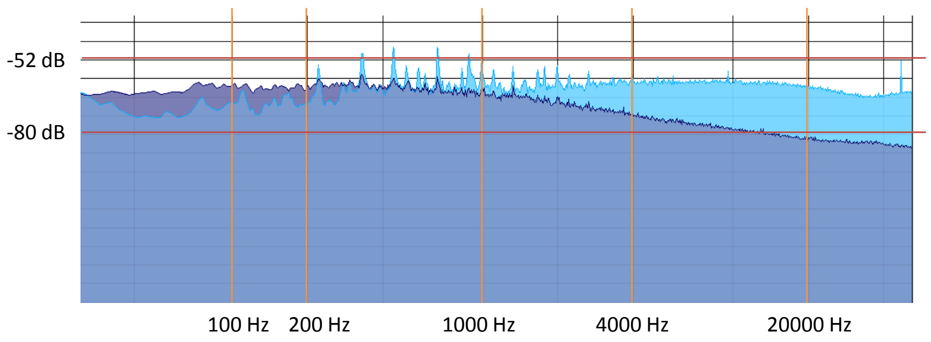


Figure 62: Record: WB, Minimization: Brute Force, Before: Blue, After: Violet

The decrease in high frequency noise has been bought with an increase in low frequency noise. The overall activity has been decreased in the audible spectrum, but the sound is now distorted. The problem in this case could be that the algorithm found a place where the noise decreases faster than the sound, which leads to a lower SNR score but not to an increased overall quality.

5.3.4 Conclusion for Generic Edge Kernel Algorithm

The results for this edge detection algorithm raise more questions than they answer. The minimization shows the weaknesses of the SNR ratio algorithm as well as the difficulties of the minimization algorithms with discrete functions. A future minimization study will have to adapt this part of the project first. The following table gives a short summary of the divergent results and the final score they returned:

Minimization	KH	KW	HF	VF	FHW	Score
NM	11	9	4	1	4	20.5
CG	21	21	2	2	10	30.5
BF	3	3	1	3	15	10.3

5.4 Generic Edge detection

Up until now, the results of the minimization algorithms have been compared, but not the edge detection algorithms themselves. This sub chapter will analyze how the two edge detection algorithms perform relatively to each other for each record. The previous chapter has shown that large kernel values for the ED1 algorithm lead to detrimental results. Therefore a medium sized kernel will be used to take advantage of the information of the surrounding lines. Because of the heavy influence which the *Fit Half Width* exerts, it will be adapted for each record with a simple line search. For MD edge detection, the parameter *Smooth* will be set to 10 constantly. ED1 will use the following parameters:

- Kernel Width: 9
- Kernel Height: 9
- Horizontal Multiplication Factor: 1
- Vertical Multiplication Factor: 1

These parameters will introduce a weak low pass filter for ED1. The first record to be compared is ChBlues [Fig: 63] :

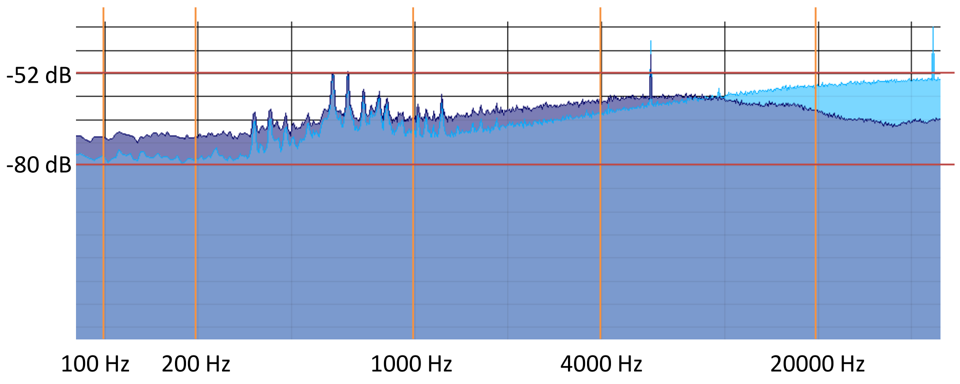


Figure 63: ChBlues: MD is blue, ED1 is violet

This record shows has better quality when analyzed with the old MD algorithm. It contains clear activity in the audible range. A quick test reveals that it sounds cleaner than the result of the ED1. This result seems to be an outlier, quite likely caused by the rating algorithm. The next image shows almost no difference between the two approaches [Fig: 64] :

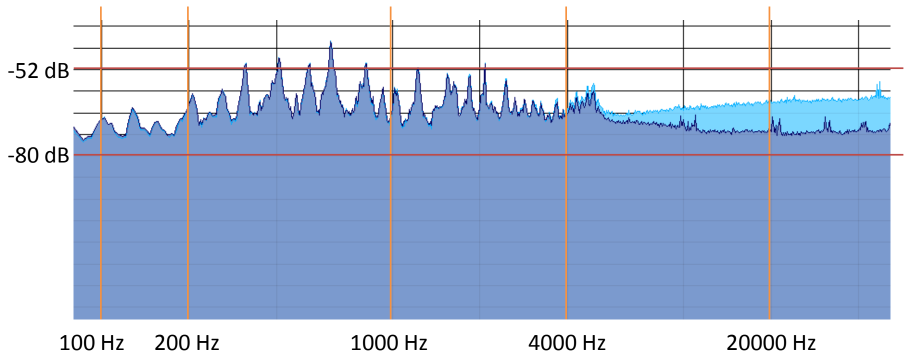


Figure 64: DCS: MD is blue, ED1 is violet

They match almost perfectly in the audible spectrum from 100 Hz up to 4,000 Hz. Afterwards the effects of the low pass filter set in and the noise level drops significantly for the ED1 algorithm. Interestingly it erases the peak at the 45,000 Hz frequency and reveals a new pattern with peaks at the 10,000, 20,000 and possibly 30,000 Hz frequency. The cause of this effect could not be determined so far. The next image shows similar characteristics [Fig: 65] :

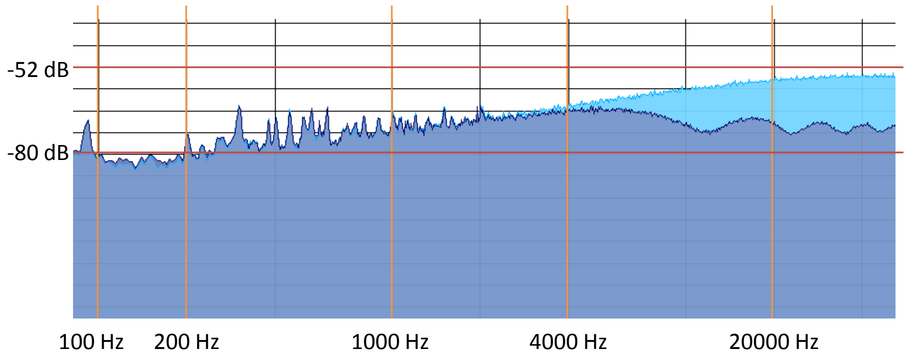


Figure 65: Silver Threads: MD is blue, ED1 is violet

The activity matches until the filter sets in; the pattern this time is more regular and is likely caused by the filter. Interestingly does this record not provoke a peak at the 45,000 Hz frequency for either of the two algorithms. Contrary to the next record which exhibits this pattern again [Fig: 66] :

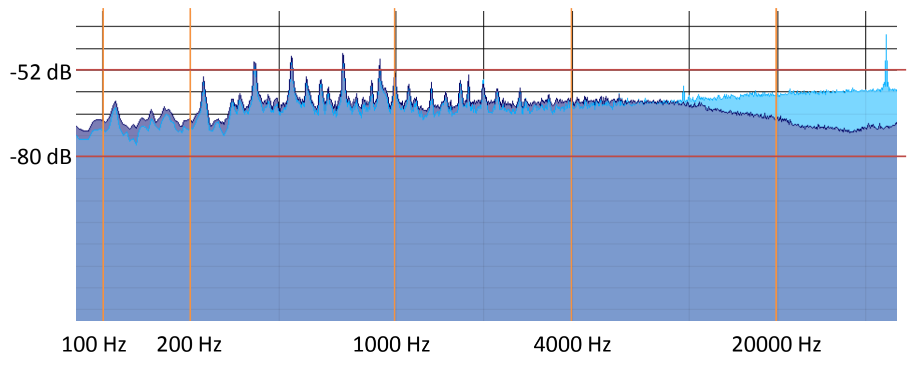


Figure 66: WB: MD is blue, ED1 is violet

This time ED1 edge detector has again slightly more noise in the frequency range between 1,000 and 4,000 Hz. However the difference is not audible when the files are being played. The main difference is again the effect of the low pass filter.

5.5 Second Derivative Edge Detector

This sub chapter will describe how the edge detector based on the second derivative performs compared to the two other edge detection algorithms. Because they do not possess the same parameters, only the *Fit Half Width* parameter will be optimized for each record and algorithm. The rest of the parameters will be fixed. *Smooth* will be set to 10 for MD. The ED2 algorithm does not require any other parameters than the *Fit Half Width* which will be determined automatically. ED1 will be executed with the following settings:

- Kernel Width: 5
- Kernel Height: 5
- Horizontal Multiplication Factor: 1
- Vertical Multiplication Factor: 1

Because we the effects of a large kernel were already shown in the previous chapter, the kernel has been reduced to the above mentioned values. This time the ChBlues record shows no abnormalities for the ED1 algorithm [Fig: 67] .

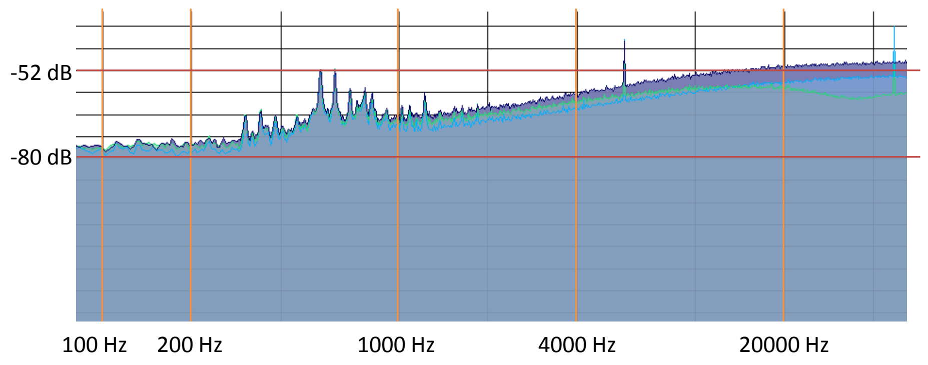


Figure 67: ChBlues: MD is blue, ED1 is green, ED2 is violet

The three algorithms deliver the same results in the audible range, but differ largely in the frequencies above 4,000 Hz. ED2 performs the worst, despite the small low pass filter effect the Sobel operator should have, the high frequency noise increases. The result is even worse for the DCS record [Fig: 68] :

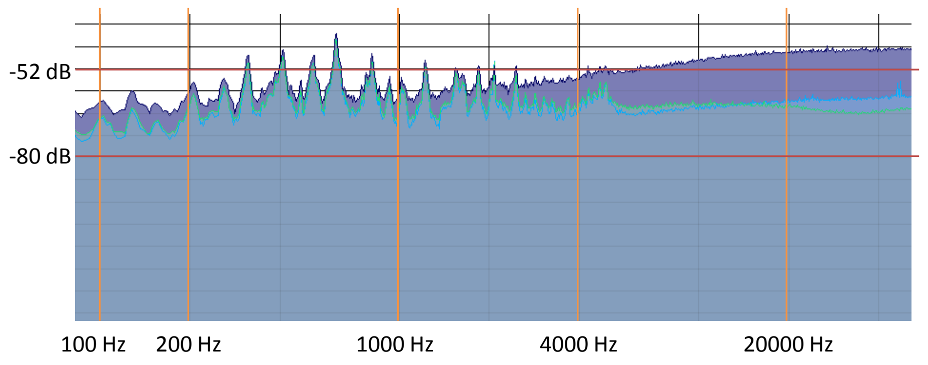


Figure 68: DCS: MD is blue, ED1 is green, ED2 is violet

Now the noise starts at lower frequencies and interferes with frequencies higher than 1,000 Hz. Even more disturbing is the increase in activity below 200 Hz compared to the other two algorithms. Without the large kernel, the results of ED1 and MD are almost similar throughout the entire spectrum. For the Silver Threads record the relative results stay the same [Fig: 69] :

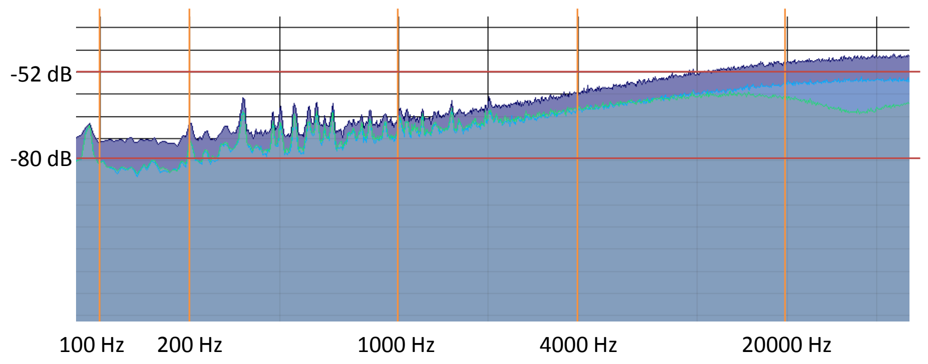


Figure 69: Silver Threads: MD is blue, ED1 is green, ED2 is violet

The noise is now throughout the entire frequency range higher than those of the other two edge detectors. Most of the music or voices will be absorbed by the high white noise. A surprise yields the last record which shows a new loser [Fig: 70] :

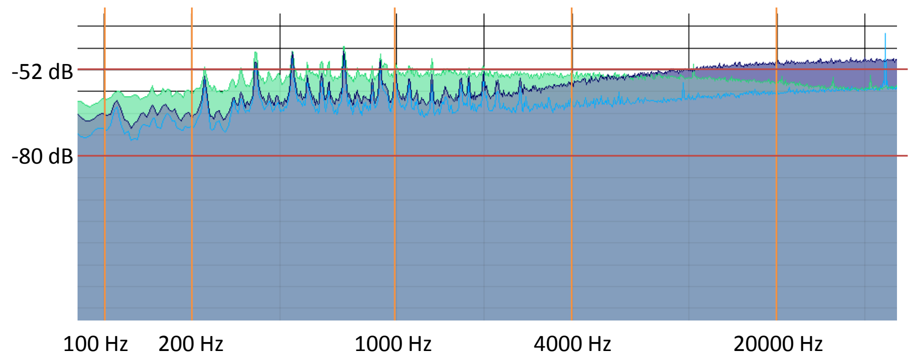


Figure 70: WB: MD is blue, ED1 is green, ED2 is violet

This time ED1 exhibits the same characteristics as ED2 in the previous pictures. This is due to the minimization process which chooses erroneously a high *Fit Half Width* value for ED1. The rest of the characteristics correspond to the findings from the previously tested records.

5.6 Simple Random Sampling

In the Analysis section it has been discussed that every member of the sample has a random distribution, this will be reflected in the sample distribution as well. Basically this means the probability distribution of the entire sample, or in this case of the average of the sample, will again show the same distribution. We have seen in the previous chapter that most of the disks have a Gaussian distribution of the bottom widths. This means the samples should be distributed in the same manner. Moreover the more samples are being taken, the closer should the mean of the samples approximate the real mean of the population. At first, we will determine the probability distribution [Fig: 71] of a sample member from the ChBlues record:

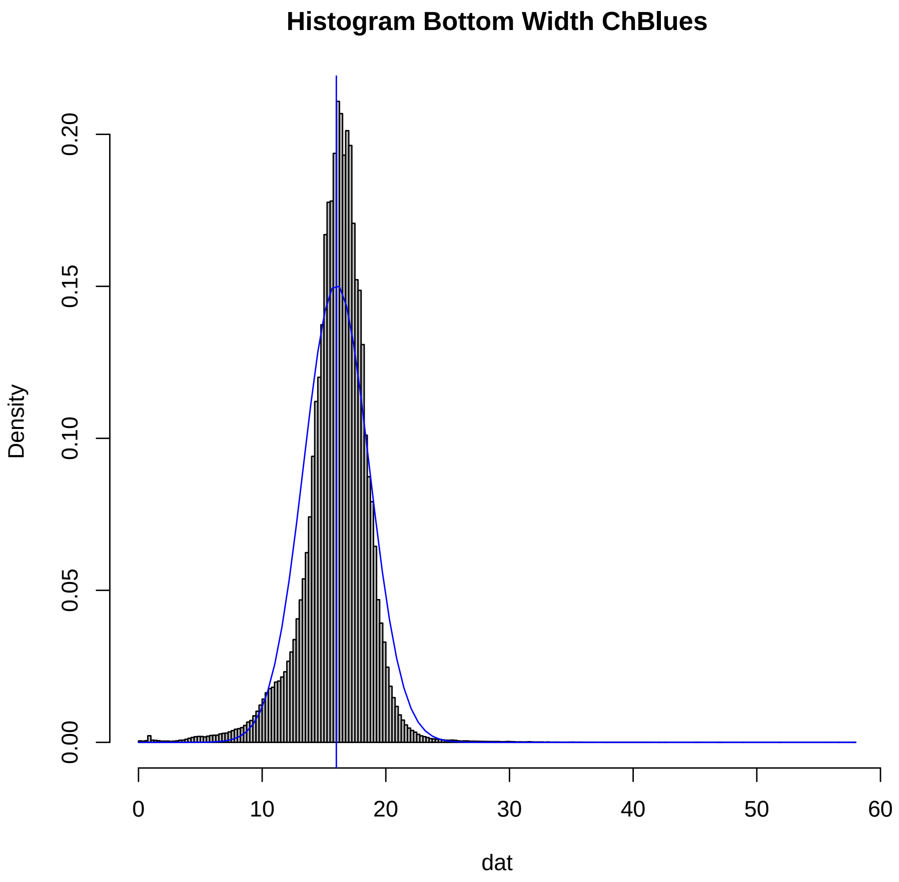


Figure 71: Histogram of all positive bottom widths in the ChBlues record, mean = 15.99777

Now it has to be checked if this probability distribution is reflected in the sample distribution, at first with sample size $n = 500$ [Fig: 72] :

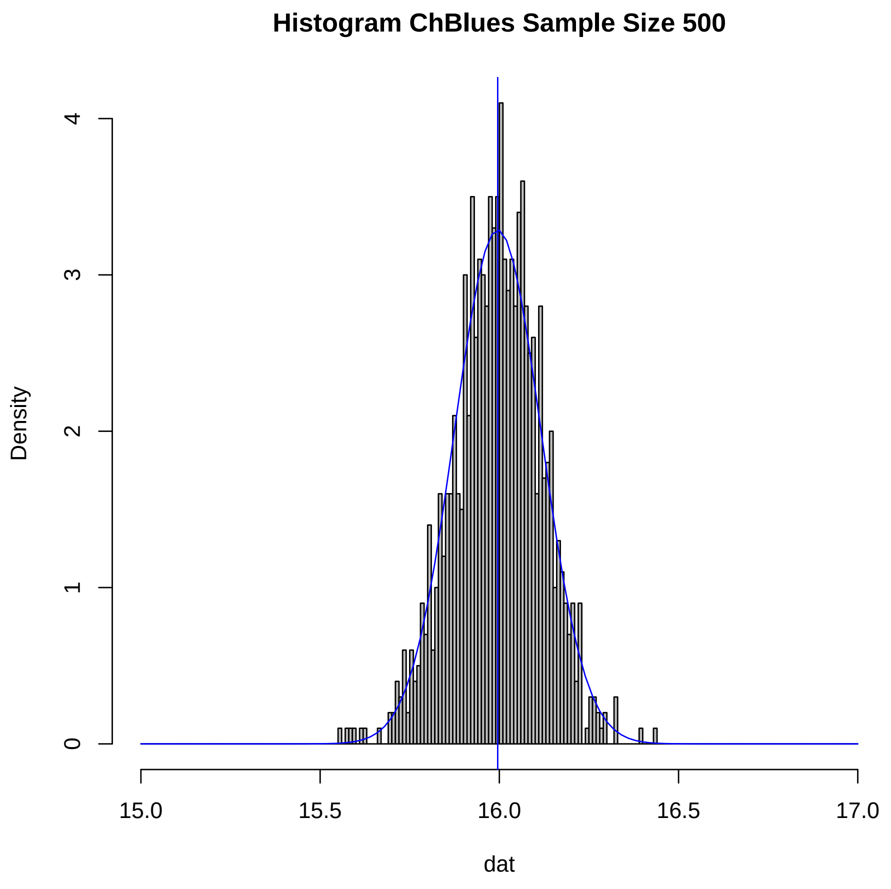


Figure 72: Histogram of the averages from 1000 samples with $n = 500$, mean = 15.99543

This distribution is very close to the original one, although the samples are distributed wider but are also slightly shifted towards the left side. The approximation of the mean is almost exact. This is not surprising considering half a million widths have been measured. The approximation becomes even narrower if the sample size is increased to $n = 2000$ [Fig: 73] .

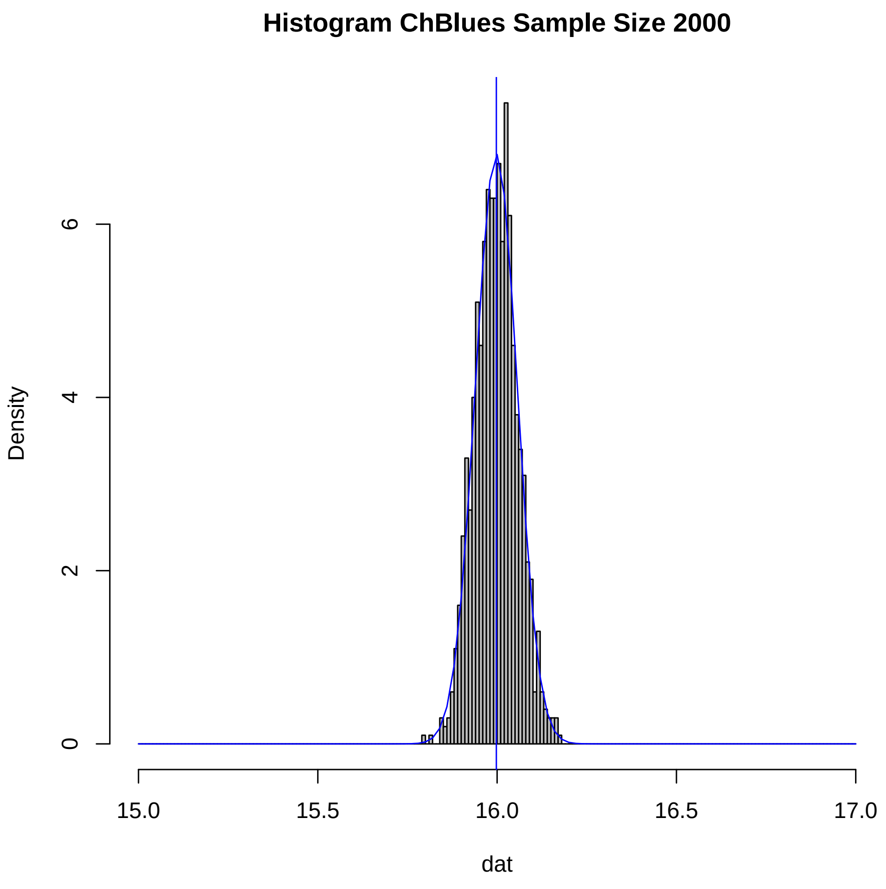


Figure 73: Histogram of the averages from 1000 samples with $n = 2000$, mean = 15.99779

It can now be safely concluded that random sampling works for this record and any other which has a similar probability distribution.

The tests show where the weaknesses are, especially for the minimization technique and the rating algorithm. But there are also some parts of RENE which deliver stable and exact results such as the silence sound transition detection.

6 Conclusion

This chapter will conclude the work which was carried out in the entire project. For each major part of the thesis a distinct conclusion will be drawn.

6.1 Discovery of quiet parts

This part of the project went well and did not pose major problems. The techniques with the peak frequency detection and Wavelet denoising deliver stable results and did not fail any test sample. However, with the current implementation only the first quiet section will be detected. Moreover it is not entirely clear if every record will contain a peak at the higher frequencies for the noisy parts. Due to the characteristics of white noise this fact is highly likely to hold true for almost any record.

6.2 Minimization

The minimization part is hard to sum up. Most of the records seem to be extracted in a way which is close to the optimum. In general the functions for the MD algorithm seem to be stable. This is harder to visualize for the ED1 algorithm which has too many parameters. However the result of the minimization is strongly dependent on the function which evaluates the process of the edge detection. The SNR was the most logical choice which took into account most of the characteristics which were important for the overall sound quality. But it has a major flaw, if the energy of the noise decreases faster than those of the sound, the distance may improve but in general the sound may be inaudible.

Another big issue is the minimization algorithms. They have been created for continuous functions, the application of them to discrete functions as in this case is imprecise at best. Especially the conjugate-gradient method requires a close approximation of the gradient to calculate the direction for the line search. Moreover the solution space is limited to positive numbers for many parameters, this requires further changes. Nelder-Mead for example takes into account boundaries for each parameter. The case for the conjugate-gradient method is more difficult, the translating function will simply take the absolute value, which corresponds to a simple mirroring of the function on the axis.

6.3 Edge detection with the second derivative

This edge detection did not at all deliver the results which were hoped for. It introduces more noise due edges which are more often detected in the wrong place compared to results of those of the first derivative approach. In general it will deliver results similar to those of the ED1 algorithm in the best case, but most of the time the overall noise intensity raises considerably.

6.4 Perspectives

Most likely the minimization approach is not going to raise the quality of the retrieved sound files considerably. Therefore a completely new approach has to be taken. One of the methods which was proposed during the project but could not be analyzed due to time constraints was the measurement of optical flows. This technique will analyze the incremental changes in each image and create a table of vectors which describe the movement. These vectors could be used to extract the movements of the grove.

6.5 Acknowledgements

I would like to thank Mr. Johnsen for his recommendations and advice during the weekly meetings. Moreover I am grateful for the numerous remarks on the various intermediary versions of my report from Mr. Bapst. The countless ideas from Mr. Haber were inspiring and often very helpful. Mr. Cornell and his immense knowledge of RENE and the entire acquisition process were also very helpful on many occasions.

This has been a highly interesting and diverse project in an amazing location with the opportunity to meet many exceptional people.

7 Appendix

This chapter contains additional information which did not fit into the previous chapters.

7.1 Configuration for Tests

A short overview of the four records which were used for all tests except the silence sound transition.

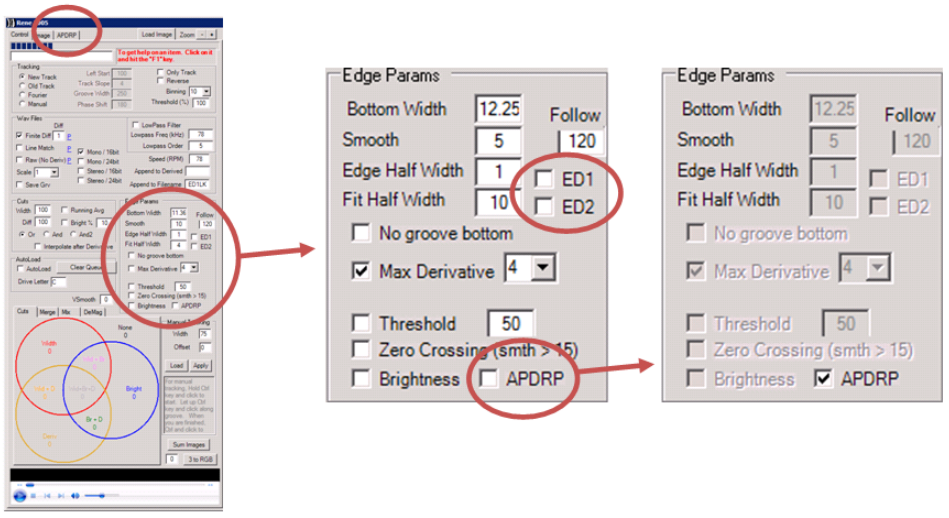
Name	Quality	Age
ChBlues	Very noisy	early 1920s
DCS	Moderate	1940s
Silver Threads	Good condition	1930s
WB	Very old	Before 1920

7.2 How-to for the new Functionalities

This subchapter contains a short introduction into the options which were added on the GUI for the configuration of the minimization process and the detection of silent parts.

7.2.1 Main GUI changes

The changes on the front GUI [Fig: 74] were minor; only three checkboxes were added to the “Edge Params” box. Two of these are the new edge detecting algorithms, called ED1 and ED2. The other one called APDRP is used to indicate when a minimization should be performed.


Edge Params
Bottom Width 12.25
Follow 120
Smooth 5
Edge Half Width 1
Fit Half Width 10
[ ] ED1
[ ] ED2
[ ] No groove bottom
[✓] Max Derivative 4
[ ] Threshold 50
[ ] Zero Crossing (smth > 15)
[ ] Brightness
[ ] APDRP

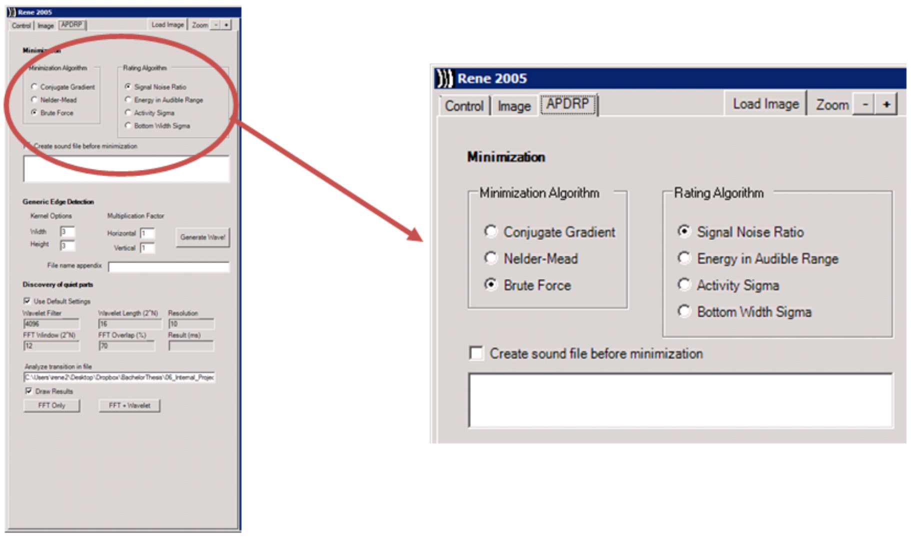
Edge Params
Bottom Width 12.25
Follow 120
Smooth 5
Edge Half Width 1
Fit Half Width 10
[ ] ED1
[ ] ED2
[ ] No groove bottom
[✓] Max Derivative 4
[ ] Threshold 50
[ ] Zero Crossing (smth > 15)
[ ] Brightness
[✓] APDRP

Figure 74: Changes to the main interface

In order to adapt the parameters of the APDRP option a new panel has been added. Note that the activation of the APDRP checkbox will automatically block all modification on any other element in the same group.

7.2.2 Options for the Minimization

There are two important options for the minimization process [Fig: 75] . The algorithm which will be used to locate a minimum, which can be chosen in the left Box called “Minimization algorithm”. And there is the option of choosing different rating algorithms for the output.



Minimization

Minimization Algorithm: Conjugate Gradient [✓ Brute Force] Nelder-Mead

Rating Algorithm: [✓ Signal Noise Ratio] Energy in Audible Range, Activity Sigma, Bottom Width Sigma

☐ Create sound file before minimization

Generic Edge Detection

Kernel Options: Width 3, Height 3

Multiplication Factor: Horizontal 1, Vertical 1

File name appendix

Discovery of quiet parts

☑ Use Default Settings

Wavelet Filter 4096; Wavelet Length (2^N) 16; Resolution 10

FFT Window (2^N) 12; FFT Overlap (%) 70; Result (ms)

Analyze transition in file: C:\Users\rene2\Desktop\Dropbox\BachelorThesis\06_Internal_Projec

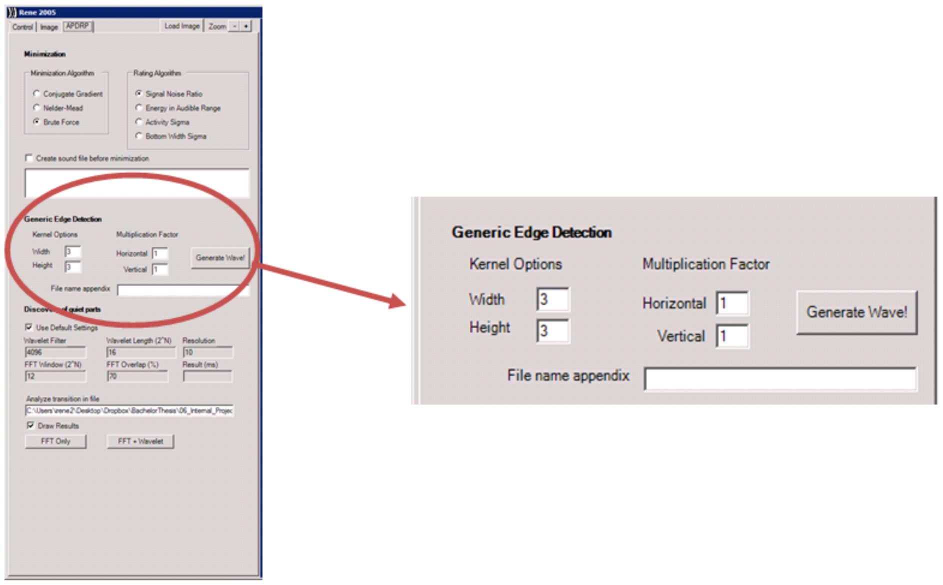
☑ Draw Results

Figure 75: Options for the Minimization

Moreover there is a checkbox which decides if a sound file will be written to the disk with the default settings. This will allow the user to compare the results of the minimization with the standard settings. The text box in the bottom of the minimization part is only used for output purposes. The result of the minimization will be printed as well as the intermediary steps to inform the user of the progress. However the parameters which were found to deliver the best sound quality will be exported directly to the interface before the final run is executed.

7.2.3 Options for ED1 or the Generic Edge Detection

The second part contains four specific parameters for the ED1 algorithm. The two fields [Fig: 76] on the left contain values for the height and width of the kernel. The larger the kernel the more information will be taken into account for the edge detection. A larger kernel will slow the edge detection process down. The values have to be integers, odd and positive, or the algorithm will fail with an exception. The two fields on the left contain the factors for the weighting inside the kernel. They have to be integers and bigger than 1.



Minimization

Minimization Algorithm: Conjugate Gradient, Nelder-Mead, [✓ Brute Force]

Rating Algorithm: [✓ Signal Noise Ratio], Energy in Audible Range, Activity Sigma, Bottom Width Sigma

☐ Create sound file before minimization

Generic Edge Detection

Kernel Options: Width 3, Height 3

Multiplication Factor: Horizontal 1, Vertical 1

File name appendix

Discovery of quiet parts

☑ Use Default Settings

Wavelet Filter	Wavelet Length (2^N)	Resolution
4096	16	10
FFT Window (2^N)	FFT Overlap (%)	Result (ms)
12	70	

Analyze transition in file: C:\Users\rene2\Desktop\Dropbox\BachelorThesis\06_Internal_Projec

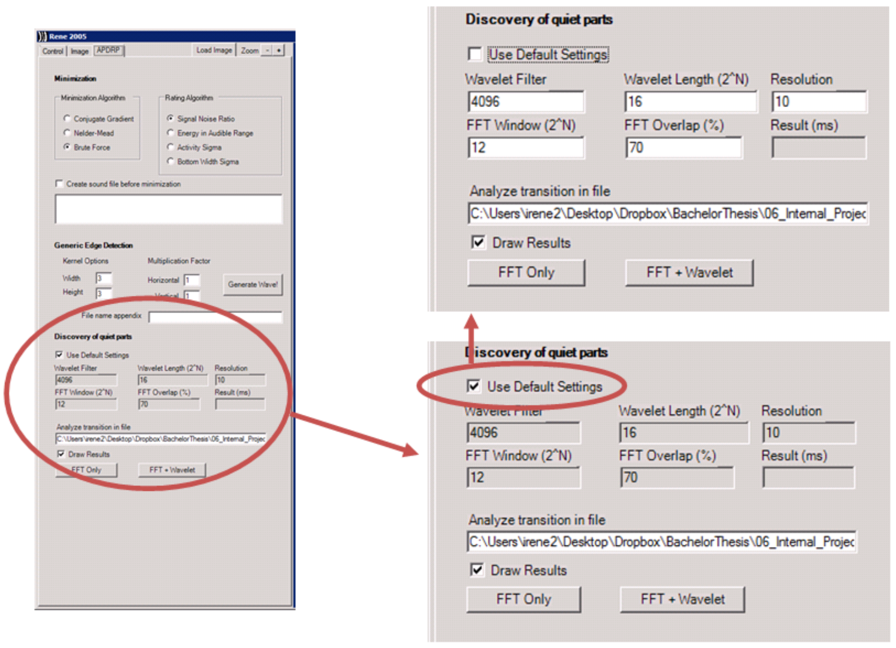
☑ Draw Results

Figure 76: Options for the generic kernel

The button “Generate Wave” should only be used when an images has been loaded into the program. It will redo the edge detection process and write a sound file to the disk. The “file name appendix” will add the text in the box to the filename. If the field is left empty, the parameters for the edge detection will be used as file name appendix.

7.2.4 Options for the Detection of Silent Parts

The last part [Fig: 77] contains the settings which are used for the detection of quiet parts. If not specified otherwise the default settings will be used which are displayed in the locked fields. Once the user decides to change the values the fields will be unlocked, but the default values will remain the same. The 5 fields must contain positive integer values. The “Wavelet Filter” value should not be bigger than 2 to the power of the “Wavelet Length” field and not smaller than 512. The “Resolution” is measured in milliseconds. The final result will be a multiple of this field. The “FFT window” size should not exceed 16, otherwise the result might be too imprecise. The last parameter field defines the overlap percentage of the FFT windows. It should be between 1 and 99.


Discovery of quiet parts

☐ Use Default Settings

Wavelet Filter	Wavelet Length (2^N)	Resolution
4096	16	10
FFT Window (2^N)	**FFT Overlap (%)**	**Result (ms)**
12	70	

Analyze transition in file: C:\Users\irene2\Desktop\Dropbox\BachelorThesis\06_Internal_Projec

☑ Draw Results

Discovery of quiet parts

☑ Use Default Settings

Wavelet Filter	Wavelet Length (2^N)	Resolution
4096	16	10
FFT Window (2^N)	**FFT Overlap (%)**	**Result (ms)**
12	70	

Analyze transition in file: C:\Users\irene2\Desktop\Dropbox\BachelorThesis\06_Internal_Projec

☑ Draw Results

Figure 77: Options for discovery of quiet parts

The text field should contain a valid path when pressing on the “FFT Only” or the “FFT + Wavelet” button. The file will be loaded and analyzed with either the FFT or both techniques. The result will be printed in the “Result” box. However if one requires more detailed output, the “Draw Results” check box will indicate the program to draw the latest results including the detected transition in a graph on the left side of the GUI.

7.3 Glossary

Technical vocabulary and abbreviations which have been used in this report will be explained for readers with a non-technical background:

Simplex Is a term from geometry. A simplex is generalized triangle in arbitrary space.[13]

Polytope Is also a term from geometry. A polytope is a the generalization of a polygon in arbitrary dimensions.[14]

Seed point Point in an image or space where the algorithm starts.

Edge An edge is point where the intensity changes abruptly.

GUI Graphical User Interface, the surface of software which allows the user to interact with a program.

SNR Abbreviation for Signal Noise Ratio which describes the ratio of the energy in the signal and the noise

RMS Abbreviation for Root Mean Square, a statistical measure.

MD Abbreviation for Max Derivative, the most recent version of the edge detection prior to this thesis

ED1 Abbreviation for Edge Detection 1, where 1 stands for the first derivative upon which the algorithm is based.

ED2 Abbreviation for Edge Detection 2, where 2 stands for the second derivative upon which the algorithm is based.

7.4 Structure of the CD

A short overview of the contents of the CD which was submitted with this report. In order to enhance the readability the _ characters have been replaced with an empty character in the report.

1. Information
2. Specification
3. Analysis
   (a) Data Representation
   (b) Multidimensional Minimization
   (c) Sound
4. Log
5. Internal Project Files
   (a) Code
   (b) Image
   (c) Mixed
   (d) Records
   (e) Sound
   (f) Tests

6. External Project Files
7. Report
   (a) Images
   (b) Subsections
   (c) Weekly
8. Presentations
   (a) Week 3
   (b) Week 10
9. Links and Sources

References

[1] Fred James, *MINUIT Tutorial - Function Minimization*, June 16, 2004.

[2] C. C. Kankelborg, *Multidimensional Minimization*, March 24, 2009.

[3] Jonathan Richard Shewchuk, *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*, August 4, 1994.

[4] Nonlinear Conjugate Gradient Methods, *http://reference.wolfram.com/mathematica/tutorial/UnconstrainedOptimizationConjugateGradientMethods.html*, August 2012.

[5] LabVIEW 2010 Advanced Signal Processing Toolkit Help, *http://zone.ni.com/reference/en-XX/help/371419D-01*, June 2012.

[6] Nelder–Mead method, *http://en.wikipedia.org/wiki/Nelder%E2%80%93Mead_method*, June 2012.

[7] Frequency Range of Human Hearing, *http://hypertextbook.com/facts/2003/ChrisDAmbrose.shtml*, June 2012.

[8] The Frequencies of Music, *http://www.psbspeakers.com/articles/The-Frequencies-of-Music*, June 2012.

[9] Human voice, *http://en.wikipedia.org/wiki/Human_Voice*, June 2012.

[10] Ottar Johnsen, Michael Ansorge, *Digital signal processing*, September 15, 2011.

[11] John A. Rice, *Mathematical Statistics and Data Analysis*, Third Edition, 2007.

[12] Wavelets and Signal Processing, `http://www.bearcave.com/misl/misl_tech/wavelets/index.html`, June 2012.

[13] Simplex, `http://en.wikipedia.org/wiki/Simplex`, August 2012.

[14] Polytope, `http://en.wikipedia.org/wiki/Polytope`, August 2012.