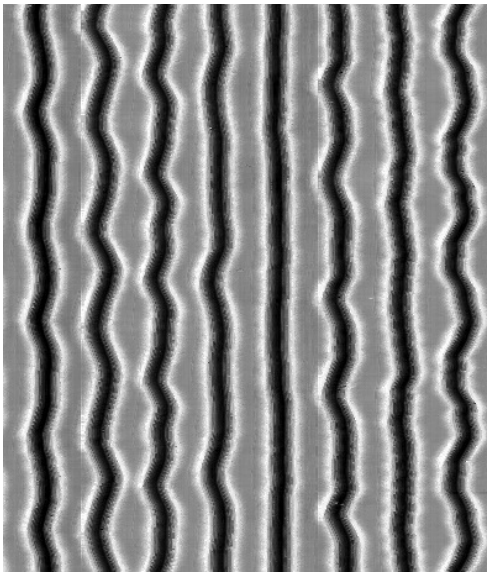
Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

Bachelor Thesis - Computer Science

Noise reduction on old records



Berkeley, August 4, 2017

Author:
Matthias Christe

Supervisor:
Carl Haber

Expert:
Noé Lutz

Professors:
Frédéric Bapst
Nicolas Schroeter

BERKELEY LAB

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts
Western Switzerland

Abstract

The history of old records is long. It begins with the appearance of the wax cylinders and is followed by the invention of the discs and vinyls and finishes in modern times with the arrival of the compact disc. Some of these records are very old and damaged. They have to be manipulated with precaution. The IRENE-team at Lawrence Berkeley National Laboratory has been working for many years to extract sound from old records in order to retrieve the precious data that they might contain. A large scanning campaign is underway for wax cylinders. Another campaign is planned to scan aluminum discs.

The team has a system that can carry out 2D and 3D scanning. 2D scannings have already been the subject of previous projects. This project focuses on 3D scannings based on aluminum discs. These aluminum discs contain dust particles and scratches that need to be treated. An idea imagined by Carl Haber from the IRENE-team consists in creating an image that contains no noisy points and no scratches. This image can be subtracted from the original image and the result will be an image containing noisy points and remains of audio information. This solution called universal shape has been implemented and tested and is very successful. Once the points detected, the sound needs to be extracted from the image. An idea based on a polynomial fitting has been imagined. This idea gave rise to other ideas and the project ends with four solutions. The results show that these ideas are good. However, improvement can be made in order to obtain a better audio quality.

Contents

List of Figures

Chapter 1

Introduction

1.1 Context

Nowadays, there are a multitude of possibilities to listen an audio file. Smartphones, Computers and CDs are now the most common ways used by people to achieve that. However, it is always possible to use MP3 players or vinyl records even if these are becoming more and more rare. The vinyl records are sometimes used by some DJs or by some audio enthusiasts, that prefer the sound of a good old vinyl over the sound of a CD[1]. It is now easy to listen to audio but what about the first recordings ?

1.1.1 Analog recording technology

Before explaining the history of the various recordings, and in order to avoid any confusion, the different techniques of analog recording will be explained below.

Analog records store the audio signal as a continuous signal. It may be stored as a physical impact (phonograph, gramophone) or as a fluctuation in the field strength for magnetic records[9]. The magnetic recordings are not interesting for this project and are not explained in detail.

An audio signal has the following characteristics. Usually, the frequency range is between 20 and 20,000 Hz. Outside, it would no longer be audible to a human.

Basically, the sound goes through a diaphragm that results by vibrating a needle, also called stylus. This needle can vibrate laterally or vertically and will etch the sound wave on the record. If the needle is vibrating laterally, the recording will be called *lateral cut record*, else it will be called *vertical cut record*.

The Edison cylinders were always cut vertically. Later, some discs records were also cut vertically. However, it was a minority (records from the Volta Laboratory).

[1] Harry Pearson, an audio critic, even said that the CDs drained the soul of the music (See section 6.7.4 - Analog warmth in the following paper[9])

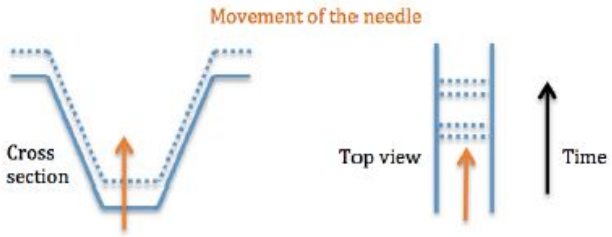


Figure 1.1: Vertical Cut system (image taken in the thesis of Romain Crausaz - section 2.1 [2])

However, the majority of the discs were cut laterally.

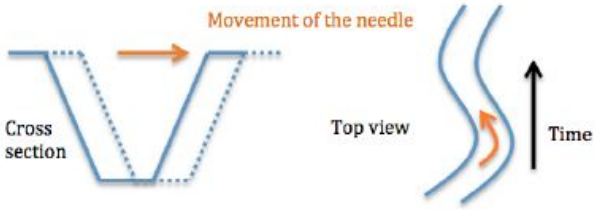


Figure 1.2: Lateral Cut system (image taken in the thesis of Romain Crausaz- section 2.1 [2])

1.1.2 The analog recording across the ages

In 1857, Édouard-Léon Scott de Martinville patented the phonautograph. It could not actually play back the sound but it was not necessary at the time because this inventor wanted to study the result. It consisted of sheets of paper on which lines have been drawn that represent the sound waves modulation. It was only in 2008 that a team was able to replay the sound and was able to appreciate the French song *Au Clair de La Lune*. The phonautograph was also the first analog recorder.

The phonautograph was only the beginning. 20 years later, Thomas Edison invented the phonograph (in 1877). It is the oldest sound recorder/player in story of mankind that actually worked. At first, the sound was recorded on tinfoil cylinders and later, thanks to Alexander Graham Bell, on wax cylinders. The system captured the sound via a microphone diaphragm that conducted to vibrate a small needle / stylus which in turn etched the sound on the cylinder. By this way, the groove produced represents the sound. The stylus performs vertical movements, depth represents the analogous sound signal. The playback process is basically the inverse process of the recording.

The drawbacks of this system are the following ones. They are easily damaged by any contact. The human interaction is one thing that could be diminished by taking precaution, but the main problem lies in the reading part. Each time that the sound was played, a needle followed the groove by a continuous contact and blanched the groove.

> "This meant that every single time a recording was played, it was one step closer to being gone forever." (Section 3.1.3 - Phonograph problems [9])

Another problem of the phonograph is that it suffers from lack of fidelity. It is extremely low and the recorded sound is far apart from the original sound.

Figure 1.3: A wax cylinder containing audio - Thomas Edison Invention

In 1887, Emile Berliner improved the phonograph to record the sound on discs: the disc phonograph, sometimes called the gramophone. This offers several advantages. They can be easier stored and can be faster and more economically reproduced. These discs have been used in mass production. They had also some disadvantages that have been inherited from the original phonograph. The low quality as well as the risks of damage (human and playback) stayed. In contrast with the Edison phonograph, the stylus of the gramophone made lateral vibrations (lateral cut system). This was the beginning of the era of discs. In the 1920s, the disc record experienced its first crisis due to the proliferation of the radio. Many discs industries are being forced out of business. Fortunately, the discs survived this and improvements were made in the 1930s. It was during this same period that the vinyl made its appearance. At first, it was used by the radio but soon became available for everyone. It was made of vinylite and shellac compound. The speed was at the start of 78 revolutions per minute (RPM). After that, other vinyls arrived on the market. These had a speed of 33 RPM and 45 RPM and were mostly made exclusively of vinyl but some were also manufactured out of polystyrene. After 1945, the vinyls were the new trend because they allowed to record more music. Those were called long-playing records and could contain an entire album. Unfortunately, they could not re-record a sound, they were a playback-only medium.

In the 1960s, a new record began to compete with vinyl: the compact cassette. Initially low in fidelity, this audio medium commercialized by the Philips electronics company was not appropriate to record music. Several years later, this problem was solved and the cassette became very popular thanks to the marketing of the Walkman. It allowed you to listen to music at any time. The cassette had the advantage that it made it possible to re-record the audio. A lot of improvements have been realized thanks to the *Dolby A noise reduction* system. Invented by Ray Dolby, this system reduced the noticeable hiss present on the cassette tape. This resulted from the miniaturization of the format.

In the end of the 20th century, the era of analog records was approaching its end with the arrival of the Compact Disc (CD). This digital recording reproduces the music without hiss and noise and is much more resistant to scratches and damages of any kind. Later, DVD, Blu-ray Disc become available and have definitively put an end to the analog era to move into the digital era.

1.1.3 Differences between Analog and Digital

It is not really clear that the digital recording is better than the analog recording.

> "It is highly dependent on the quality of the systems (analog or digital) under review, and other factors which are not necessarily related to sound quality" (Section 6.1 - Overview of differences [9]).

One of the main differences is that a digital duplication is an exact copy in contrast to an analog duplication. This offered a considerable advantage for the music industry. Another difference is that a digital recording can apply an error correction on the sound. The only disadvantage is that it takes about 20% of the place. It saves redundant data and mixes up the bits in a predetermined way. By this way, if there is a scratch, it won't corrupt the audio file. It can even resist to hundreds of imperfections. Of course, digital recording can also lose data. This was one of the reasons for switching to digital because a small defect on an analog disk could corrupt a lot of data. Analog records had also a lot of noise. Sometimes, it was possible to hear a rumble. A rumble is a noise caused by imperfections of the recorder during the recording process.

1.1.4 About IRENE Project and scope of the project

Dr Haber and his team had an idea. A lot of old records have been collected through the years and could not be played back. They decided to create a project called IRENE. This would take pictures from the record and insert the pictures into a software and play back the sound.

Different devices exist to take the pictures. There is a 2D system and a 3D system. The 2D system is basically a very good camera. It only takes pictures and is not able to give information about the depth of the groove but it could anyway help to find the motion of the signal. They have two 3D cameras: MPLS and Precitec. The one that is actually used is the Precitec. The reason is that it has a better resolution. The disadvantage of the Precitec is that it has a smaller range (200 µm) compared to the 350 µm of the MPLS. The IRENE team thinks that a higher resolution is better than a higher range.

The project currently use 3D images but has worked with 2D images. A software has been implemented at the beginning of the project that used only 2D images. This software has not been modified for a long time. Later, another software called *Prism* has been developed that can actually track the sound on 3D images and replay it. It applies a noise reduction on the image called blob detection. This software, based on C#, works very well but is slow and is not well structured. For those reasons, the team decided to create a new software called *IRENE* also based on C#. It is based on a plugin system where each functionality can be inserted and moved at any time. It has not yet integrated all the features of Prism. In addition to this plugin structure, this new software uses threads to compute the different algorithms. The contribution of the project has to be implemented in IRENE. Another feature of IRENE is that it will also make it possible to the user to use 2D images and 3D images.

Scope of the project: One problem already mentioned in the previous section is that the discs as well as the cylinders can easily be damaged or soiled. This noise is a huge problem and algorithms have already been implemented by previous students to remove it. Those work well on cylinders but are not perfect concerning the discs. The main goal of this project is to improve the Noise reduction on old discs. The laboratory has a few discs from the Milman Parry collection as well as some other records. Those have been sent by

the Harvard University. This project will mainly use these discs to test the implementation. However, it is important to keep in mind that there are other discs.

Milman Parry was born in 1902 and studied at the University of California in Berkeley. After that, he went to Paris(Sorbonne) to write his Ph.D. He was a poetry and was the founder of the discipline of oral tradition. He went several times to Yugoslavia to study the oral poetry. During this period, he recorded many aluminum discs (over 3000) in order to preserve the stories.

Figure 1.4: Milman Parry's aluminum collection

1.2 Goals

As mentioned, the main goal of this project is to improve the noise reduction on old discs that have been cut laterally. Two algorithms already exist and can be used. New ideas have emerged and will be tested during this project. The idea that will be tried first is the following one. A tracking will be applied on the image in order to find the audio signal. That is achieved by using a lower resolution of the image, result of a binning algorithm. These two parts are already implemented. After that, the algorithm will track the sound a first time in order to create a universal shape. Once done, it will loop a second time through the tracking points and adjust the universal shape at each point. It subtracts this adjusted shape to the original one and will produce an image containing most of the blobs as well as few audio data. A blob detection will be applied and the result will be added with the original image. Normally, the blobs will disappear and the sound audio will still be intact. Finally, the tracking will be followed again and an interpolation will be made at each point by using the shape of the groove. The interpolation will give a function which can be minimized. The minimization will give a new point that will represent the real position of the sound. The last task is to create an audio file by using these new values. The universal shape can be made locally or globally, both will be tested during this project.

1.3 Source Code Management

GIT is used to manage the source code. It allows the developer to manage different versions of an application. The development is conducted on a repository provided by the IRENE team.

`https://bitbucket.org/ewcornell/irene.git`

1.4 Approach and structure of this document

The approach of the project was the following. An idea has been imagined by Carl Haber of the Lawrence Berkeley National Laboratory. This idea has been analyzed and consisted of two parts. The first part consisted of finding the noisy parts on the aluminum discs by using the concept of flat and sub-flat image. The second part consisted of handling the noisy parts and of recreating the sound by using a polynomial fitting. The analysis and the time allowed to explore different solutions of polynomial fitting. Four solutions have been analyzed and implemented during the project. The entire report reflects that four solutions have been found. This begins in the analysis and ends up in the testing chapter. The report also reflects the difference between finding noise and treating it.

But before doing all that, it was necessary to understand the system of scanning the discs as well as the current state of the code. Indeed, the contribution of this project must be integrated into an existing project. Finally, it was also necessary to understand how to test and validate the solution.

The report contains the following elements:

- Introduction: describes the historical context of the project as well as the goal of this project.
- Analysis: contains the analysis of the scanning system and the existing project. Contains also the analysis for the noise detection and for the polynomial fitting solutions.
- Design: designs the algorithm that will be implemented for the noise detection and the different algorithms that will be implemented for the polynomial fitting idea.
- Implementation: contains three important parts of the Code that deserve attention. it also contains a documentation of all the plugins that have been implemented during the project. This part has been written for people who would like to understand how to use the different plugins.
- Tests and Validation: contains the results for the different tests that have been carried out. The procedure as well as the parameters that have been used are also presented. Similar to the analysis, this chapter is divided in two parts: noise detection and polynomial fitting.
- Improvements: contains a short explanation about the different improvements that could be made to improve the results.
- Conclusion: summarizes what worked and what did not work. It also describes the future work.

Chapter 2

Analysis

This chapter is the second step in the software development life cycle. In this case, it implies many things. The first thing is to analyze the scanning system in IRENE. They use a software that needs to be analyzed in order to carry out scanning autonomously. Once the scanning done, a software is used to process the scanning and to recreate the sound. This software contains some features that will be used. It is important to understand the concepts behind the software in order to implement the project's contribution effectively. The general idea of the project is also analyzed. Last, it is not a project that allows to integrate tests automatically. This project requires a testing phase on its own. How to test will be the last part of this analysis phase.

2.1 Analysis of the Lab View software

In order to reproduce the sound, the aluminum discs need to be scanned. To do that, it is possible to use 2D systems or 3D systems. 2D systems consists of taking pictures with a high-resolution line scan camera BASLER rAL4096-24gm. 3D systems consist of getting the depth at each point by using laser-optical systems. The IRENE team has many 3D systems like the Precitec and the MPLS. The files that this system produces are TIFF files. TIFF files are used by PRISM and the IRENE-plugin system to reproduce the sound. This section presents how a scanning work and how the LabView software, that makes the scanning possible, work.

2.1.1 How does the scanning work?

The aluminum disc lies on a vertical rotating motor. The precitec or the MPLS lies on an air-table. When the scanning begins, the disc begins to rotate with a time interval between two positions. During this time, the Precitec or the MPLS measures the depth on a certain range. The Precitec has a range of 960 µm and the MPLS has a range of 1800 µm. Because it is a laser-optical system, it needs enough time to measure the depth. This time is called exposure time by the IRENE team. Once this time has elapsed, the disc starts to rotates again until it reaches the next location. And so goes on, until the disc has made one revolution. At this point, the system saves the depth values that have been collected in a TIFF file. The disc rotates continuously until it reaches the initial position again. This is made because it can be that the rotor did not rotate uniformly during the entire revolution. This means that the initial position and the final position after one revolution might not exactly match. Usually, one rotation is not enough to see this phenomenon but many revolutions can lead to this problem. Once the disc is placed again on the initial position, the system can start to measures one revolution of the disc again. During this project, the Precitec is the system that has been used because it has a higher resolution.

This system can be seen in the figure 2.1.

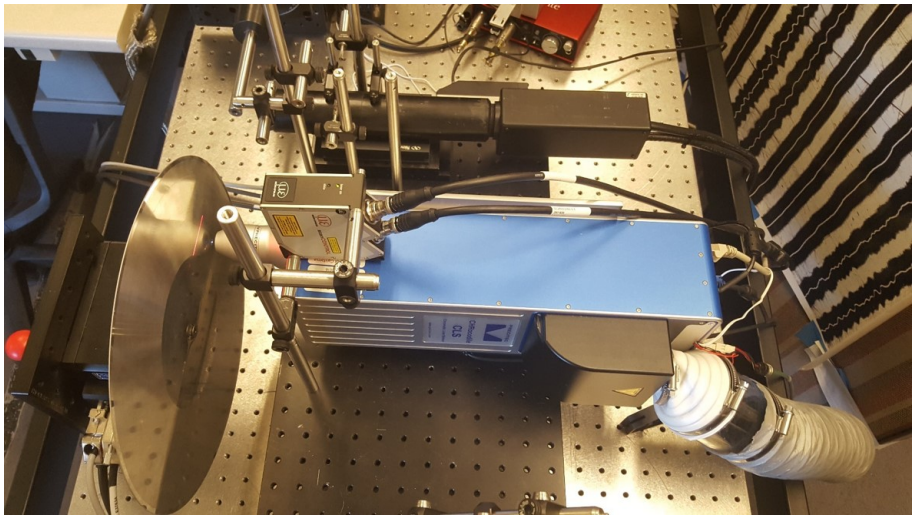


Figure 2.1: IRENE system with the precitec

The disc, on the left, rotates while the Precitec, in blue on the right, is measuring the depth at each point. The files produced by this system are TIFF files and are used in the other IRENE software.

2.1.2 LabView IRENE software

A LabView program called IRENE 3D has been developed in the last years that make it able to scan the aluminum discs. The graphical user interface (GUI) of this program is shown on figures 2.2 and 2.3:

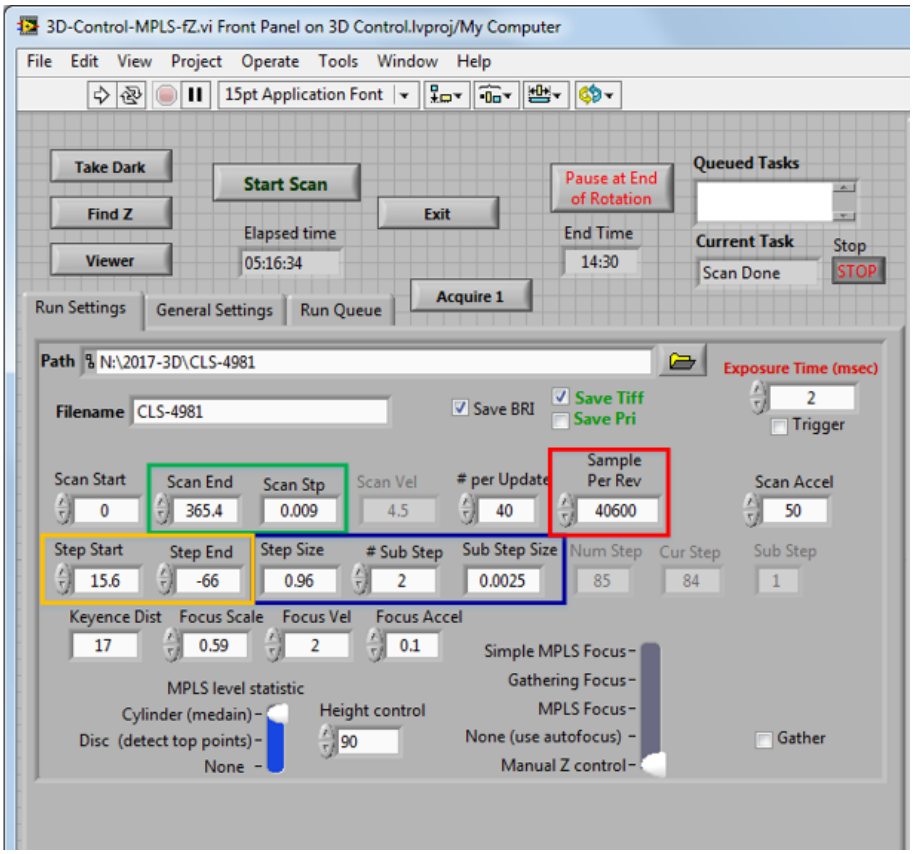


Figure 2.2: IRENE's graphical user interface in LabView (1)

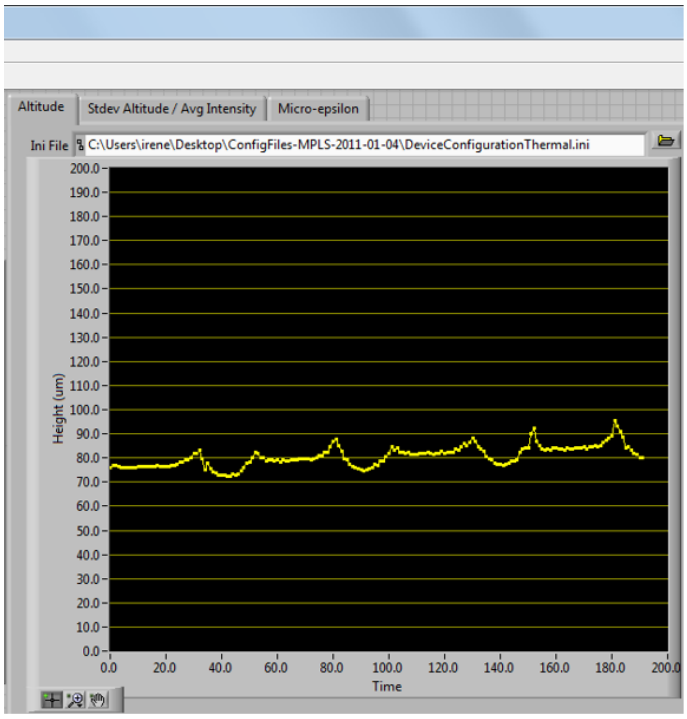


Figure 2.3: IRENE's graphical user interface in LabView (2)

In order to scan an aluminum disc, the user needs to know the values of several parameters before starting a scanning. The first thing is to know where the groove begins

and where the groove ends. The viewer part in the IRENE software (that can be seen on figure 2.3) will help to find the groove. The viewer can be started by clicking on the button "Viewer" in figure 2.2 or by starting a scanning. The Precitec lies on a motor that moves laterally from the indexes 105 to -105. By moving the motor with a tool, it is possible to see the groove as it can be seen in figure 2.3. It only remains to find both the outer and inner limits of the disc. This will give two numbers *Step Start* and *Step End*. Those have to be inserted in the orange part of figure 2.2.

Scan Start and *Scan End* describes the size of one revolution. By this mean, it could be possible to make more than one revolution before the precitec moves. In the example in figure 2.2, 365.4°. This means that the rotating motor will make a little bit more than one revolution before the Precitec makes a lateral movement. This is done in order to make an overlap that will solve focusing problems at the end of the rotation. *Scan step* is the step of the rotating motor, in this case it is 0.009. This number is related to the sampling number *Sample Per Rev* in red on figure 2.2.

$$\frac{365.4^\circ}{40600 SamplesPerRev} = \frac{360^\circ}{40000 SamplesPerRev} = 0.009^\circ \tag{2.1}$$

The number 600 comes from the 5.4 extra degrees in order to make the overlap. the question could be why 40'000? The following formula helps to explain that:

$$40000 SamplesPerRev * \frac{78 RPM}{60 seconds} = 52 kHz. \tag{2.2}$$

The aluminum discs of the Milman Parry run at 96 kHz but a sampling of 46 kHz should give the same results. With the previous formula, it can be seen that the scanning guaranteed the 46 kHz. This also introduces another concept. The aluminum discs have a speed of 78 RPM. It is not always 78 RPM, it can also be 77.5 or 78.2. It is impossible to know the real speed if it has not been written somewhere. If the difference is as small as the previous examples, it does not really matter. If the difference becomes really high, it will make a huge difference. To simplify that, 78RPM is a good assumption.

40'000 samplings per revolution is a good number to find the 46kHz but 80'000 samplings per revoltions would be a better number:

$$80000 SamplesPerRev * \frac{78 RPM}{60 seconds} = 104 kHz. \tag{2.3}$$

This would be even better than the 96 kHz that we might want to reach. The problem is that the scanning will be twice as long.

Step Size is the lateral shifting of the precitec. This number is in millimeters. 960 µm is the width of the precitec range. If this number is smaller, it means that an overlap between two passes will happen. The disc will be scanned with an overlap of 160 µm. It means that this value will be 0.8.

Sub Step introduces the concept of sub-pass. This concept is explained in section 2.1.3. If the number is two, there will be two sub-passes. *Sub Step Size* is the shifting between the different sub-passes.

Exposure time is a concept that has already been mentioned below. It is the time that the Precitec has to scan the depth before two steps in one rotation of the disc. This number is in milliseconds. For the Precitec, this number can not be lower than 0.5. However, this value give results that are not precise at all. 1 millisecond should be a good parameter but the lens of the precitec can lead to change this value to a higher value.

2.1.3 Sub-Pass concept

When the precitec measure the depth at one point one the range of 960 µm. It measure 192 points.

$$\frac{0.96\mu m}{192 points} = 0.005mm \tag{2.4}$$

If the number of sub-passes is two, this number will increase to 384 points. The concept of sub-passes is presented in figure 2.4.

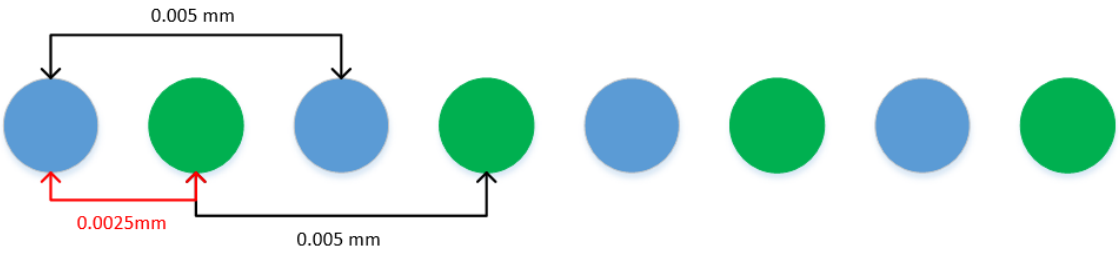


Figure 2.4: Sub-pass concept

The blue circle represents the first sub-pass and the green circles represents the second sub-pass. The Precitec takes a first scanning of 192 points spaced by 0.005 mm each. Then the system shifts the precitec from 0.0025 mm and make a second scanning. This number has been defined by the parameter *Sub Step Size* on figure 2.2. This corresponds to half of the distance between two points in a range.

The concept of sub-passes helps to have a higher resolution of the data.

2.2 Analysis of the IRENE software

It has already been mentioned in the section 1.1.4 that IRENE is a plugin-based system. It means that each functionality is a plugin that can be added to or removed from the user interface. It offers the flexibility to create different scenarios by changing the order of the plugins, by duplicating some plugins and by changing the parameters. Let us take a look about this plugin-based structure.

2.2.1 Graphical user interface

First of all, the graphical user interface (GUI) is presented in figure 2.5:

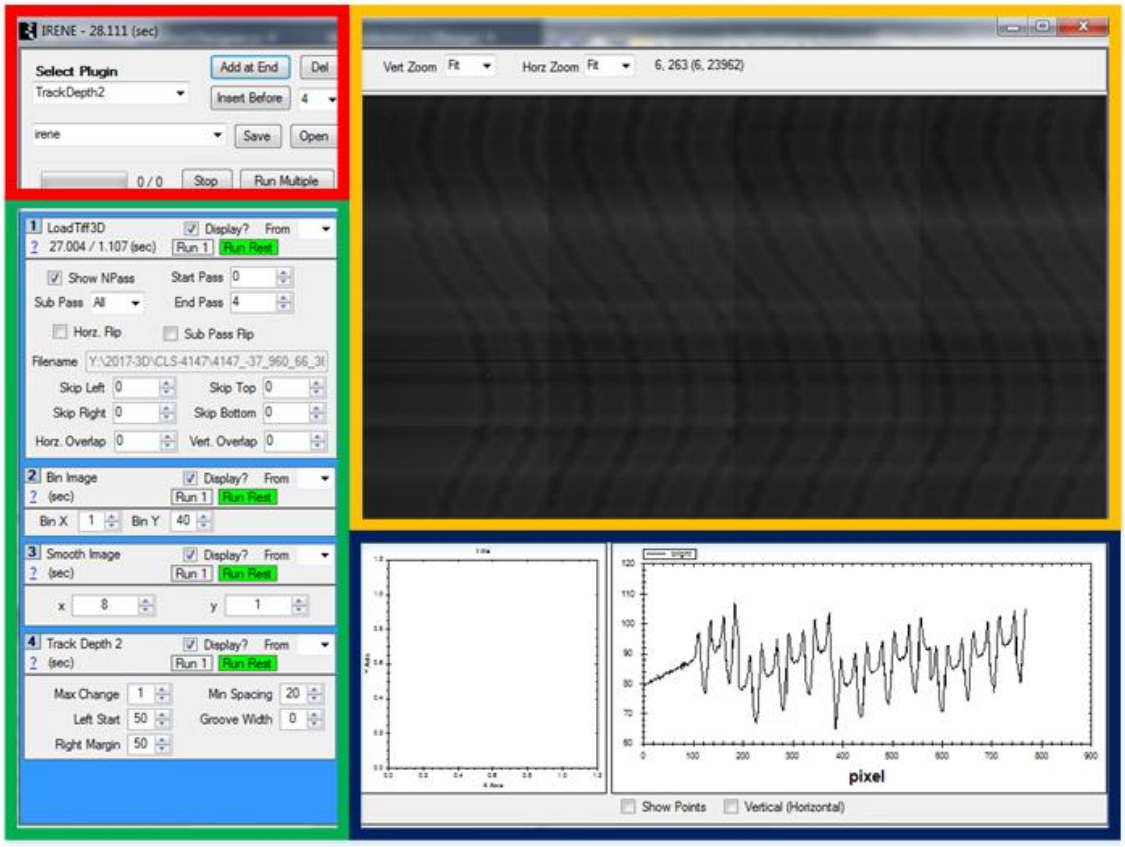


Figure 2.5: Irene's graphical user interface

It can be seen that it contains four areas. These are the following ones:

- Red: Insertion and deletion of plugins
- Green: Plugins area allows to modify the parameters and to execute the plugin
- Orange: Visualization area, if the check-box **Display** is checked for the plugin that is currently executed, the result will be displayed in this area.
- Blue: Graphic area. It displays a function of the depth of the image on the selected line or column.

In this example, four plugins are used. The first one loads the 3D image from the passes 0 to 4. These passes can be seen on the visualization where black lines split vertically the image in four parts. After that, the image is binned in the height. The idea of that is to lower the resolution of the image. The image is then smoothed, and the tracking is made. These plugins are explained more precisely below.

2.2.2 Presentation of the useful existing plugins

2.2.2.1 LoadTiff3D

This plugin takes a TIFF file as input. In the context of this project, a TIFF file can contain the depth or the brightness of the image. Those files are created by LabView, the software that takes the 3D images. For each disc, two TIFF files exist. The file name of the one containing the brightness information ends with the suffix *.bri.tif* and the file name of the other containing the depth information ends with the suffix *.tif*.

The plugin loads the selected file and inserts the values of the tiff files into an array. Usually, only the depth tiff file is used. This plugin is the first one that will be called in the project.

Before explaining the parameters, one must understand what a pass means. A pass is a ring where the difference between the inner radius and the outer radius is 1mm. After that, it takes a 3D line picture of the pass and once done, it goes to the next pass. This is explained in the following picture.

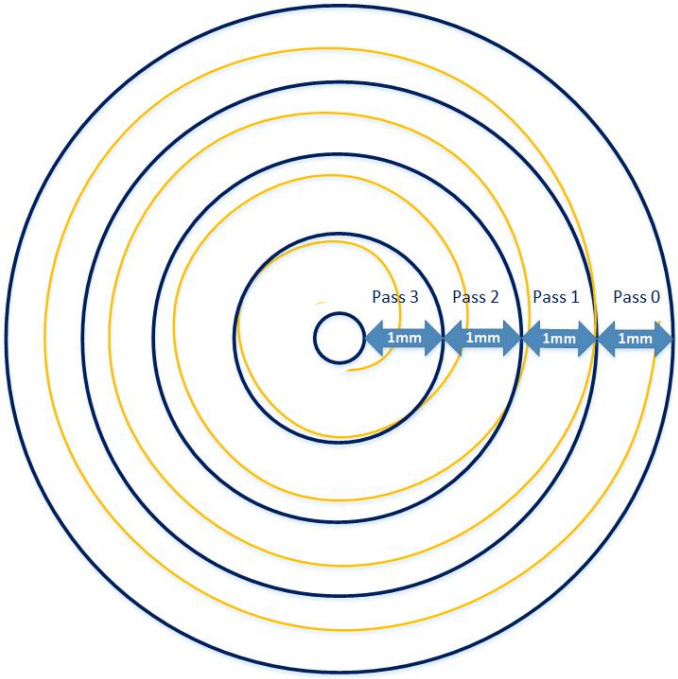


Figure 2.6: Explanation of the pass system - Example with four passes and a very small disc

In the previous example, each pass contains one groove but depending on the discs, it can be that it contains more than one groove in 1mm. The result is saved in a tiff file. This plugin is therefore able to select the passes that the user wants to use for this audio measure.

This plugin has many parameters that can be used to personalize the measure. These are the following:

- Start Pass - End Pass: range of the pass that is used for this measure. This can be used to select whether the measurement is taken over the entire disc or a part.
- Skip Left, Right, Top, Bottom: four parameters that can be used to skip pixels on each side.
- Horz|Vert Overlap : used if an overlap is needed between the passes.

To sum up, the tiff file will fill a two-dimensional array with the values taken by the precitec.

2.2.2.2 Bin Image

This plugin takes two parameters **BinX** and **BinY** that can be seen on figure 2.7. The idea is to concatenate different lines or columns together. By this way, the new image will be at a lower resolution of the original one. Usually, a binning in Y is made. Let us take

an example where BinY is 40. The plugin will loop through each group of 40 lines and average the values of the depth. At the end, the height of the image will be divided by 40 but the width will still be the same.

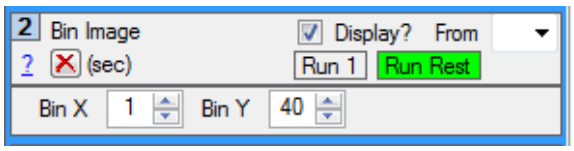


Figure 2.7: Parameters of the plugin "BinImage"

The advantage of the binning is that it lowers the resolution so that the tracking won't be too precise. Indeed, if the tracking is too precise, it will be more strongly influenced by the noise.

It is also possible to make a binning in X but it makes no sense in this scenario because the audio is stored on the motion of the groove. If a binning would be applied, the resulting image would no longer contain any valuable groove.

2.2.2.3 Smooth Image

Very common feature in Image processing, it consists of erasing the very small differences in an image. In this project, the image is a tiff file containing depth information. The shape of the groove present on the image is not perfect and may contain some irregularities. The smoothing will therefore flatten the curve a little bit like in the following example.

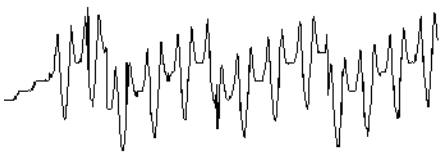


Figure 2.8: Before Smoothing (image taken from the software)

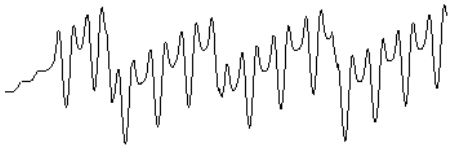


Figure 2.9: After Smoothing (image taken from the software)

These images have been taken with the software under the graphic part. They represent the horizontal curve on a same point. The figure 2.9 shows the curve before the smoothing (but after the binning) and the figure 2.9 shows the curve after the smoothing. It can be seen that the curve has lost some irregularities.

This may help the tracking to find the best center part of the groove. The reason is that if the bottom of the groove is not flat, it may be that the tracking does not correspond to the current groove.

2.2.2.4 ImageThreshold

This plugin takes an image as input and gives an image on which values will be either 0 or 127. The plugin works like a filter. It takes three parameters that can be seen on the figure 2.10:

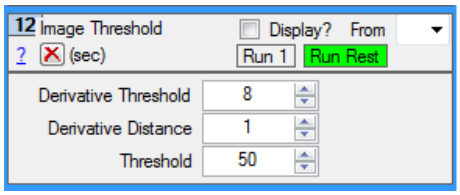


Figure 2.10: Image Threshold plugin - parameters

The first parameter allows a maximal difference between two values. It means that if there is a high difference between two points (8 here), the point will be labeled as a blob. The second parameter is used to check the derivative of a point with more distant points. This is a complement to the first parameter.
The third parameter is used to mark as blob all the points that have a higher value than this parameter. It means that if the points have a value higher than 50, they will be labeled as blobs.

Created Image This plugin will create an image containing 0 and 127. At the end, the image will look as follows:

0	0	0	0	0	0
0	0	0	0	127	0
0	0	0	0	0	0
0	0	127	0	0	0
0	0	0	0	127	0
0	0	0	0	0	0

Figure 2.11: Image Threshold array

2.2.2.5 WriteWav

This plugin is the last plugin that will be used during the process because it will create the audio file. This plugin requires a tracking. The different parameters can be seen on the figure 2.12:

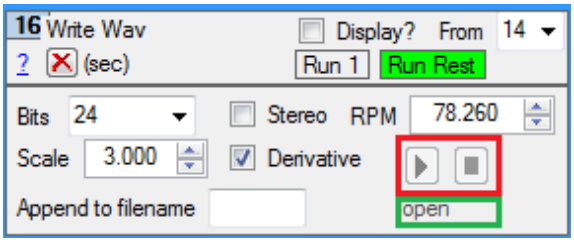


Figure 2.12: Write Wav plugin - parameters

Bits (24) is a parameter that defines how much place is stored for each data. If this number is higher, the number of possible values for the sound can be higher. It also means that the data file will be bigger. The *scale* parameter allows the user to change the volume of the audio file. It can be really helpful if the wish of the user is to compare the audio file with another audio file.

RPM is defined by the disc. The majority of the Milman Parry collection runs around 78 RPM. Because of the time and the loss of information, it might be that some of the discs run at 80 RPM or at 79 RPM...

Derivative means that the data that will be stored is not the tracking but the derivative of the tracking.

A red square can be seen on the figure 2.12. It allows the user to play and stop the sound by using the two buttons.
The green square is also a button and will open the audio file in Sound Forge Pro.

2.2.2.6 TrackWidth

Sometimes, the tracking is not in the middle of the groove. This is explained in more detail in section 2.2.4. In order to solve that, a plugin has been implemented that will, from a given point, find the edges on the sides. It requires an image and a tracking. It will produce a new tracking with two X points. The plugin will follow the tracking and find the two extremities of the groove by using the following concept.

The concept consists of drawing a line in the groove in such a way that it remains straight and that it corresponds to the desired width.

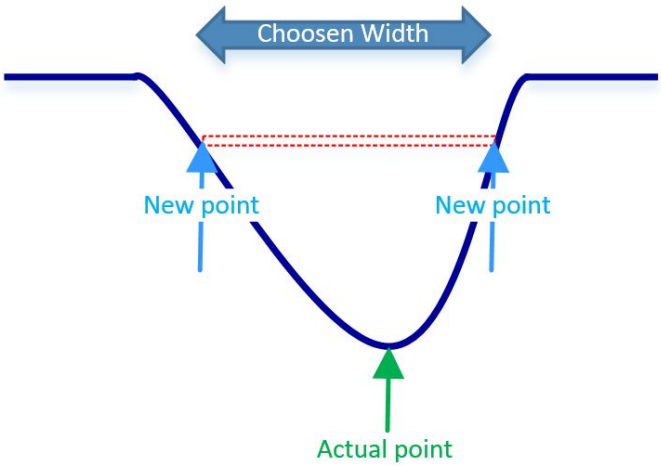


Figure 2.13: TrackWidth - Vertical Line concept

The interface of the plugin can be seen on the following figure:

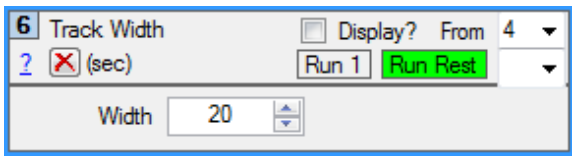


Figure 2.14: TrackWidth plugin - parameters

This plugin creates a tracking with two X points and one Y point. In order to compute the middle of this point, a new plugin is implemented. It is explained in the section 2.6.2.

2.2.2.7 Match Pass

The LabView system, that have been explained in section 2.1, make it possible to scan a disc with the concept of sub-passes. Reminder: a sub-pass is the concept of collecting a

higher number of points than the usual 192 points for one pass. If this number if two, the number of points will be 384. The plugin that is presented in figure 2.15 helps to mix the points together. Indeed, it will merge the two sub-passes together.

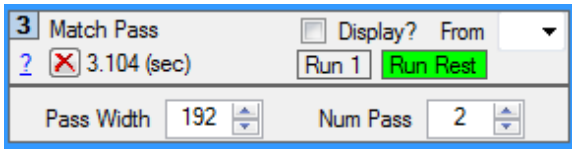


Figure 2.15: Match Pass plugin - parameters

This plugins takes two parameters: *Pass Width* and *Num Pass*. *Pass Width* represents the number of points for one sub-pass and *Num pass* represents the number of sub-passes. In this example, two sub-passes have been scanned for each pass.

This plugin is not necessary if the concept of sub-passes has not been used during the scanning.

2.2.2.8 Overlap Pass 3D

Another feature explained in section 2.1 is that the the user can choose that the shifting between two passes is smaller than the width of the precitec's range. If it is the case, there will an overlap of data. This helps to avoid the problem of focusing on the end of the pass. Indeed, it can be that the position of the groove does not exactly match when the system is scanning nearby the previous location.

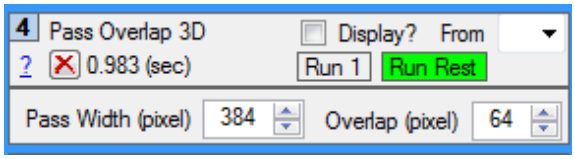


Figure 2.16: Overlap Pass 3D plugin - parameters

This plugins takes two parameters: *Pass Width* and *Overlap*. *Pass Width* is the real width of the pass by taking into account if the concept of sub-passes has been applied or not.

In order to set the value for the overlap, one should know if there is an overlap. If there aren't any overlap, this plugin would not be useful, so lets us assume that an overlap has been made and that the concept of sub-passes has been applied with two sub-passes. Reminder: the width of the precitec is 960 µm for 192 points but 384 points for two sub-passes. The shifting between two passes that have been chosen is 800 µm. It means that the difference is 160 µm. The distance between two points is 0.0025 mm. If 384 points represent a width of 960 µm, how many points represent a width of 160 µm? The answer is the good old rule of three:

$$384 * \frac{160}{960} = 64 \tag{2.5}$$

In this case, 64 points represent the overlap between two passes.

2.2.2.9 TrackCrop

Another overlap that has already been mentionned in section 2.1 is due to the number of degrees that have been set for one revolution. If this number is higher than 400, it might be necessary to remove these points at the end of the process. A plugin "Track Crop"

exists that will removes points at the beginning or at the end of each pass. In the example that can be seen in figure 2.17, 400 points have been cut at the end and the start of each pass. But why 400? To understand that, one must know that the number of samplings per revolution has been set to 40'800. 40'000 represent 360°. 800 points represents 7.2°because the step between two scans was 0.0009. 800 represent the overlap. It means that these points can be cut at the beginning or at the end of the pass. In this specific example, 400 were cut on both side of the pass.

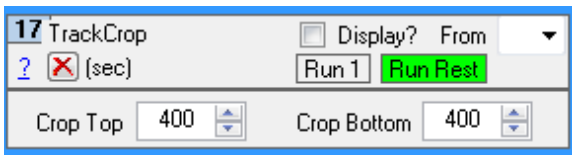


Figure 2.17: Overlap Pass 3D plugin - parameters

This plugin takes two parameters: *Crop Top* and *Crop Bottom*.

2.2.3 Tracking algorithms and TrackDepth2

Three tracking algorithms exist and can be used. *TrackManual* is a plugin that allows the user to select the points himself. The performance of this plugin highly depends on the user precision. If he makes a good tracking, it will work well but this is not guaranteed. *TrackFFT* applies a Fourier transformation. It works well on cylinders but not on discs. There are two reasons for that. The first is due to the etching of the sound during the recording that pushed the inside of the groove on the borders. This effect can be seen in figure 2.18.

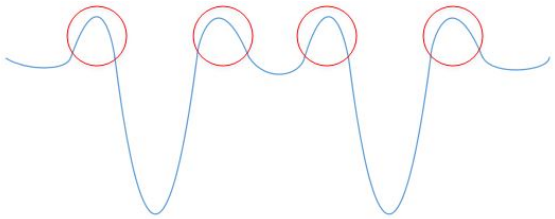


Figure 2.18: Etching of the sound using a lateral cut system

Fourier transformation will therefore find many peaks. The second reason is that the Fourier transformation works well if the brightness differences are higher. That is not true in this case where everything is very near. For more details about this tracking algorithm, please see Romain Crausaz's thesis [2].

The last tracking that is available is a tracking that works by using the depth: trackDepth2. This algorithm works as follows. It starts by finding the minimum depth on the first line of the image (figure 2.20). As seen in the image, it searches in a wide range because it may be that there are no grooves on the left or on the right of the image[1]. Once the minimum found, it saves the point(figure 2.21) and decrements the Y value. After that, it can search for the next minimum nearby. In order to understand why it works, one has to understand the following feature of the groove.

[1]This would mean that the precitec started too early to take the pictures.

Groove Nature: The stylus is always playing the record from left to right and from top to bottom. That is only the case in the picture. In fact, there is only one groove that is being followed by the stylus. Because the process is done on images, some features are added. An important feature is that instead of a groove, there are many side-by-side grooves visible in the images. On the following figure, five distinct grooves can be seen (the blues lines). The dashed lines mean that the image does not stop at these grooves. By looking at the coordinates of a point at the top of the groove or at bottom of the groove, it is possible to see an interesting feature. The coordinate x of the point **B** is identical to the coordinate x of the point **A**. Only the coordinate Y has changed from 0 to the maximal value of Y (height of the image).

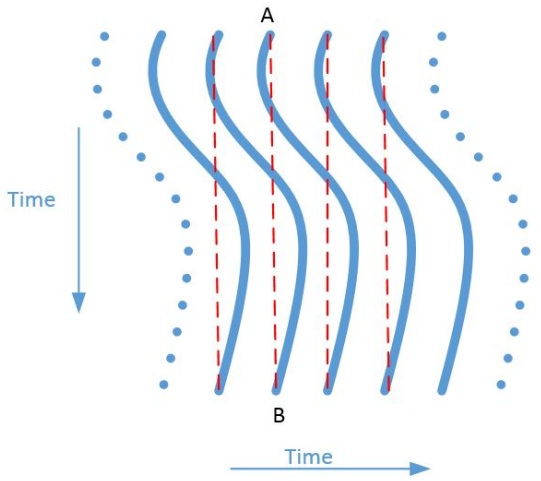


Figure 2.19: Nature of the groove

After that, the algorithm follows the groove to the top of the image (figure 2.22) by finding each next local minimum. At the top, it decrements again Y and so goes on until it reaches the left margin (figure 2.23). At this moment, the left part is tracked but not the right part. To do that, the algorithm starts at the first point and follows the groove to the bottom of the image (figure 2.24). Arrived there, it changes Y with zero and searches the next local minimum (same as before). And so goes on until it reaches the margin right of the image(figure 2.25). At this point the algorithm has completed the tracking.

An illustration of the algorithm can be found on the next page.

Note: Usually, the tracking is applied on a low-resolution image because one hopes that the blobs don't screw the tracking. By changing the resolution, the averaging can help to find the correct deepest point. Furthermore, the tracking will be faster because the algorithm is in $O(n^2)$.

Input parameters: This algorithm takes some parameters:

- Max Change: Change the sensitivity of the algorithm. If it is too high, it might be that the algorithm jumps from one groove to another.
- Left Start: Left margin, where does the algorithm stop when it follows the groove on the left?
- Right margin: Where does the algorithm stop when it follows the groove on the right?

- Min Spacing: Used when the algorithm searches the local minimum. It is used to minimize the range where it can search.
- Groove Width: It is only useful when the user wants to play a stereo audio file.

Output: This algorithm inserts the points in the following list.

```
public List<List<double>> track = new List<List<double>>();
```

This list is sorted from the first point on the left of the image to the last point on the right of the track. Each entry contains the following elements in this order:

- Position Y
- Position X(1)
- Position X(2) - this is used for the stereo mode

The difference between X(1) and X(2) is the size of the input parameter **grooveWidth**.

As previously mentioned, the following images may help to understand the process of the tracking depth algorithm:

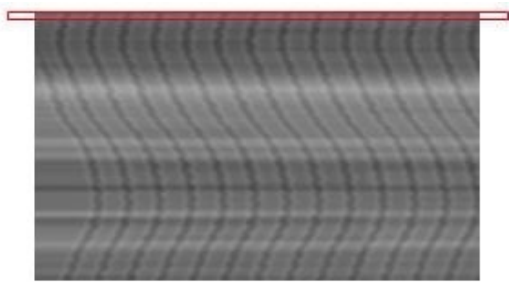


Figure 2.20: 1) Find minimum depth in the first line

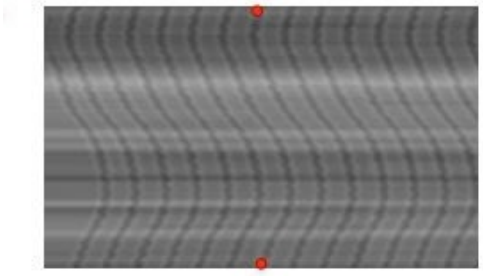


Figure 2.21: 2) Subtract 1 to Y and find the next minimum

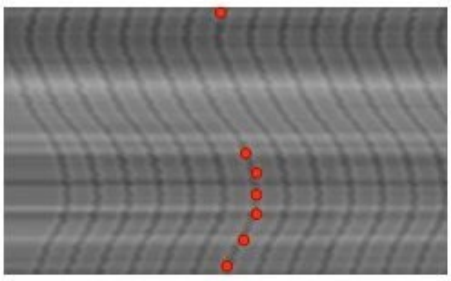


Figure 2.22: 3) Follow the path to the top of the groove

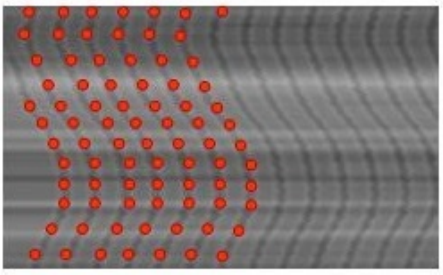


Figure 2.23: 4) Stop the tracking at the left margin

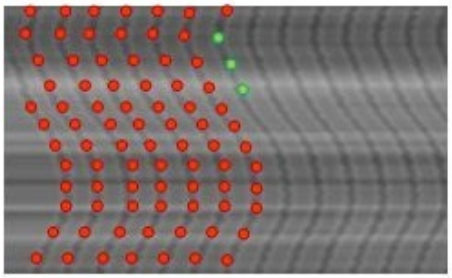


Figure 2.24: 5) Follow the groove to the right part of the image

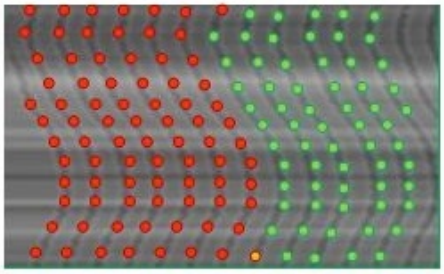


Figure 2.25: 6) Stop the tracking at the right margin. Tracking is done

It is important to understand this algorithm because our contribution requires to follow the tracking.

2.2.4 Non-centered tracking phenomena

The tracking is usually applied on a low-resolution image. In order to follow the tracking points on the original image, the points have to be translated into the original image. This can be done by multiplying X and Y by their corresponding bins. The depth tracking works, but some points can be found on the border of the groove. This phenomenon can be seen on the figure 2.26.

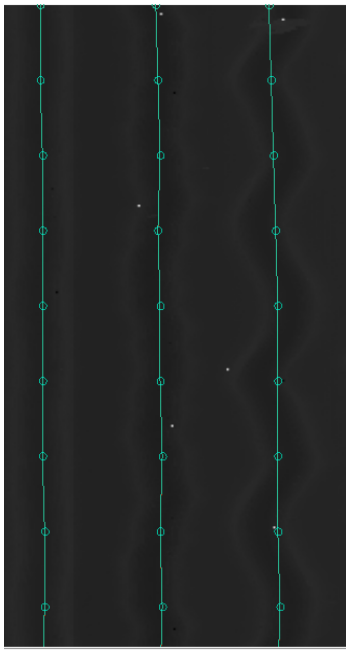


Figure 2.26: Non-centered tracking

The tracking points on the first groove and on the second groove are well placed but that is not the case on the third groove. Even wors, one point is on the left side and the other point is on the right and so goes on with the other points.

Solution: The tracking is applied on a low-resolution image, it could mean that the real deepest point cannot be seen on this image. To avoid that, it is also possible to make the tracking on the original image. It might take more time and would be more sensitive to blobs. This could be a solution, but there is no guarantee.

2.2.5 Structure of the software

IRENE is a plugin-based software as it can be seen in the section 2.2.1. Each feature is a plugin. The project contains the main file **IRENE_Main.cs**. This file describes the main interface and instantiates all the elements. The folder **plugin** contains all the different plugins.

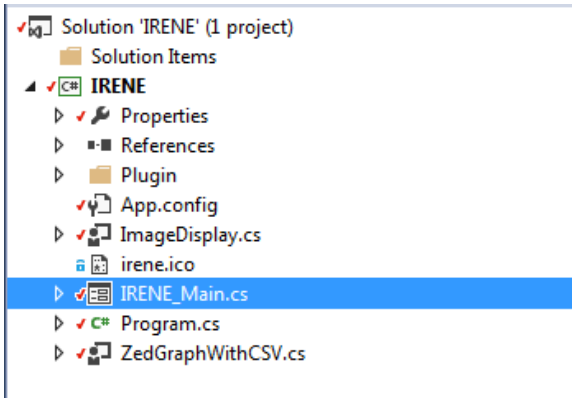


Figure 2.27: Structure of the code

In this main file, the plugins are listed and inserted in the list that can be seen in figure 2.5 (Under "Select Plugin"). It also saves the list of the plugins that are currently present on the user interface in a file *last_plg.txt*. At the start of the program, the main file loads all the plugins present in this file.

2.2.6 Creation of a plugin

It is easy to create a plugin. The plugin must inherit from *PluginAbstract*. The constructor has to call at least the same methods as those that can be seen in the following code:

```
public partial class <NameOfThePlugin> : PluginAbstract
{
    public <NameOfThePlugin>()
    {
        InitializeComponent();
        setName("Name that will be displayed on the
            GUI",<Plugin_Type>, <Plugin_link>);

        // other initialization
        ...
    }
}
```

Listing 2.1: Overview of the plugin class

Then, the class must override the method Run from PluginAbstract:

```
public override void Run()
{
    // Put the code of your plugin at this place
}
```

Listing 2.2: Location of the code

At this point, the real coding can begin but one must know the relations and the properties of the class *PluginAbstract*. First, it is important to understand that the user can always choose if the plugin takes the input from the previous plugin or from another one. Some

plugins may require two inputs and the user will have to choose the plugin input for the two inputs (see ComboBox on figure 2.28).

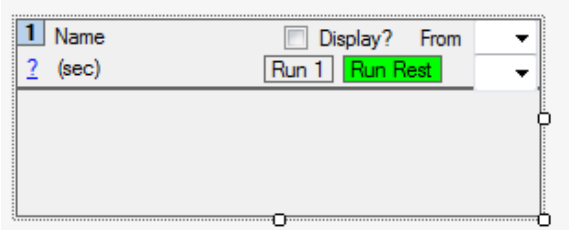


Figure 2.28: Interface of the plugin abstract without additional information

2.2.7 Presentation of attributes and methods

Regarding the relations and the features, these can be seen in the following class diagram:

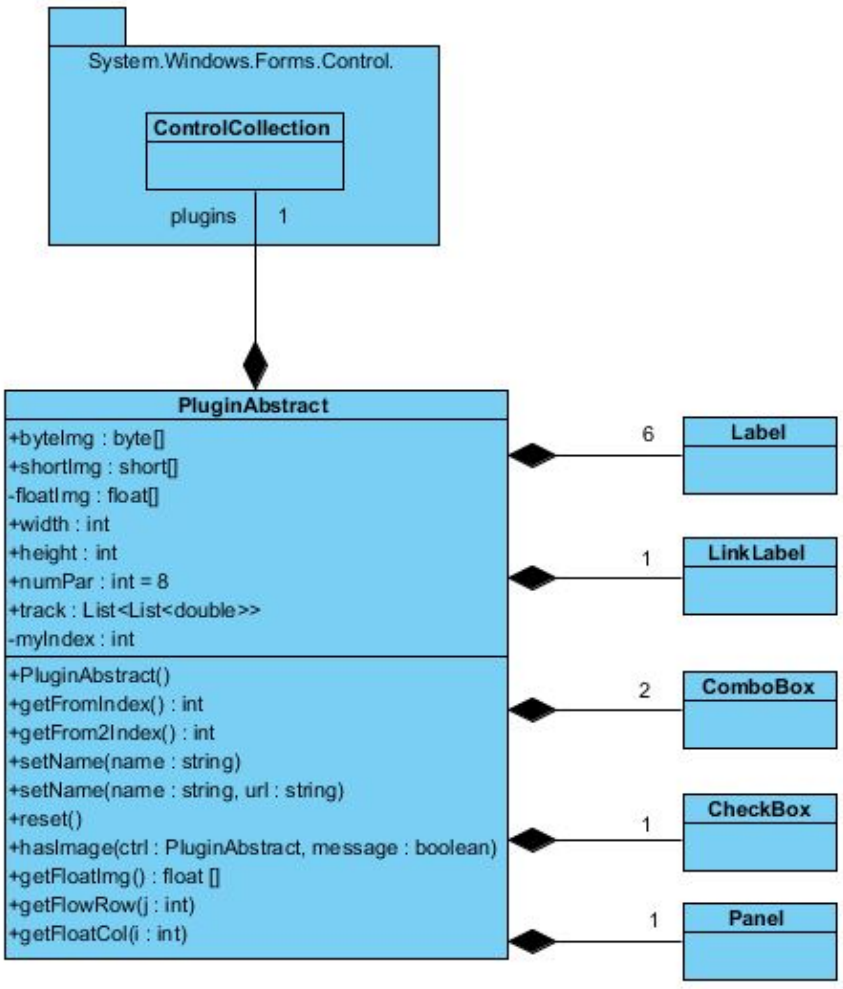


Figure 2.29: Class diagram showing the properties and the relations of the class Plugin Abstract

LinkLabel, Panel, CheckBox, Label, ComboBox are the view elements that are present in every plugin.

Note: This class diagram is not exhaustive. Some attributes and methods are missing. Only those that are interesting and useful for this project have been presented here.

Arrays containing the tiff information: The plugin has three arrays (byteImg, shortImage, floatImg) that contain the value at each point of the image in the different types. Some operations may be faster executed with a byte array than with a float array. Although the image is in 2 dimensions, the table is one dimension. The rows follow each other in the table. This can be seen in the next figure:

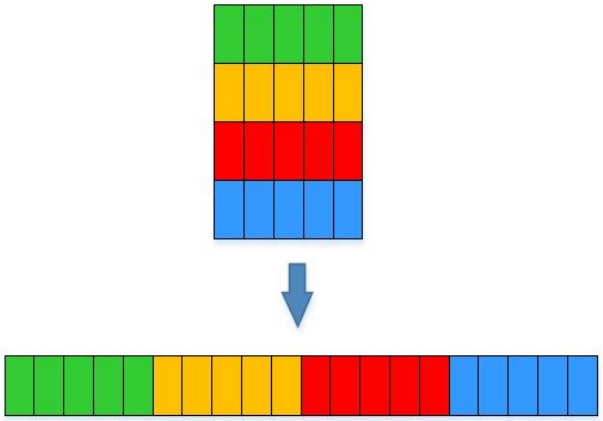


Figure 2.30: Transformation from a 2D array into a 1D array

That means that the index of each cell has to be computed each time this cell needs to be read or update.

ControlCollection plugins ControlCollection allows to maintain a list of UserControl elements [3]. Indeed, *PluginAbstract* inherits from UserControl. Each plugin has a reference to this list. It allows to retrieve the images from each plugin.

Presentation of the other attributes *Width and height* are related to the image and are updated automatically when the plugin is launched by taking the values of the input image of the first plugin.
NumPar is used when a plugin uses threads to compute the result. It corresponds to the number of threads.
Track contains a list of all the points resulting from a tracking.
myIndex represents the index of the plugin in the plugins array (ControlCollection plugins).

Presentation of the methods The methods of the class diagram are just a few of the whole list of methods available in the PluginAbstract class. The *setName()* is used to set the label, the type of the plugin as well as the link to the online documentation. This method has to be called early in the plugin, typically in the constructor.

reset() initializes the array with **NULL** and creates a new list for the attribute *track*.

getFloatImg() returns a one-dimensional array containing the value at each point of the image. This method copies the original float array into a new one that can be modified.

getFloatCol() returns an array with all the values from a specific column. It is very useful when values need to be read/compared in the same column. However, it has to be used carefully because the array is optimized to access the rows but not the columns.

getFloatRow() returns an array with all the values from a specific row. It is very useful when values need to be read/compared in the same row.

getFromIndex() and *getFrom2Index()* are the most important methods in this class. They are used in each plugin in order to get the correct image from the input parameter. The input of these functions are a number that represents the index of the plugin in the list of existing plugins. Reminder: each plugin receives an image from a previous plugin. Some plugins need two entries from two different plugins. An example is the QuadAlu plugin that requires the original image and the tracking.

Each plugin has a unique index attribute that represents its position in the plugins array. This array contains a reference to all the plugins objects. By calling the method getFromIndex, a reference **ctl** is made pointing at the plugin that has been given as the first parameter. It is possible to call the different PluginAbstract methods on the **ctl** object. The second method does the same but allows to use the reference **ctl2**. This is used to get the information from the second input plugin.

2.3 Noise detection - Universal Shape Solution

The universal shape solution is an idea imagined by Carl Haber that consists of improving the noise detection. This section presents the meaning of a universal shape by a step-by-step process.

2.3.1 Stage number one: Creation of the Universal shape

A universal shape is a shape that represents the groove or a part. This is an average of how one would see the groove if the stylus would have cut the image uniformly and without defects. This one can be computed on the whole disc or locally.

The shape of the groove is not always the same for each disc. A lot of different shapes exist. Three of them can be seen in the following image:

Figure 2.31: Different universal shapes

It is not a problem but it means that each disc or disc collection is different. It can also be that the shape is slightly different at each point of the disc.

2.3.1.1 What does it need?

Basically, it requires an image and a complete tracking. A complete tracking is a tracking that contains the point for each Y value. This is particularly important if the tracking has been realized on a binned image. It means that this tracking has to fit on the input image.

2.3.1.2 Fit the tracking points on the original image

As mentioned in section 2.2.3, the tracking is usually applied on a low-resolution image. Reminder: the low-resolution image is produced by binning the image in X and Y. If a binning of 40 in Y is applied, it means that 40 points on a column will be averaged into one single point. It might be that a binning in X has also been applied. However, this highly

unlikely because the motion of the groove would be greatly affected by this. Therefore, it means that a new plugin must be implemented that will fit the tracking on the input image. This plugin would have to take care the fact that the tracking has to be adjusted in Y but also in X.

Each plugin has the information about the width and the height of the image (see 2.2.7). In this case, the input image will have a different width and height than the input tracking width and height. The following image shows the idea of this plugin:

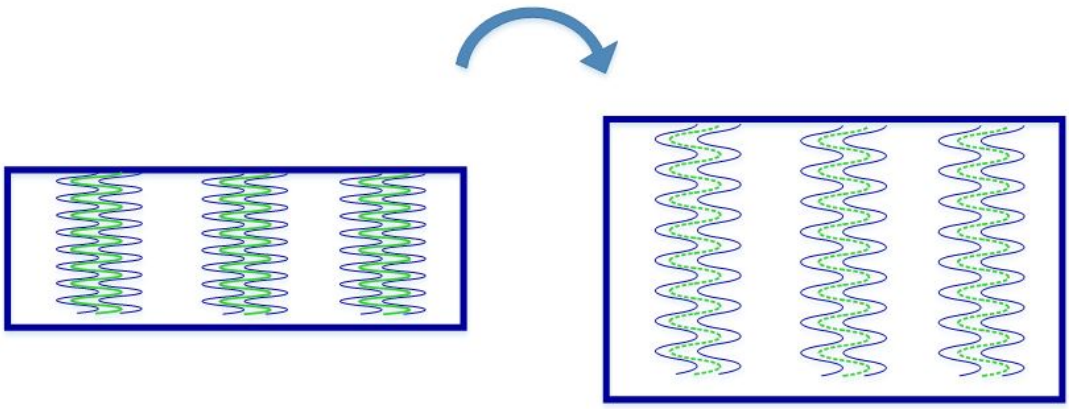


Figure 2.32: Adjust the Tracking on the original image

The previous example shows an image that has been binned in Y by two(left) and the original image (right). On the binned image, it is possible to see that the green tracking is continuous. That is not the case in the original image once the adjustment applied. This is normal since the height is 2 times greater and it would therefore require 2 times more points to have a continuous tracking.

2.3.1.3 Continuous tracking

We are not done yet, since the tracking is not complete. Hopefully, there are at least two possibilities to have a continuous tracking:

- Tracking on the original image
- Interpolation

A first possibility could be to track the groove on the original image but that takes time and can be badly influenced by the noise. An interpolation is faster and might give better results. Different interpolations exist and two of them will be presented in this section.

Linear Interpolation: A linear interpolation is an interpolation where the missing points are computed using a linear function. The following linear system has to be true for each pair of two neighbors points.

$$a * x1 + b = y1 \tag{2.6}$$

$$a * x2 + b = y2 \tag{2.7}$$

The first point is (x1,y1) and the second point is (x2,y2). **a** and **b** can be found using the previous linear equation system:

$$a = \frac{(y2 - y1)}{(x2 - x1)} \tag{2.8}$$

$$b = y1 - a * x1 \tag{2.9}$$

The following example shows how a linear interpolation works on points:

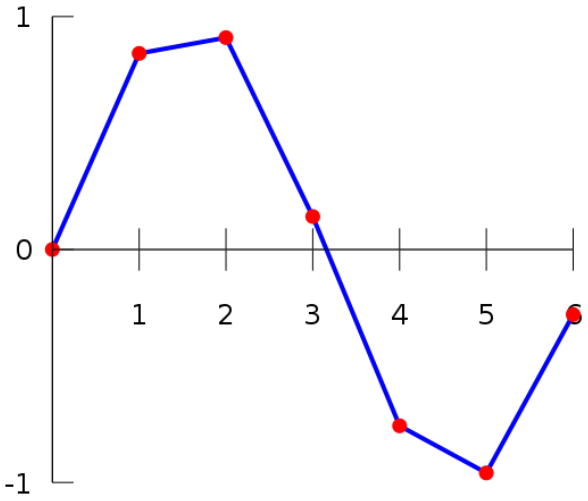


Figure 2.33: Linear Interpolation on random points (image-source: [11])

On the previous example, 6 linear interpolations have been computed. One between two neighbors points.
The linear interpolation can be very useful when the points are close and when a higher precision is not required.

Polynomial interpolation: The problem of linear interpolation is that it rarely represents the function perfectly. The goal of the polynomial interpolation is to use all the points in a one single interpolation where each point has to lie on the function. In this interpolation, the highest exponent of the interpolation has to be decided. Usually, the highest exponent corresponds to the number of points in the list minus one . For instance, if there are 5 points, the highest exponent will be 4 and the function will look as follows:

$$a * x^4 + b * x^3 + c * x^2 + d * x + e = y \tag{2.10}$$

The five points have to fit in this function. To find **a,b,c,d,e**, there are many ways to do that but most work by using a matrix. The following image shows a polynomial interpolation of degree 6:

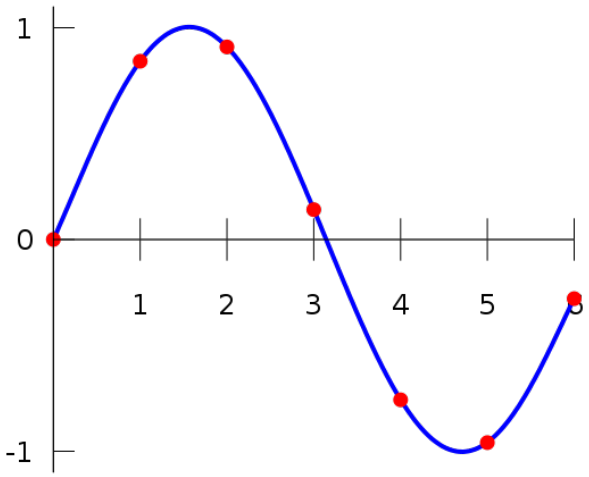


Figure 2.34: Polynomial interpolation (image-source: [15])

Polynomial interpolation works well with a small amount of point but has severe issues with a higher number of points. If the interpolation uses high degree polynomials, it creates a phenomenon called Runge's phenomenon. This can be seen in the following figure:

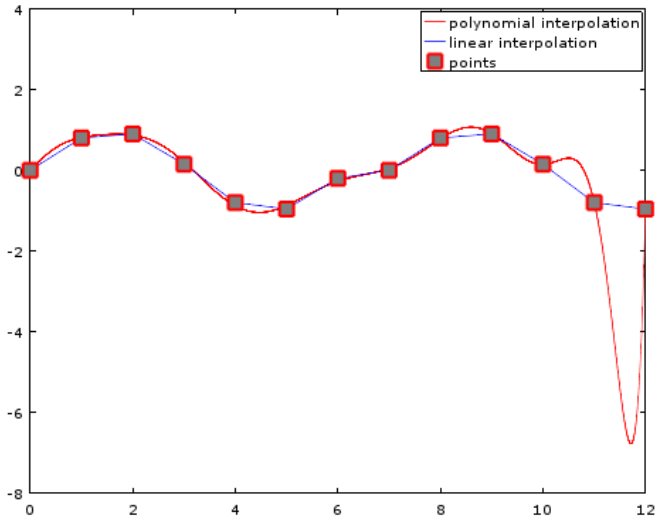


Figure 2.35: Runge's phenomenon

The issue is that if the number of points is high as well as the function's degree, a problem of oscillation will appear on the border of the function (see [16] for more details).

In this case where the number of points could be very high, the Runge's phenomenon has to be avoided. It has to be avoided because the idea is to fill the points missing in the tracking. Those points cannot be too far away from the tracking points. For this reason, polynomial interpolation is not a good solution.

Spline interpolation: Spline is an interpolation that solves the Runge's phenomenon. It creates a piece-wise polynomial interpolation (see [17] for more details).

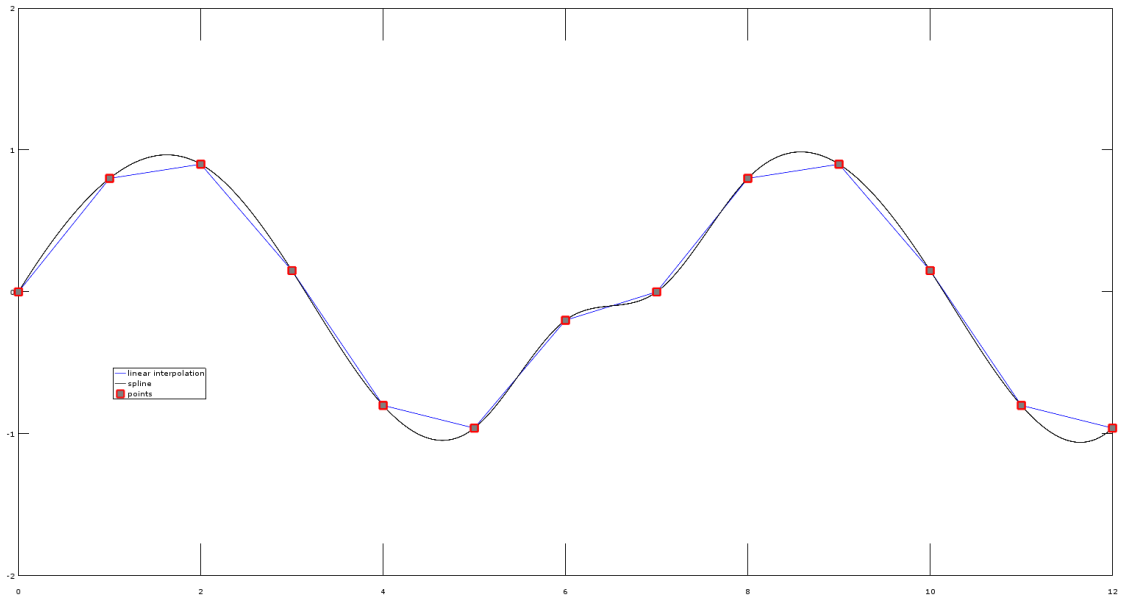


Figure 2.36: Spline Interpolation (image-source: [17])

The linear interpolation and the spline have been implemented.

2.3.1.4 Computing of the universal shape

The universal shape will be computed by following the groove and by saving the value of the shape at each point. A shape is defined by a middle point (the tracking point) and by a width. The width of the shape defines how big will be the universal shape. It can be that there is only one universal shape for the disc but it can also be there are many different universal shapes for different parts of the disc. The design phase will answer this question.

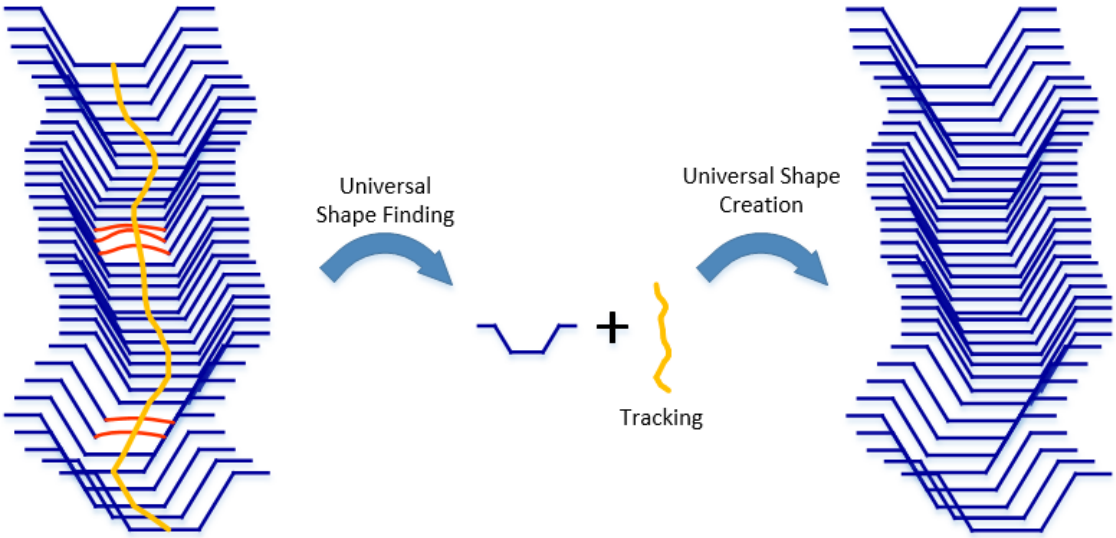


Figure 2.37: Creation of the universal shape

The previous example shows a tracking (on the left in orange) found for a part of the groove. At each point of the tracking, the shape has been saved and a universal shape has been created (in the middle). It results by following the tracking again and by creating a

new image on which the universal shape will be copied at each tracking point. One might hope that the result will be an image containing no errors, no scratches, only a perfect groove.
There are two ways to calculate the universal shape during the follow-up:

- Average
- Median

The average is easier to calculate, but is sensitive to extreme values. The median is more difficult to compute because it consists of sorting the values and by taking the middle point. The complexity of the median is higher(O(n log(n)) than the complexity of the averaging(O(n)).

Between the grooves? The image is not ready yet. For the moment, there is no data between the grooves. To solve that, there are three possibilities.

1. Interpolation: use a linear interpolation between the end of a groove and the start of the next groove. Allow to find the blobs between the grooves.
2. Put the value at zero: not really interesting because the next step(subtraction) will only remove a few things.
3. Use the original value: can be useful if the blob detection doesn't need to be applied between the grooves. Indeed, the next step (subtraction) will change the values between the grooves to zero.

The next image displays an example for each of these possibilities. The third one is an example of what would be the original value.

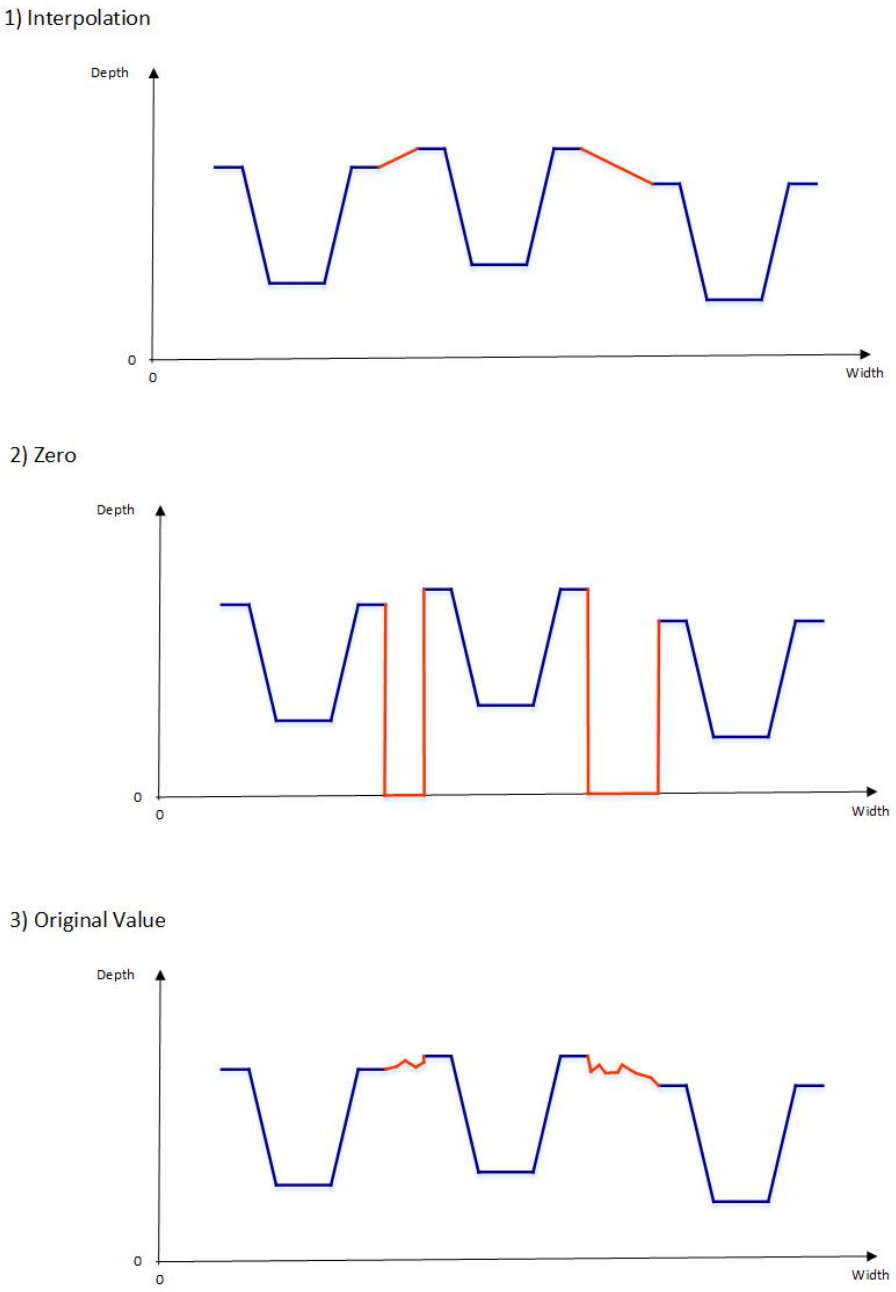


Figure 2.38: Three possibilities to fit between the grooves

This entire process is now finished and the result is an image containing a groove that will be subtracted from the original image. The image that will be subtracted from the original image is called "flat image" and the result will be a "sub-flat image".

2.3.2 Stage number two: Subtraction of the image to the original image and noise detection

At first, one must understand what a flat image is. A flat image is a kind-of perfect image. This image is very similar to the original image but won't have any irregularities, scratches, white points... These elements are called blobs. This image will be subtracted from the original image and will create the sub-flat image. By this way, the blobs will remain and can be treated.

There is a small detail to take care in this idea: some original data will also stay in the subtraction's result. Indeed, the original image, that contains noise, has helped to create this flat image. It means that it is impossible to create a flat image that will allow the subtraction to erase the data and not the noise. It is also important to understand that the purpose of this concept is detect the dirty parts. At the end, the sound will be identified by using the valid data points.

The figure 2.39 displays the result of subtracting the original image with the sub-flat image.

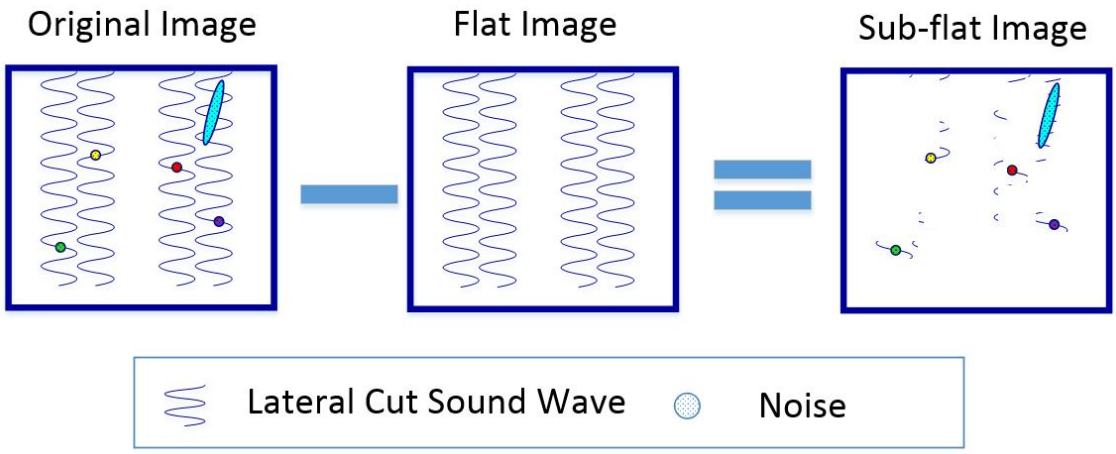


Figure 2.39: Example of a sub-flat utilization

The resulting image contains the blobs as well as some remaining data information. Ideally, the remaining good data should be as small as possible and the remaining noise to be as wide as possible.

2.3.3 Stage number three: Finding of the noise

Once the sub-flat image created, the noise needs to be detected. By taking the example of figure 2.39, the idea is to find the four points(yellow, red, green and blue) and the big blue scratch.

The colors are gone but it does not matter anymore. The entire point of the blob detection is to find the noise and only the noise. The figure 2.40 display what the result of the process will be. An image that have labeled the location of every blob.

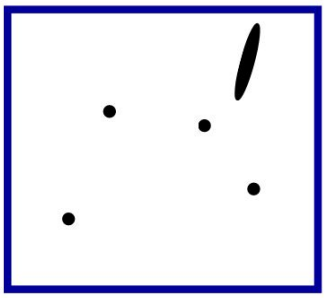


Figure 2.40: Blob detection concept

For the noise detection, the plugin "ImageThreshold" will help.

2.4 Polynomial fitting solutions

The section 2.3 mentioned that the result would be an image that has labeled the blobs. The idea is to follow the tracking again and apply a fitting at each point. This fitting can be minimized in order to find a new point. This new point will be a better representation of the audio that the tracking that has been applied during the blob detection process. To achieve that, four solutions will be analyzed, implemented and tested.

The four solutions are the following ones:

- Polynomial fitting over a single line
- Iterative polynomial fitting over a single line with outlier
- Polynomial fitting over two lines with outlier
- 3D polynomial fitting - over many lines

2.4.1 What does it need

This process will need three elements. The first thing is an image on which the different polynomial solutions can be applied. This image does not have to be the original image but must be of the same size. It also needs a tracking. This tracking does not need to be a complete tracking, but this tracking will have the same size. The reason is that the polynomial solutions will be applied over the points present in the tracking list. The last thing that these solutions needs is a blob detection image. This image will contain a label for each blob. The location of the blobs will always be ignored during the process.

2.4.2 Shape of the groove

To understand the four solutions, one might know how does the shape look like. This can be seen in figure 2.41

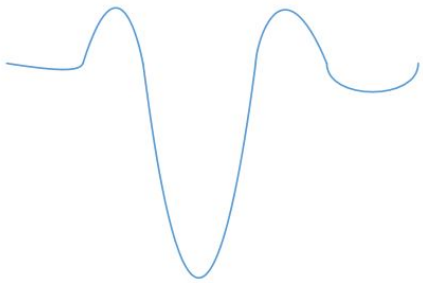


Figure 2.41: Shape of the groove on aluminum discs

The shape looks like because the etching of the sound on the aluminum disc had the effect to push the inside of the groove on the borders. In the previous pages, the shape of the groove always looked like a trapezoidal shape. It was not false, one some discs, it does have a trapezoidal shape. This is not the case for this project.

Moreover, the shape of the groove is polynomial. It means that a polynomial fitting might be a good idea.

2.4.3 Polynomial fitting idea

This first idea is to follow the tracking again and to make a polynomial fit with the points of the shape. The blob points will be ignored in the process during the fitting. Once done, the minimum point of the shape will be a better representation of the sound than the current tracking point.

2.4.3.1 Compute the fitting

By following the tracking, the idea will be to use the shape of this point by knowing the width of the groove and to create a polynomial function with these values. This can be seen on the left side of the figure 2.42. It is normal that not all the points fit on the function but it is not a problem for the moment. After that, the minimum of this function can be computed and will be saved in the tracking. On the right side of the figure 2.42, the new minimum can be seen as well as the old minimum.
Note: the points have been taken randomly.

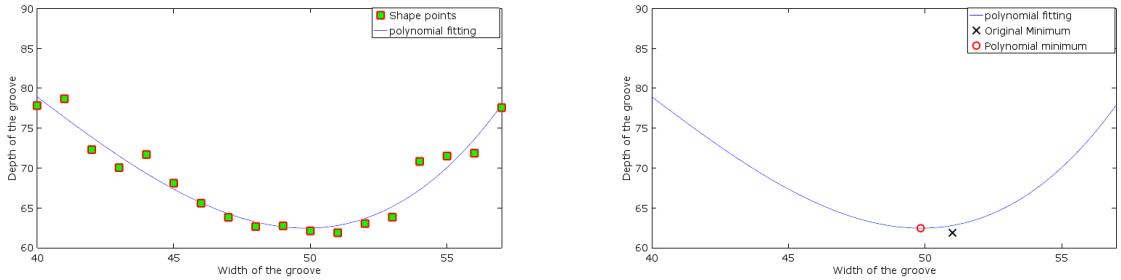


Figure 2.42: Example of a shape on which the polynomial fitting has been applied

If this process is done on the whole disc, the new tracking might be better than the other tracking.

Computing of the minimum In order to find the minimum of the polynomial function, it is possible to use the following formula:

$$x_{minimum} = \frac{-b}{2 * a} \tag{2.11}$$

This works if the function is a function of the third degree:

$$a * x^2 + b * x + c$$

Noise on the shape: It might be that some points have been labeled as a blob (see 2.2.2.4 for more details). When the solution take the shape, it will ignore some points. The left side of the figure 2.43 shows the same groove as before but on which the left part has been labeled as blobs. It means that these points will not be used for the fitting. This can be seen on the right side of the figure 2.43 where the red curve represents the shape without the blobs.

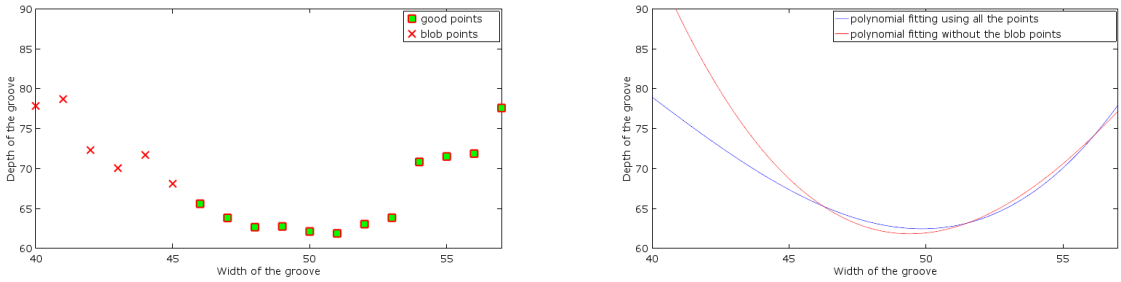


Figure 2.43: Same shape, but contains noise

Anyway, the goal is to find the minimum of this function in order to create a new tracking where the tracking point is more accurate than what a tracking plugin can do.

2.4.3.2 What does it need?

It will require an image, one tracking and a blob image. The image will be the original image or another on which the polynomial fitting will be done. The solution will follow the tracking and at each point of the tracking save the shape of the groove and compute the polynomial function that represents the function the best. It will ignore the points that are labeled as blobs in the blob image.

Input parameter This plugin will also need a way to define the width of the groove so it can make the polynomial fitting. It will also require a parameter that defines how many blobs a shape can have before the program decides not to make the fitting and to ignore this point. For instance, if a width is 20 and there are 5 points labeled as blobs on this shape, it might be good to make a fitting, but if there are 12 points labeled as blobs, it might not be a good idea to make the fit. This means that some points will be missed at the end of the process. Those needs to be filled because the tracking has to be continuous.

2.4.3.3 Why would it help?

It has already been presented in section 2.4.2 that the original shape of the groove looks like a polynomial shape. Therefore, there are good chances that a polynomial fitting might be a good idea to represent the original shape.

2.4.4 Iterative polynomial fitting with outlier

The idea is to follow the tracking again and to make a polynomial fitting with the points of the shape. The first thing is to throw away the points that have been labeled as blobs. After that a polynomial fitting will be applied at one point. The problem is when a fitting is made, it can be that some points are really far apart from the solution. The idea is to find which point is the furthest from the function. The algorithm will throw it away and make a new fitting with the other remaining points. It checks again if one point is too far away and makes a fitting away. This will be made during many iterations and at each point of the tracking. At the end of the iterations, the minimum of the function will be computed and used for the final tracking. The entire process can be seen in figure 2.44.

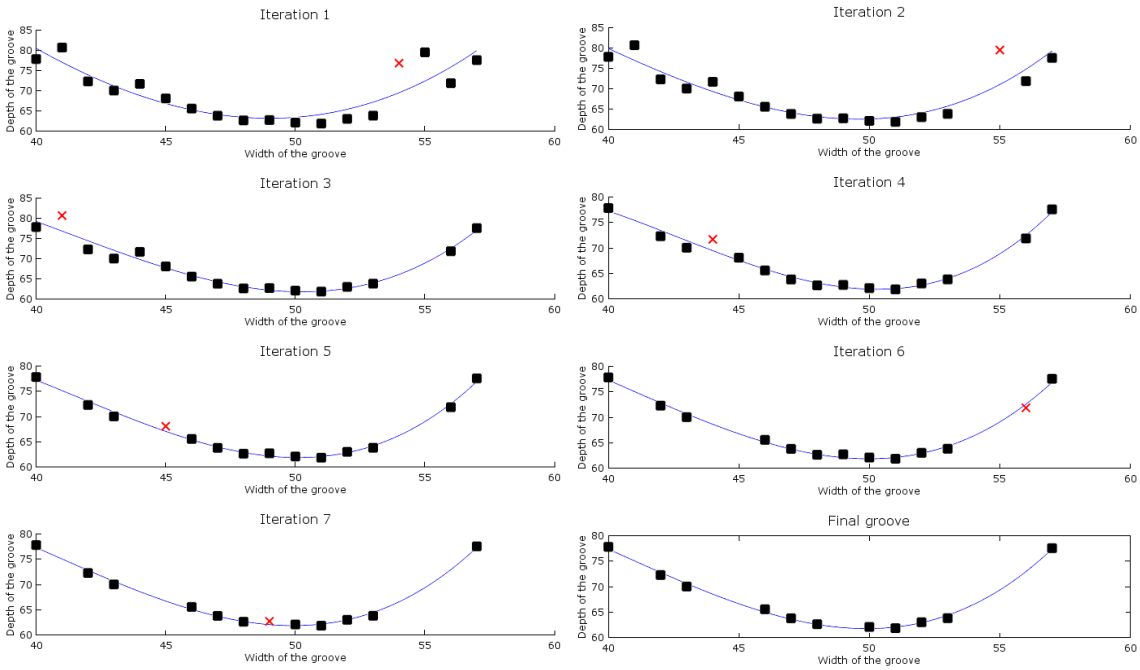


Figure 2.44: Iterative polynomial fitting by throwing away the farthest point

The x-axis represents the width of the shape and the y-axis represents the depth of the shape. A red-cross can be seen on the first 7 shapes. This red cross represent the point that is the farthest away from the polynomial curve. The next step, this point is removed and the polynomial fitting is applied on the remaining points. The figure 2.44 represents seven iterations on which the farthest point has been removed. It can be seen that, it does not change a lot

It is clear that this can goes on indefinitely until no points are left on the shape. This is not the purpose of this solution. It means that this version needs to have some parameters.

2.4.4.1 What does it need?

It will need two parameters. One point that specify how far away a point can be from the fitting. This introduces a certain error tolerance. In the previous example, the iterations five and six might not be useful. It can be seen that it does not make a big difference if there are thrown away or not. Even worst, it can be that too many points are thrown away and that the fitting becomes worse. Therefore, a parameter is required in order to avoid a degradation of the results. The goal is to improve the results not to degrade the results.

Another parameter will specifies the number of modifications allowed during the process. If the number of allowed iterations is five, the polynomial fitting that will be used in the last one that has been modified even if there are still points that not close enough. This parameters can also be a number that says how many points can we at least at the end of the iterations.

2.4.4.2 Why would it help?

This solution should help to improve the fidelity of the polynomial function with respect to the original groove. The groove that has been etched during the recording of the sound. Alain Benninger in his bachelor project in 2013 also tried this approach and seemed to

be successful (see Bachelor thesis of Alain Benninger [1]). However, his idea was slightly different than this one. Instead of removing points during the iterations, it replaced the furthest point with the point of the function.

2.4.5 Polynomial fitting by correlation (second outlier version)

The first two solutions make a polynomial fitting over one line at the time. The first is very basic and the second deals with outlier notion. This time, the notion of outlier will be slightly different. There will be no iterations but a subtraction between two lines. The idea is to know how two lines are different from each other. Indeed, two lines of a groove cannot be too different because a groove is continuous, there must be a correlation between two consecutive lines. It means that if the difference between two consecutive points is too high, something must be wrong. This subtraction is presented in figure 2.45:

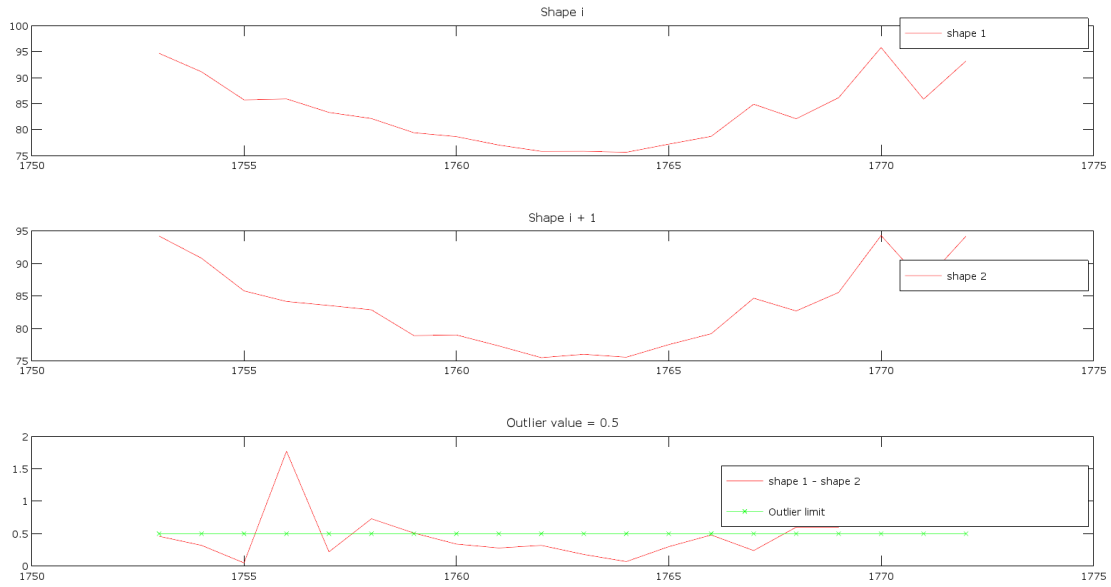


Figure 2.45: Polynomial fitting by correlation (second outlier version)

In figure 2.45, the points that are above the green line will be ignored. The blob points will also be ignored. At the end of the process, the fitting will be made by ignoring the points that have been labeled as blobs as well as the points where the difference between the two lines is too large. At the end, a polynomial fitting will be applied by using the remaining points on the first line.

2.4.5.1 What does it need?

This solution would need one parameter outlier that set the maximal allow difference between two lines.

2.4.5.2 Why would it help?

It will help because it takes into account the fact that two consecutive lines are correlated. It does not make any interpolation or fitting, but it does not blindly look one the current line but also look on the next line. The problem with the two first ideas might be that they are solitary ideas.

2.4.6 3D polynomial fitting

This idea uses also the fact that there are correlations between the lines. This time, a polynomial fitting is applied over many lines. A function with two variables need to be found. The problem is that this cannot satisfy all the values. A minimization has to be done.

The function that needs to be found is the following:

$$f(x,y) = a*x^2 + b*y^2 + c*x + d*y + e*x*y + f \tag{2.12}$$

This is a second-order-function. This has the highest chances to find the best shape. A function of first order would give a surface and a function of highest order would definitely be slower and more difficult to implement.

If the fitting has to be done over five lines containing 20 points each, it would mean that this function should represent 100 points. No function can perfectly suit for the 100 point,s but a minimization of errors can be done. The method of least squares will be used: the square sum of all errors needs to be as close as possible to zero. There is a good explanation on the internet page WolframMathWorld[18] as well as a paper from the Lawrence Berkeley National Laboratory (LBNL) that explains the reasons of using the least squares method.

The equation that need to be minimized is the following:

$$\sum_{i=1}^{n} \frac{(a*x_i^2 + b*y_i^2 + c*x_i + d*y_i + e*x_i*y_i + f - z_i)^2}{\varphi_i^2} = 0 \tag{2.13}$$

φ is a weight that can be applied independently for each point. This lead to minimize the effects of some points. If the weight is big, the impact of this point will be less great.

This function needs to be derived in order to find the minimum. This gives the following function: a derivative has to found for each of the six parameters (a,b,c,d,e,f). The partial derivative for each parameters are presented below:

$$\frac{\partial f}{\partial a} = 2*\sum_{i=1}^{n}(a*x_i^4 + 2*b*x_i^2*y_i^2 + c*x_i^3 + d*x_i^2*y_i + e*x_i^3*y_i + f*x_i^2 - x_i^2*z_i)/\varphi_i^2 = 0 \tag{2.14}$$

$$\frac{\partial f}{\partial b} = 2*\sum_{i=1}^{n}(a*x_i^2*y_i^2 + b*y_i^4 + c*x_i*y_i^2 + 2*d*y_i^3 + e*x_i*y_i^3 + f*y_i^2 - y_i^2*z_i)/\varphi_i^2 = 0 \tag{2.15}$$

$$\frac{\partial f}{\partial c} = 2*\sum_{i=1}^{n}(a*x_i^3 + b*x_i*y_i^2 + c*x_i^2 + d*x_i*y_i + e*x_i^2*y_i + f*x_i - x_i*z_i)/\varphi_i^2 = 0 \tag{2.16}$$

$$\frac{\partial f}{\partial d} = 2*\sum_{i=1}^{n}(a*x_i^2*y_i + b*y_i^3 + c*x_i*y_i + d*y_i^2 + e*x_i*y_i^2 + f*y_i - y_i*z_i)/\varphi_i^2 = 0 \tag{2.17}$$

$$\frac{\partial f}{\partial e} = 2*\sum_{i=1}^{n}(a*x_i^3*y_i+b*x_i*y_i^3+c*x_i^2*y_i+d*x_i*y_i^2+e*x_i^2*y_i^2+f*x_i*y_i-x_i*y_i*z_i)/\varphi_i^2 = 0 \tag{2.18}$$

$$\frac{\partial f}{\partial f} = 2*\sum_{i=1}^{n}(a*x_i^2+b*y_i^2+c*x_i+d*y_i+e*x_i*y_i+f-z_i)/\varphi_i^2 = 0 \tag{2.19}$$

Six equations for 6 unknown: system of linear equations. To solve a system of linear equations, there are many solutions but the most common way is to put them into a matrix system Ax=b that can be solved by Gaussian elimination. This was also done in [18].

Computing of the minimum: Finding the function is good but that does not give us directly a point. The 3D polynomial fitting should also create a fitting. To obtain that, a second minimization has to be done. Before this, it is necessary to recall that the position y of the line is known and that the x is searched. If y is known, that gives a function with only two remaining unknowns x and z. z needs to be minimized. The following formula find the minimum of x:

$$x_{min} = \frac{(-c - e*y_{fixed})}{2*a} \tag{2.20}$$

A piece of code in octave has been written in order to validate this analysis. The result can be seen on figure 2.46:

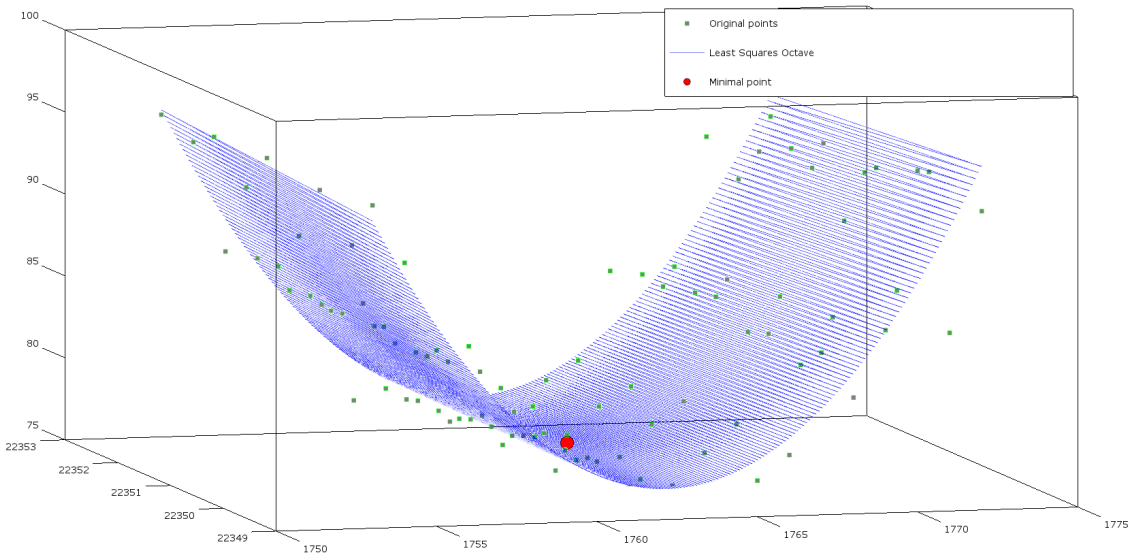


Figure 2.46: 3D Polynomial fitting over five lines

In this example, five lines have been fitted by using the formulas described above. The red point is the point that will represents the tracking. In this solution, the blobs are also ignored during the fitting.

Roll-off in the high frequencies: It might be that if the number of lines is to big that the high frequencies are dropping down. It is important to be careful on the number of lines. This has to be a parameter of the new coming plugin.

2.4.7 Missing points?

It can be that there are too many blob points on a shape and that makes it impossible to make a polynomial fitting. If this is the case, no fitting will be applied at this point and the algorithm will continue to the next tracking point. This means that some points will be missing no matter which of the four solutions is implemented. To solve that these points will be filled with an interpolation such as spline or linear interpolation. For more details about these two interpolations, please refer to section 2.3.1.3.

2.5 Testing of the solution

The main idea is to improve the noise reduction on aluminum discs. What does it mean to improve the solution? Is there a number to compare with?

Often, it is impossible to have an optimal solution and no one can say that the noise reduction is perfect because there is always an improvement potential. However, audio scientists can easily compare two solutions and find the best of them.

Basically, there are two ways to analyze the result of the program. The first one is subjective and consists in listening to the two audio files. Sometimes, a noise can be heard which is not natural to the original audio.

The second is less subjective and consists in analyzing the spectrum of the sound:

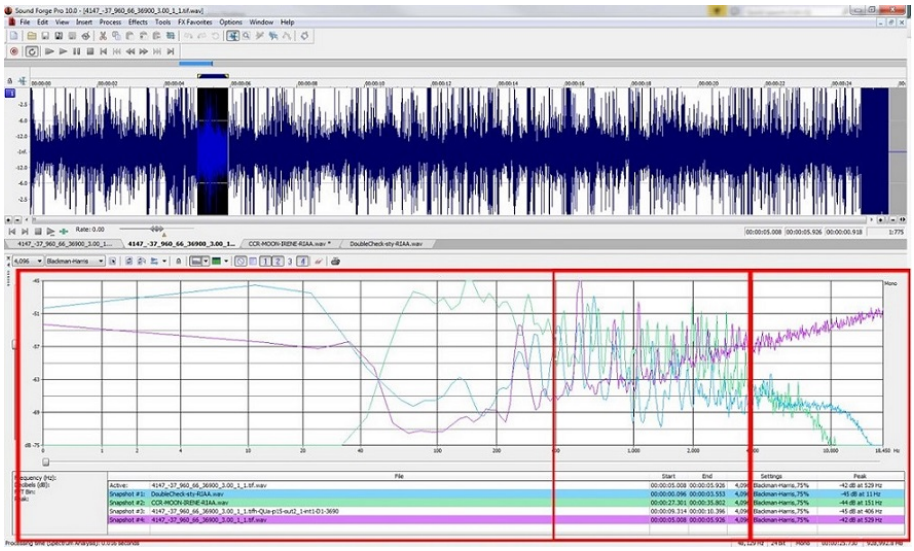


Figure 2.47

The spectrum is a graph that represents the vibrations of the sound over the different frequencies. The vibrations are measured in decibels, and the frequencies are measured in hertz.

In the previous figure(2.47), the blue and the green lines are the spectrum for well-preserved aluminum discs that have been processed with IRENE. These are referencing versions that the project can compare with. It is highly possible that kind of quality won't be reached because the discs available for this project have more noise and scratches. Furthermore, these lines have been measured on a part of the audio where can hear some audio.

The purple line is different. This is the spectrum from a file that has been treated with Irene recently and for which the quality of the disc is lower than the two others. It is possible to see that there is a huge difference in the high frequencies. That means the

hiss is more important than by the two others. Usually, the high frequencies(higher than 10 kHz) are really difficult to hear if not impossible as well as the low frequencies(lower than 500 Hz). In that case, it is also true but they give information: there are too many vibrations.

2.5.1 Hiss

The aluminum groove is not perfect. During the recording of the audio, it was impossible to print straight lines. It has always some defaults due to the nature of the material. The figure 2.48 shows an example where the groove is not perfectly flat.

Figure 2.48: Hiss visualization

The idea is to deal with that and to diminish the bad effects of the hiss. Even if there is no sound, this hiss can be heard. In a silent part of the audio, if the hiss would not exist, the decibels should be very low. This is also a good idea: take the spectrum in a silent zone of the audio file. If there are too many vibrations, it means that there is still an improvement potential.

2.5.2 Root mean square

Having a number to compare two audio files can sometimes be a good idea to quantify the quality of the sound. In Sound Forge, it is possible to open a statistic window. The root mean square (RMS) level that can be seen in the figure 2.49 would be the most interesting number.

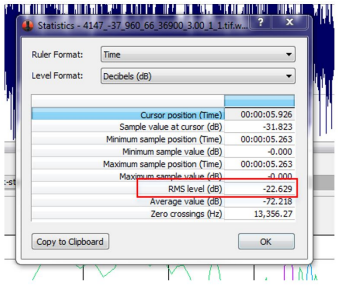


Figure 2.49: Root mean square of the sound

The RMS is, also known as quadratic mean, is the square root of the mean of the squares of the values:

$$rms = \sqrt{\frac{1}{n} * (x_1^2 + x_2^2 + ... + x_n^2)} \tag{2.21}$$

It is a generalization of the mean using the square instead of the value. If the value is higher than another, it means that there are more vibrations in the sound.

2.5.3 Noise visualization on the sound wave

Instead of looking the spectrum, it is also possible to look on the sound wave. On the right side of the figure 2.50, peaks(in red) can be seen. These are not natural. This may

mean that the noise has not been found during the search for noise, or that it is poorly processed. The left side of the figure shows a good sound that does not contain any peaks.

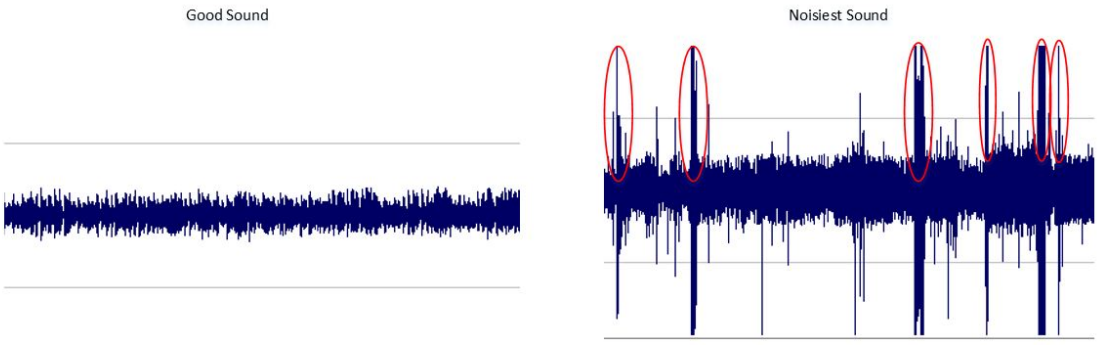


Figure 2.50: Noise Manifestation visible on the sound function

2.5.4 ReaFir

It is not a test but can still help to compare two audio files. As previously mentioned, the very high frequencies can usually not be heard and the high frequencies don't contain any valuable sound. It means that their removal should not change the sound at all. The problem is that the hiss is also visible in the high frequencies and damages the sound. Sound Forge has a plugin that allows the user to diminish or to erase the decibels for specific frequencies. The following figure shows an example where the decibels for frequencies above 5 kHz have been decreased or reduced to the minimum.

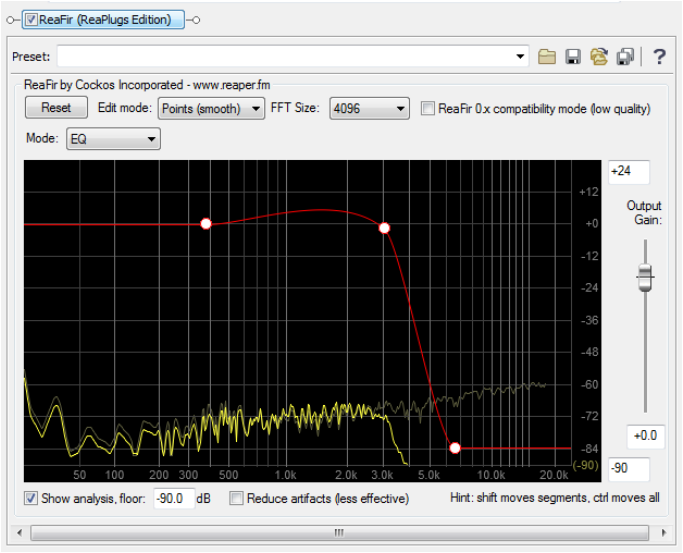


Figure 2.51: Plugin ReaFir in Sound Forge

2.6 Additional contribution

This section will add any additional analysis that had to be done in order to implement the universal shape solution. It means that this part was not specific to the solution but is necessary in order to test the program.

2.6.1 Slope Correction

Sometimes, the tracking fails because the groove is not similar on the entire disc. It can be that there are several differences between the depth at one point of the groove with the depth at another point of the groove. This is due to a focus problem during the recording with the precitec. In order to solve that a plugin needed to be implemented. This section will present the analysis of the slope correction.

In IRENE, it is possible to see the function of the depth on a selected column or line. It is possible that this function looks like a staircase function.

Figure 2.52: Slope correction idea

This phenomenon is repeated for each pass of the file. It means that the recording with the precitec made some mistakes with the focus. To solve this, this plugin requires two parameters:

- **PassWidth**: the width of one pass. This can be seen on the main image of the GUI where black lines split the different passes. It is also defined by the precitec pass width.
- **Slope**: the difference between the lowest point of one pass and the highest point. The figure 2.53 shows how to get this value.

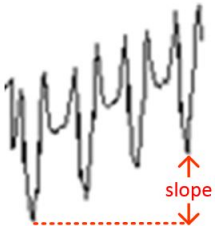


Figure 2.53: Slope parameter meaning

Once done, this plugin can adapt the values at each point with the following formula:

$$newValue = oldValue - slope * \frac{i}{passWidth} \qquad ,i \in [0, PassWith[\tag{2.22}$$

2.6.2 Compute Middle

The idea of this plugin is really simple. It will take a tracking that has two different X values for each tracking point and compute the middle point. This can be seen in the figure 2.54.

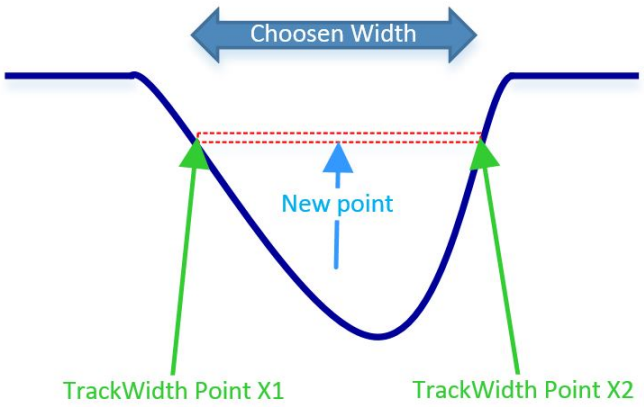


Figure 2.54: Compute Middle Concept

This plugin produces a tracking where the new point is the middle point between the previous values. One might hope that it will be a better middle than what a tracking algorithm can do. Mostly, this plugin will be used in addition to the *TrackWidth* plugin (see section 2.2.2.6 for more details).

2.7 Conclusion

The analysis has been focused on five different parts. The first part consisted of analysis the LabView software because the project consisted of testing the solutions with the new Precitec. Only a few scanning have been made before the project and this analysis made it possible to understand how does a scanning work and to start the scanning in order to have enough data to work with.

The next analysis resided by analyzing the IRENE-plugin software. This plugin-system had a lot of existing plugins that needed to be used. One can definitively not use the plugins without understanding them. This leaded also to understand features of the aluminum discs.

After that, two analyses have been carried out specifically to the objectives. The objective was to detect the noise and to playback the sound via different polynomial fittings. The universal shape solution that will create a flat image has been analyzed. There are different ways to implement this solution. Will the universal shape be global or local? There are good reasons to think that global might not be a good idea. There are a lot of differences between the passes. The design phase will decide what decision to adopt.

Concerning the playback of the sound, four solutions have been analyzed that are somehow related but also very different. They all use the fact that a shape in aluminum discs has a polynomial shape but in different ways. They might give totally different results. It is also impossible to say which algorithm will give the best results. The analysis of these solutions resulted in several mathematical formulas, especially for the 3D polynomial fitting.

In order to implements these two features in the IRENE-plugin-system, additional contribution is required. Some plugins that are not directly related to the solution will be implemented.

The last analysis resided about understanding which tool can be used to validate the solution. The tool that came out is: SoundForge, this powerful software can help to under-

stand and compare the different audio files that will be produced after the implementation. It is also the tool that is used by the members from the IRENE-team.

Chapter 3

Design

The project consists of two distinct parts. The first part consists of detecting the noise on TIFF files. The second part consists of handling the noise and of creating a new tracking that will represent the sound. The idea that has been presented in the analysis phase use the fact that the groove on aluminum discs is polynomials. During the project, four ideas have been analyzed and need to be designed in the current chapter. This chapter answers the following question: How can these ideas be integrated in the form of an algorithm?

3.1 Universal shape solution

The universal shape solution is a plugin that will create a flat image containing the grooves without blobs. These grooves will not exactly correspond to the original groove. In order to find the noise more easily, the idea will be to subtract this image from the original image. That will produce a sub-flat image.

The general concept of this plugin is described by the following activity workflow:

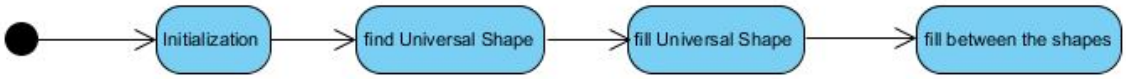


Figure 3.1: General activity diagram of the computing of the universal shape

There will be a part of initialization, followed by the computing of the universal shape. This will be done by following the tracking, by saving the shape at each point and by computing a universal shape at the end of the process. Once done, the universal shape is inserted in a new image (fill universal shape). After this step, this new image will contain the groove but no other value. The space between the groove is empty and that will be filled in the last step: fill between the shapes.

3.1.1 Find universal shapes

In the analysis part, it is mentioned that the universal shape can be made locally or globally. It has been decided that there will be one universal shape per distinguishable groove on the image. This is illustrated in the following figure:

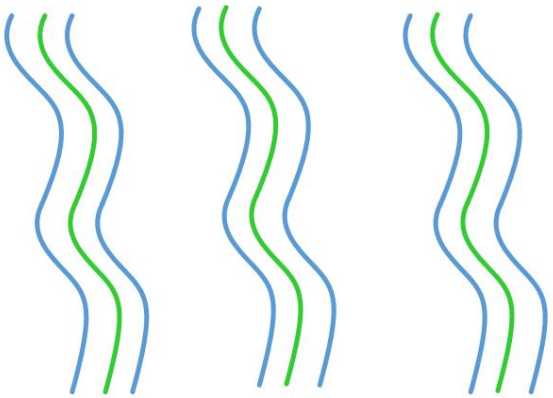


Figure 3.2: Example where three universal shapes will be computed

In the previous example, the image contains three distinguishable grooves. In fact, there is only one groove but the image shows three grooves: one for each rotation of the cylinder.

Furthermore, it has been decided that the universal shape can be computed with the average value or with the median value. That will be the user's choice.

The algorithm that will compute the universal shape is presented in figure 3.3:

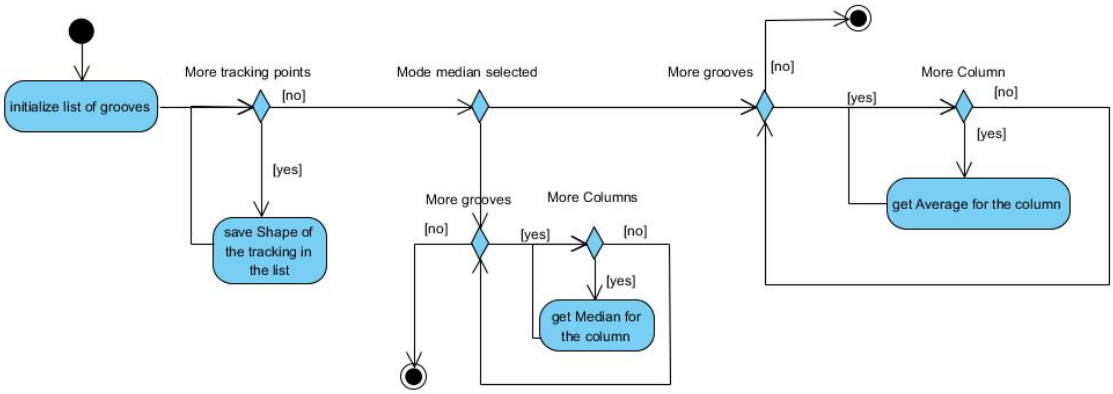


Figure 3.3: Activity diagram describing the algorithm that will find the universal shape

The algorithm will follow the tracking and will save the shape at each point. If the user has selected Median or Average it will be slightly different. If median has been selected, it will loop over the different grooves and over the column in order to get the median value for each column for each groove. At the end 3 shapes of the desired width will be saved in the list.

3.1.2 Fill universal shapes

The universal shapes found previously must be saved in a new image at the same position as the original tracking.

This part of the plugin will follow the tracking again and save the corresponding universal shape at the right place in a new image. The following activity diagram describes the filling process of universal shapes:

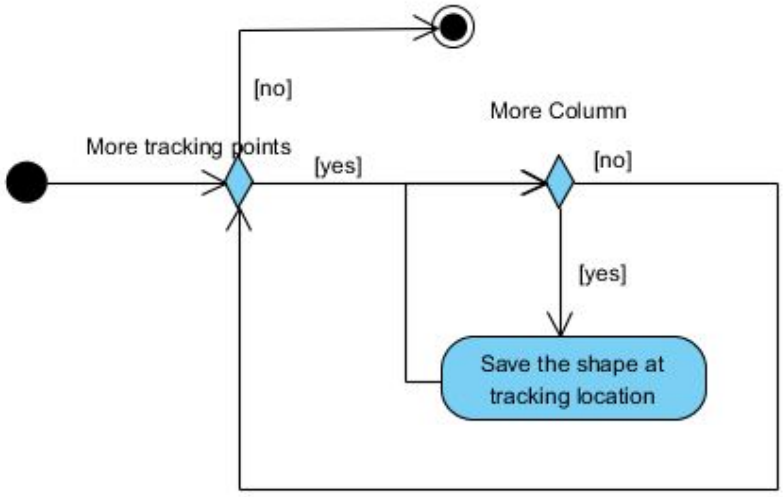


Figure 3.4: Activity diagram describing the algorithm that will copy the universal shape in a new image

3.1.3 Fill between the grooves

Last but not least, the new image has to contain some information between the grooves. This will be done on each line independently of the other lines. At first, the algorithm will try to find the first groove. At this point, it will copy the value of the first point of the groove all to the left until it reaches the left border of the image. After that, it will find the end of the groove and save the point's value and location. The algorithm will try to find the second groove. As soon as it reaches the new groove, a linear interpolation is made between these two points. In other words, it will interpolate between the right side of the first groove and the left side of the second groove.

It will try again to find the end of the groove, it will interpolate with the next groove and so goes on until it reaches the last groove. Similar to the beginning, the value will be copied to the right until it reaches the right border of the image.

This is explained in the following activity diagram:

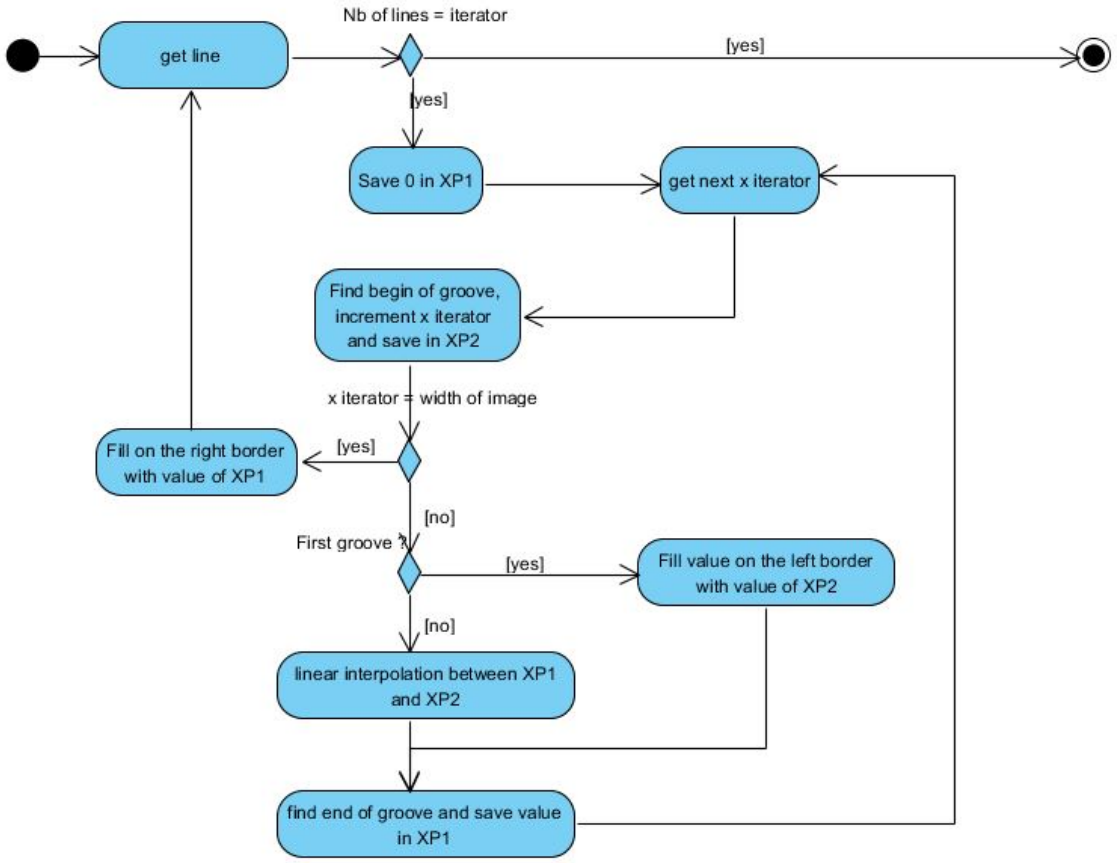


Figure 3.5: General activity diagram describing the algorithm that will fit the depth between the shape previously inserted

3.2 Polynomial fitting solutions

Four polynomials fitting solutions have been analyzed. The use will have to choose between one of these solutions. It might be that the test phase will clearly say the algorithm that should be used. Before that, these solutions need to be designed. These will have the same inputs: an image, a tracking and a blob-image. The design assumes that this condition has been fulfilled. One feature that all the solutions will have is the ability to ignore the noise that is contained in the blob-image. Each solution will follow the tracking and apply a different polynomial fitting on the tracking point. The difference between these versions is explained in the corresponding sections of this chapter.

3.2.1 Polynomial fitting

The plugin that will be implemented will simply apply a polynomial fitting over the shape of the groove at each point of the tracking. The function to compute the polynomial fitting already exist. The algorithm can use it and can use the function described in the analysis to compute the minimum. It might be that the fitting is too bad and that the minimum is outside the range of the shape. This behavior is not acceptable, and the point will be ignored. The point will also be ignored if there are too many noisy points on the shape.

The algorithm is described in the activity diagram on figure 3.6:

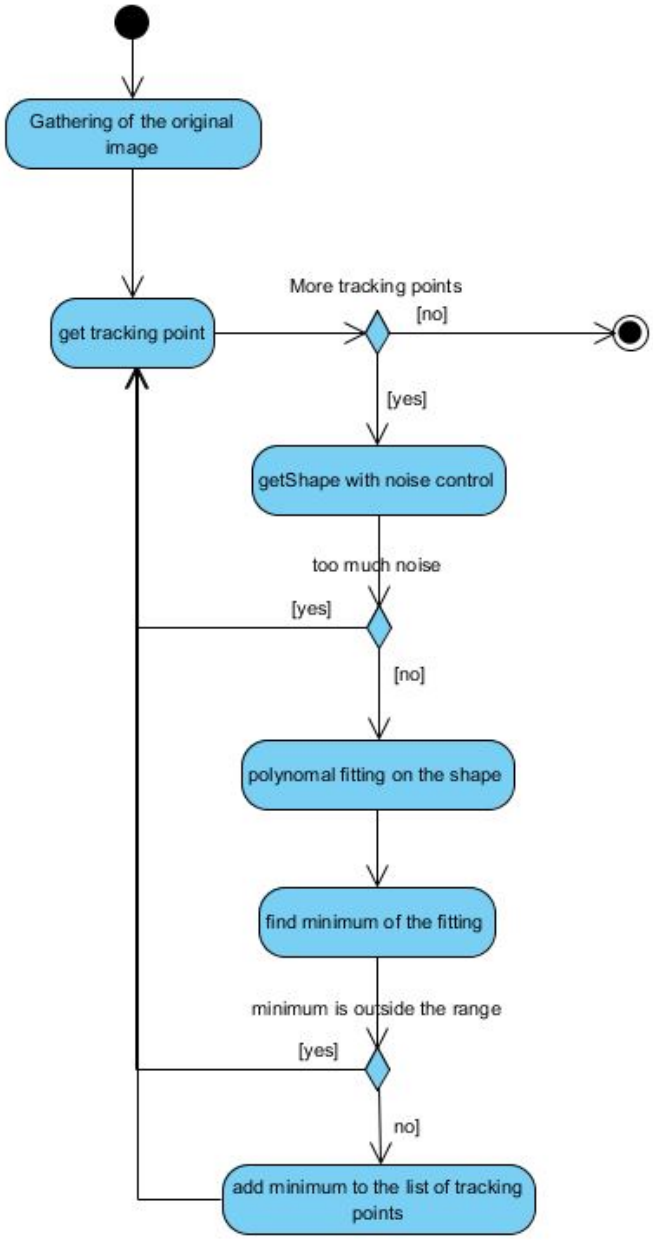


Figure 3.6: Activity diagram of the polynomial fitting idea

3.2.2 Iterative polynomial fitting with outlier

This second version should improve the first solution. When a polynomial fitting is applied, it may be that a single point is really distant from the produced function. The idea is to make many iterations until two conditions are fulfilled:

- No point is further from the function than a certain distance
- The number maximal of iterations has been reached

This algorithm will apply a fitting, check that every value is good. If not it removes the most distant point, make a fitting again, check that every value is good. If not, it removes the most distant point, make a fitting again. So goes on, until all the points are good or that the number maximal of iterations has been reached. **A maximal number** of iterations can also be translated by **a minimum allowed number of points**.

The algorithm is described in the activity diagram on figure 3.7:

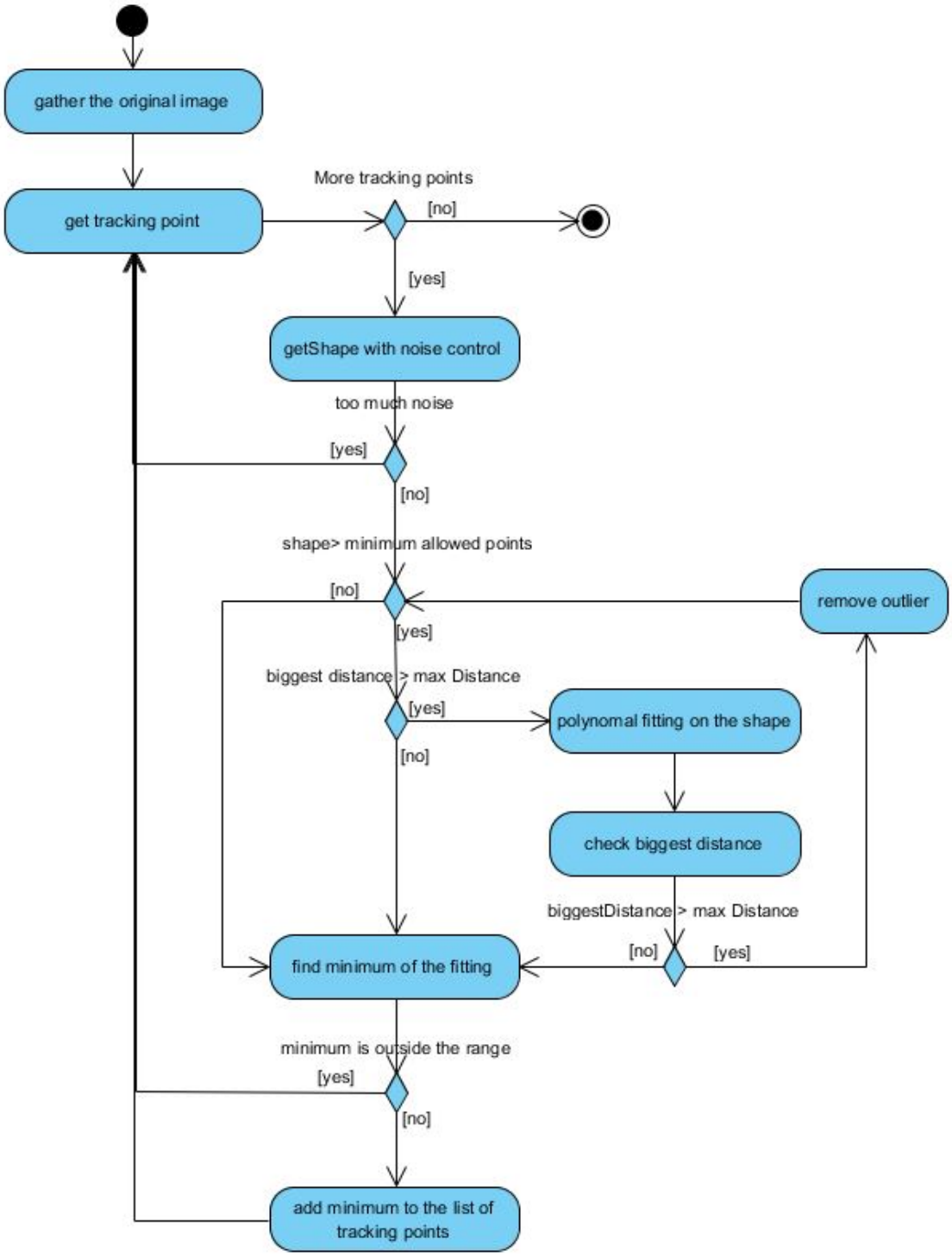


Figure 3.7: Activity diagram of the iterative polynomial fitting with outlier idea

3.2.3 Polynomial fitting by correlation with outlier

This third version will loop over each tracking point but will get the shape of two lines at the same time. If the algorithm is processing the tracking point **i**, it will get the shape of **i** but also the shape of **i+1**. After that, it subtracts one shape to the other and check that the result for each point is acceptable. Acceptable means that the difference between two

points should not be bigger than a certain number. This number will be a parameter of the solution. The idea is that if there are noisy points on the shape **i**, why would not this noise be present on the shape **i**+**1**.

The algorithm is described in the activity diagram on figure 3.8:

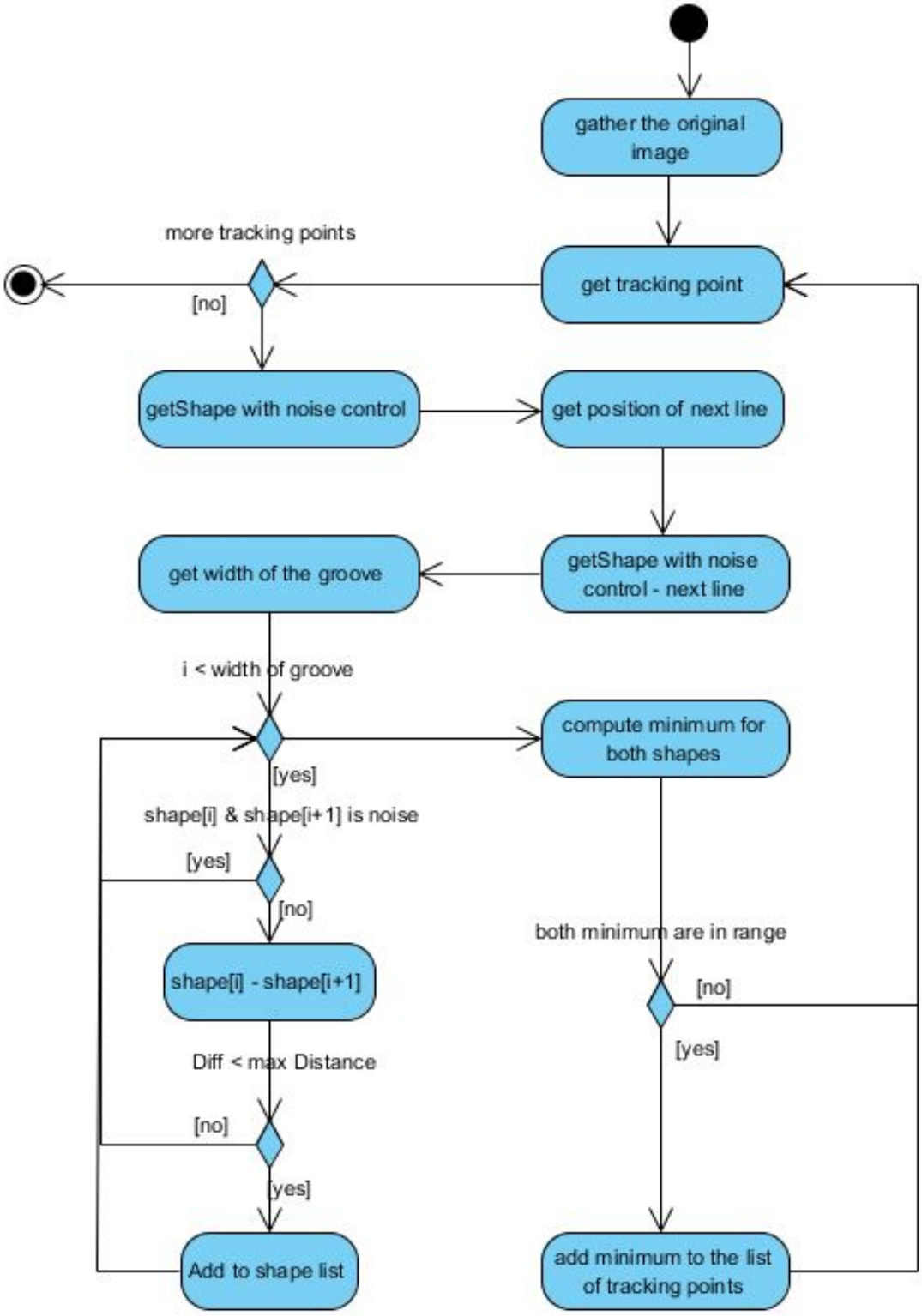


Figure 3.8: Activity diagram of the polynomial fitting by correlation with outlier

3.2.4 3D polynomial fitting

The plugin that will be implemented will make a different polynomial fitting than the three other ideas. The three other ideas only make a fitting over one line. This will make a fitting over one line. The formulas to do that are described have been the analysis phase and don't need to be designed. The following section presents the general schema of the plugin.

The plugin retrieve the number of lines and create a matrix containing the values x,y and z for each point for each line. A parameter containing the weight for each line will be added to the matrix. The function, tat make the 3D polynomial fitting, requires to solve the system of linear equations that has been presented in the analysis. After that, the minimum over the line **i** can be computed. There is still a risk that the point is outside the range of the groove. If it is, the point will be ignored and the algorithm will continue to the next point. Otherwise, it adds the point to the list of tracking points.

The algorithm is described in the activity diagram on figure 3.9:

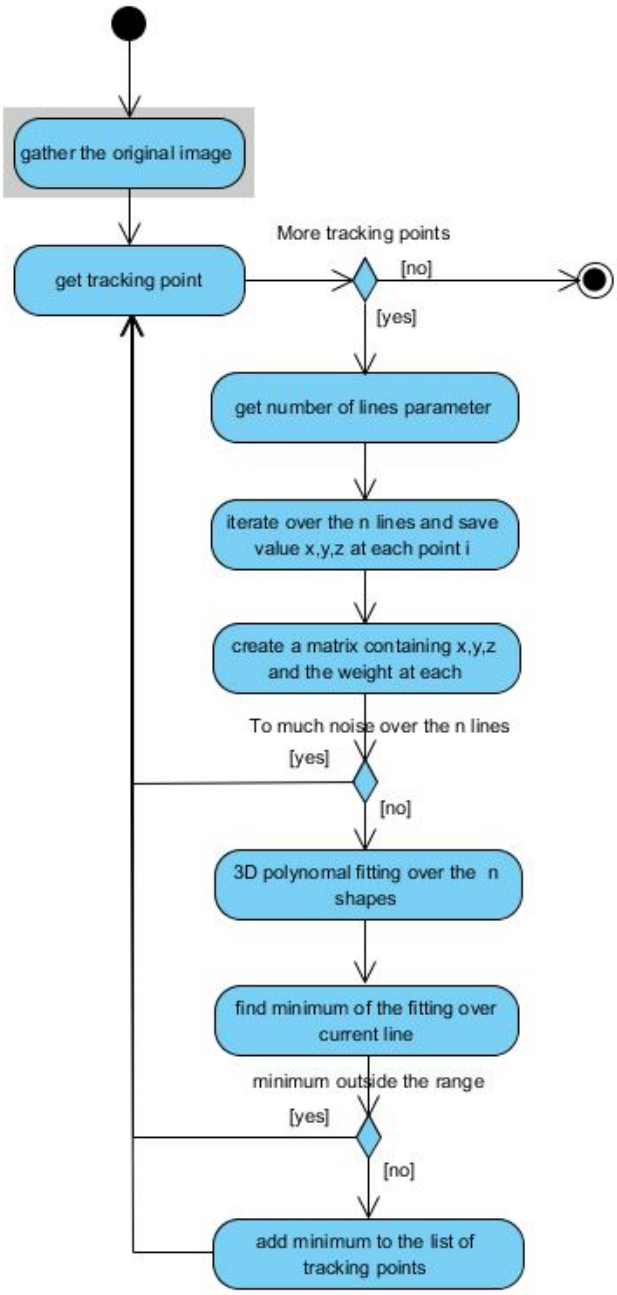


Figure 3.9: Activity diagram of the 3D polynomial fitting idea

3.3 Conclusion

This chapter has helped to imagine the algorithms that will be implemented in the next phase of the software development life cycle. It has helped to understand how the universal shape solutions work. This will be divided into three parts, first it will find the universal shape, then it will fill the universal shape at the right place and finally, it will fill the image between the grooves.

Once the universal shape solution applied. The system will need to apply a polynomial fitting at each point by ignoring the blobs. Four different versions have been imagined that will be implemented in the next phase.

Chapter 4

Implementation

This chapter presents some parts of the code that deserved additional attention. It also presents the different plugins that have been implemented with figures that displays the graphical user interface. Each plugin also has a figure that displays the expected output. This part is similar to the online documentation on the GIT repository.

4.1 Features of the code

The code is not the complicated part of the project. This section contains three parts that needed more explanation than the comments present in the code. First, the spline that is very easy to understand but more complicated to implement has been taken from the web. This section therefore presents how this code can been used. The four polynomial solutions needed a third comboBox as input. The existing code needed to be changed and this is explained in this section. Last, one of the four solutions needed a datagridView. This has never been used in this project before. The code also needed to be changed.

4.1.1 Spline interpolation

A plugin called "Spline Tracking" was created in order to create a continuous tracking. Indeed, if the tracking has been applied on a binned image, a lot of points will miss. Those need to be filled with an interpolation.

The code of the plugin is really easy to understand. It creates a list containing all the x values and a second list containing all the y values. However, the spline is not applied on the entire tracking. The reason is that the values of y are being repeated. It would be necessary to adjust the values for y. Another idea could be to make a spline for each pass. This is what have been implemented.

```
List<List<float>> xGrooves = new List<List<float>>();
List<List<float>> yGrooves = new List<List<float>>();
```

Listing 4.1: Plugin Spline - x and y values for each groove

The two lists will be filled until the end of the groove has been reached. Than a second loop will make a spline interpolation for each groove. A spline needs to have x and y values to fill the groove and to have x values where the spline will be evaluated at. The spline algorithm has not been implemented. A code on internet has been downloaded. This code can be found on CodeProject[7]. For that, four files have to be downloaded:

- ArrayUtils
- SplineInterpolator

- TestInterpolation
- TriDiagonalMatrix

These four files need to be added to the project. The plugin communicates with these files by calling the following method:

```
float[] xs = SplineInterpolator.Compute(yValues, xValues, ys,
    0.0f, Single.NaN, false);
```

Listing 4.2: Plugin Spline - call of the spline

The method SplineInterpolator takes six parameters. The first two are the points that will be used to create the spline. The third parameter contains the points on which the spline will be evaluated. The fourth and the fifth parameter constraint the first point and the last point. The sixth parameter can be set to TRUE for debugging purposes.

4.1.2 Addition of a third input plugin

The plugins PolyBlobReduction, PolyOutlierBlobReduction, PolyOutlierBlobReduction2 and Poly3DBlobReduction needed a third input plugin. All plugins inherit the abstract class PluginAbstract, this class needed to be changed. The following methods needed to be changed in order to give the possibility to the developer to have a third plugin as input:

- getFromIndex()
- getFrom2Index()

The two methods did not allow to have a third input. There were directly connected to the view elements **cboFrom1** and **cboFrom2**. Therefore, the method getFromIndex() has been generalized to be used with any comboBox:

```
public int getFromIndex(bool message, ComboBox comboBox, out
    PluginAbstract ctl)
{
    ...
}
```

Listing 4.3: Plugin Abstract - getFromIndex() changed

This allowed to call this method from any plugin as presented in the following code:

```
int from3 = getFromIndex(false, cboFrom3, out ctl3);
```

Listing 4.4: Plugin Abstract - getFromIndex() call

Each plugin that needs a third plugin should insert the code below in the running part of the plugin.

The original methods getFromIndex(bool) and getFromIndex2() need to be adapted in order to correspond to the new method:

```
public int getFromIndex(bool message)
{
    ...
    int from = getFromIndex(message, cboFrom, out tmpCtl);
    ...
}

public int getFrom2Index()
{
    ...
```

```
    int from2 = getFromIndex(false, cboFrom2, out tmpCtl);
    ...
}
```

Listing 4.5: Plugin Abstract - getFromIndex(bool) and getFromIndex2() changed

Addition of a ComboBox The combobox that allows to choose the third plugin should be inserted in any plugin that needs it. This plugin will also need to call the method mentioned below. The last thing that the plugin has to do is to insert the following code at the end of the file:

```
// It is specific to the third input plugin comboBox
public override void setIndex(int index, ControlCollection
   plugins, Del saveList)
{
    base.setIndex(index, plugins, saveList);
    int from3 = cboFrom3.SelectedIndex;
    cboFrom3.Items.Clear();
    cboFrom3.Items.Add(-1);
    for (int i = 1; i <= plugins.Count; i++)
    {
        if (i != index)  // ---- can't call self ----
        {
            cboFrom3.Items.Add(i);
        }
    }
         // used when the program restarts and the values have
            already been setted
        // If we fill the values, the selectedIndex will be loosed
        if (from3 > -1 && from3 <= cboFrom3.Items.Count)
        {
            cboFrom3.SelectedIndex = from3;
        }
}
```

Listing 4.6: Plugin Abstract - Overriding of the method setIndex

4.1.3 DataGridView element in 3DPolyFit

The plugin "3DPolyfit Blob" uses a Datagrid view to set the weight for each line. Indeed, the user can choose if a line is more important than another one. Adding a DataGridView imposes to change a few things in the Plugin Abstract. In the plugin Abstract, there are two methods that are used to save the values for each plugin and to restore the values for each plugin when the program starts. The methods are the following ones:

- ReadCTL
- WriteCTL

WriteCTL is used to save the values for each plugin in a ".plg" file. It works a follows: it loop through all the plugins through all the view elements. It detect which elements it is and save the value that are necessary to recreate the same element. If it is a label, it saves the text. If it is a checkbox, it saves the boolean behind it... The code to handle the DataGridView did not exist. This is the code that need to be inserted in the writeCTL method:

```
else if (typ.Contains("DataGridView"))
{
    DataGridView dgw = (DataGridView)uctl.Controls[i];
    StringBuilder strDgwRows = new StringBuilder();
    foreach (DataGridViewRow row in dgw.Rows)
    {
        strDgwRows.Append(row.Cells[0].Value);
        strDgwRows.Append("|");
        strDgwRows.Append(row.Cells[1].Value);
        strDgwRows.Append("|");
    }
    sw.WriteLine(parent + dgw.Name + ":" + strDgwRows.ToString());
}
```

Listing 4.7: Plugin Abstract - WriteCTl DataGridView

The restoration in the readCTL method has also been changed:

```
else if (typ.Contains("DataGridView"))
{
     DataGridView dgw = (DataGridView)uctl.Controls[i];
    if (hash.Contains(parent + dgw.Name))
    {
         dgw.Rows.Clear();
        string sRows = (string) hash[parent + dgw.Name];
        string[] spl = sRows.Split('|');
        for (int iSplit = 1; iSplit < spl.Length; iSplit += 2)
        {
            dgw.Rows.Add(new string[]{spl[iSplit - 1],
               spl[iSplit]});
        }
    }
}
```

Listing 4.8: Plugin Abstract - ReadCTl DataGridView

Finally an abstract method has been added to PluginAbstract: **endOfCTL**. This method is an event that will be called once the reading of the ".plg" file has been done. This method does nothing. The plugin choose what this method will do. The signature of the method is the following:

```
public virtual void endOfCTL(){}
```

Listing 4.9: Plugin Abstract - endOfCTL

4.2 Plugins

This section presents the different plugins that have been implemented. It presents the GUI as well as the visual results that can be awaited from this plugin. A short description is also presented.

These descriptions can also be found on the git repository.

4.2.1 Slope correction

Sometimes, the tracking fails because the groove is not similar on the entire disc. It can be that there are several differences between the depth at one point of the groove with

the depth at another point of the groove. This is due to a focusing problem during the recording with the precitec.

This feature can be seen on figure 4.1. The result after execution of the plugin can be visualized in figure 4.2. On figure 4.2, it can be seen that the function representing the depth of the groove is straight. In figure 4.1, this was not the case where it looks like a staircase.

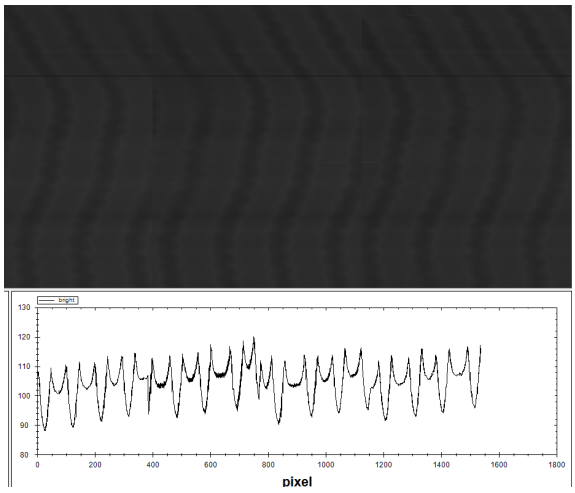


Figure 4.1: Plugin "SlopeCorrection" - before execution of the plugin

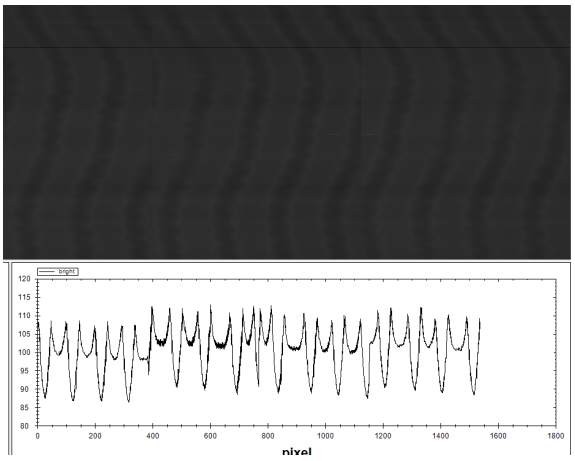


Figure 4.2: Plugin "SlopeCorrection" - after execution of the plugin

This plugin takes an image as input. The GUI of the plugin "Slope correction" can be seen in figure 4.3:

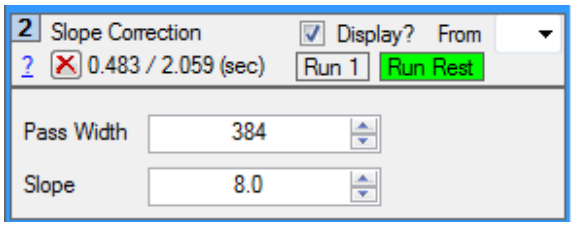


Figure 4.3: Plugin "SlopeCorrection" - graphical user interface

This plugin takes two additional parameters: *Pass Width* and *Slope*. *Pass Width* is the size of one pass. For instance, the precitec has a range of 192 points. If the scanning has been carried out with two sub-passes, this number will be 384. *Slope* is the difference between the lowest point of one pass and the highest point.

4.2.2 Adjust Tracking

This plugin takes an image and a tracking as input. As defined in the IRENE-plugin-system, each plugins knows the width and the height. This plugin takes the scalar between the two heights and the two widths and multiply the x and y coordinates for each point of the tracking with the scalar.

The tracking that will be inserted as input can be seen in figure 4.4 and the new tracking resulted after execution of this plugin can be seen in figure 4.5. By looking at the top left

side of the two pictures, it can be realized that the positions are different: the y has changed from 4051 to 40305.

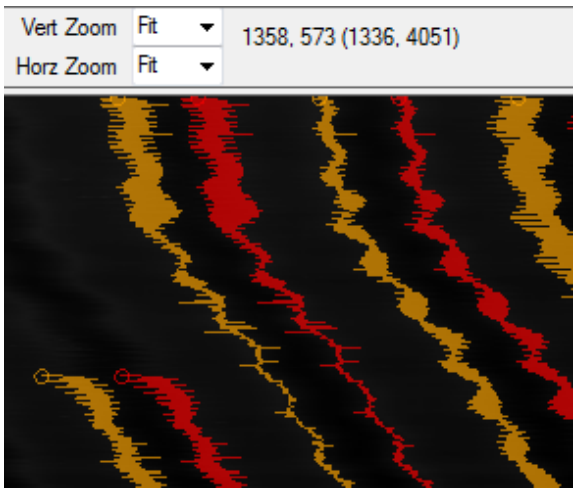


Figure 4.4: Plugin "AdjustTracking" - before execution of the plugin

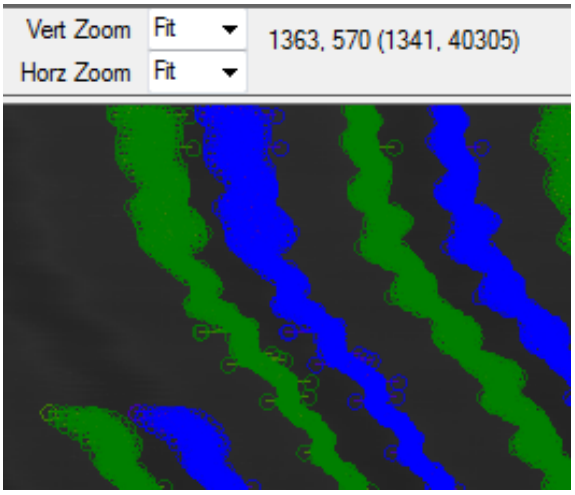


Figure 4.5: Plugin "AdjustTracking" - after execution of the plugin

The GUI of the plugin "Adjust Tracking" can be seen in figure 4.6:

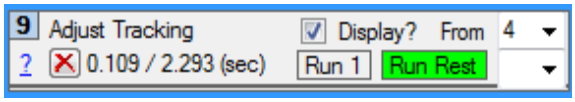


Figure 4.6: Plugin "Adjust Tracking" - graphical user interface

This plugin takes no additional parameters.

4.2.3 Compute Middle

A point in a tracking can be represented by a single point (x,y) or by two points. This can be used for stereo mode. This plugin computes the average between the two x values and creates a new tracking with the average values.

The figure 4.7 shows a tracking that contains two different x positions for each point of the tracking list. The figure 4.8 shows the result after execution of the plugin.

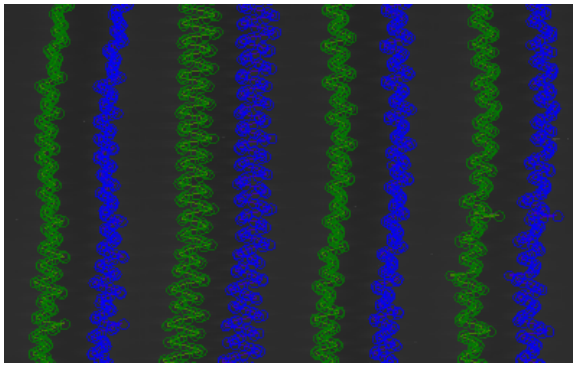


Figure 4.7: Plugin "Compute Middle" - before execution of the plugin

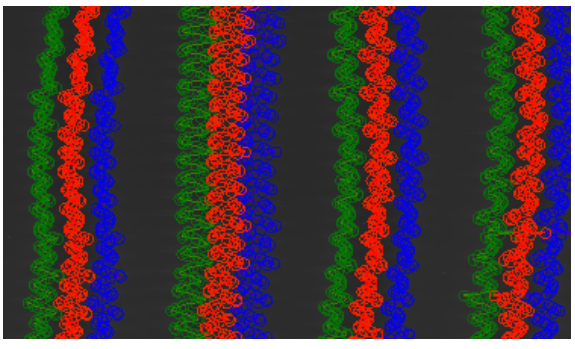


Figure 4.8: Plugin "Compute Middle" - after execution of the plugin

The GUI of the plugin "Compute Middle" can be seen in figure 4.9:

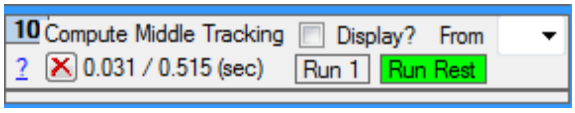


Figure 4.9: Plugin "Compute Middle" - graphical user interface

This plugin takes no additional parameters.

4.2.4 Spline Tracking

If a tracking is not continuous, it might be useful to fill the points that are missing. This plugin will apply a spline interpolation on the tracking points. The tracking in red in figure 4.10 represents a tracking that is not continuous and the blue tracking represents the tracking after execution of the plugin.

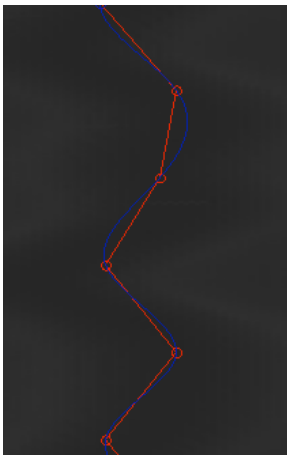


Figure 4.10: Plugin "Spline Tracking" - before(red) and after(blue) execution of the plugin

The GUI of the plugin "Spline Tracking" can be seen in figure 4.11:

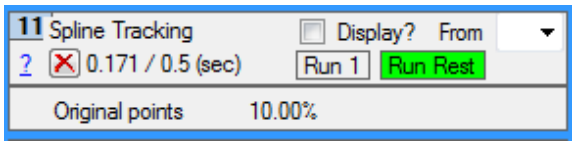


Figure 4.11: Plugin "Spline Tracking" - graphical user interface

This plugin takes no additional parameters.

4.2.5 Linear Interpolation

If a tracking is not continuous, it might be useful to fill the points that are missing. This plugin will apply a linear interpolation on the tracking points. The tracking in red in figure 4.12 represents a tracking that is not continuous and the blue tracking represents the tracking after execution of the plugin.

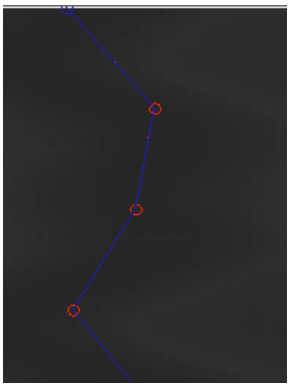


Figure 4.12: Plugin "Linear Interpolation" - before(red) and after(blue) execution of the plugin

The results is very similar but the big difference is that the tracking has a point at each y coordinate. The red one is similar only because the graphical user interface plot it like this.

The GUI of the plugin "Linear Interpolation" can be seen in figure 4.13:

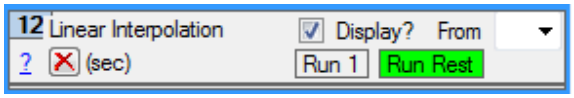


Figure 4.13: Plugin "Linear Interpolation" - graphical user interface

This plugin takes no additional parameters.

4.2.6 Universal Shape

The universal shape is a plugin that create a flat image. This flat image can be subtracted from the original image in order to create a sub-flat image. This plugin works as follow. It follow the tracking and compute a universal shape for each groove on each pass. In the example that can be seen in figure 4.14, 13 different universal shapes have been computed. The data between two grooves has been filled with a linear interpolation.

Figure 4.14: Plugin "Linear Interpolation" - before(red) and after(blue) execution of the plugin

This plugin takes two plugins as input. The first one should be an original image on which the universal shape will be computed and the second one should be a tracking that

the plugin can follow in order to create the flat image. The tracking should be continuous in order to obtain better results.

This plugin takes two additional parameters: *Groove width* and *Median Mode*. *Groove width* has to be chosen by looking at the original image and by estimating the with of the groove. This is required by the plugin to decide the width of the groove.

Median Mode can be checked if the universal shape used the median of all the shapes or the average of all the shapes.

The GUI of the plugin "Universal Shape" can be seen in figure 4.15:

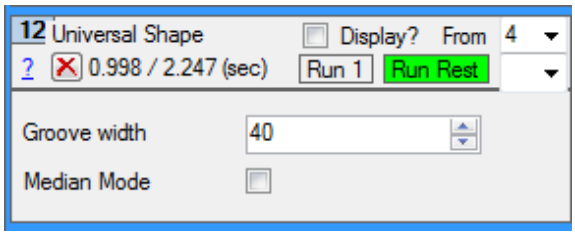


Figure 4.15: Plugin "Universal Shape" - graphical user interface

4.2.7 PolyFit Blob

This plugins takes three plugins as input. The first input plugin is the original image. The second input plugin is a tracking that has to be continuous. The third input plugin is an image that has labeled the blobs. A point is labeled as blob if the value is different than 0. These points will be ignored for the process.

The plugin makes a polynomial fitting at each point over the shape of the groove. The first parameter *Groove width* set up the size of the groove. If this value is 40 like in figure 4.16, the plugin will take 20 points on the left of the tracking point and 20 on the right. It will use these 40 points and create a polynomial fitting. Once done, it minimizes the fitting and insert the position of x and y on the shape where the depth is minimized.

Sensitivity is a parameter that will decide how many noise is allowed on a shape before applying the fitting.

The GUI of the plugin "Polyfit Blob" can be seen in figure 4.16:

Figure 4.16: Plugin "Polyfit Blob" - graphical user interface

The result after execution of the plugin will give a new tracking.

4.2.8 PolyFit Outlier Blob

This plugins takes three plugins as input. The first input plugin is the original image. The second input plugin is a tracking that has to be continuous. The third input plugin is an image that has labeled the blobs. A point is labeled as blob if the value is different than 0. These points will be ignored for the process.

The plugin makes a polynomial fitting at each point over the shape of the groove. The first parameter *Groove width* set up the size of the groove. If this value is 40 like in figure 4.16, the plugin will take 20 points on the left of the tracking point and 20 on the right. It will use these 40 points and create a polynomial fitting. After that, it will test that every point is not too far away from the fitting. This is defined by the parameter *Max distance*. If it is too far away, it removes the points and makes a new fitting. This plugins iterates many times until no point is too far away or that the minimal number of points has been reached. This is defined by the second parameter *Min points*. This plugin also have a parameter to defined sensitivity. It means that the plugin will not start the iterations process if there are too many noisy points.

The GUI of the plugin "Polyfit Outlier Blob" can be seen in figure 4.17:

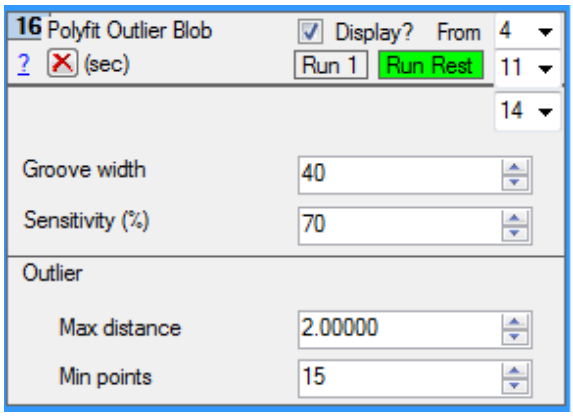


Figure 4.17: Plugin "Polyfit Outlier Blob" - graphical user interface

The result after execution of the plugin will give a new tracking.

4.2.9 PolyFit Outlier 2 Blob

This plugins takes three plugins as input. The first input plugin is the original image. The second input plugin is a tracking that has to be continuous. The third input plugin is an image that has labeled the blobs. A point is labeled as blob if the value is different than 0. These points will be ignored for the process.

This plugin uses the fact that there is a correlation between two consecutive grooves. It will create a polynomial fitting by ignoring the blob points and by ignoring the points that are too different with the next groove. What does it mean? The plugin subtract one shape to the other shape and check that there difference matches with the parameter *Max distance*. If the difference is higher, the point will be ignored. After that, a polynomial fitting is applied and the minimum is computed.

The GUI of the plugin "Polyfit Outlier 2 Blob" can be seen in figure 4.18:

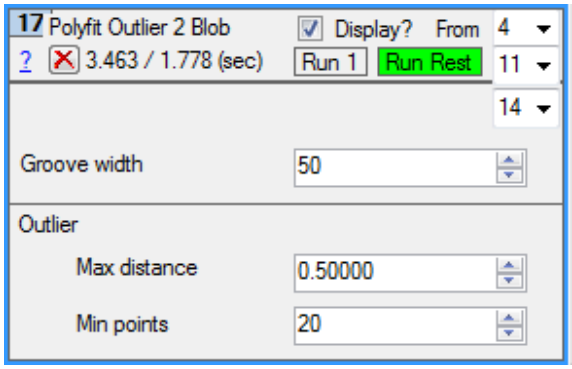


Figure 4.18: Plugin "Polyfit Outlier 2 Blob" - graphical user interface

The result after execution of the plugin will give a new tracking.

4.2.10 3D PolyFit Blob

This plugins takes three plugins as input. The first input plugin is the original image. The second input plugin is a tracking that has to be continuous. The third input plugin is an image that has labeled the blobs. A point is labeled as blob if the value is different than 0. These points will be ignored for the process.

This plugins also uses the fact that there are correlations between the grooves. It makes a polynomial fitting over many lines. The number of lines is defined by the parameter *Number of Lines*. By changing this value, the plugin also to set the weight for each line. By this way, the impact of some lines will be reduced if the weight is big.

The GUI of the plugin "3D Polyfit Blob" can be seen in figure 4.19:

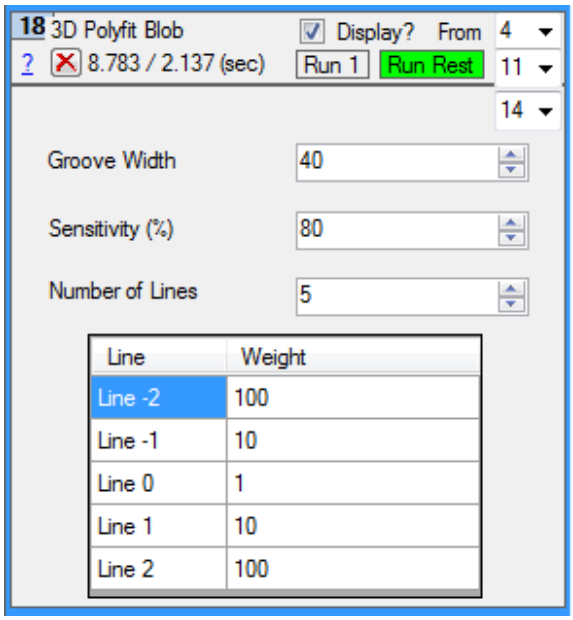


Figure 4.19: Plugin "3D Polyfit Blob" - graphical user interface

The result after execution of the plugin will give a new tracking.

4.3 Synthesis

The code himself is very intuitive and does not deserve a lot of attention in this report. However, two parts are important. The addition of a third input plugin took some time. The view needed to be changed but also the class "Plugin Abstract". The method getFromIndex needed adjustment to be more general.

"Plugin Abstract" also needed to be changed to integrate DataGridView elements. The readCTL and writeCTl have been changed. Now the elements present in DataGridView can be saved and restored.

This section has also presented the GUI for each plugin that has been implemented during the project. It contains a small description and figures to illustrate the execution of the plugin. This part can be very interesting for a developer that wants to know how a plugin works without having to read the entire report.

Chapter 5

Tests and Validation

This chapter will test the different implementations and make a comparison between them. It will be divided in two parts. The first part will test the universal shape solution in order to validate if the noise is well detected or not. The second part will test the different polynomial fitting implementations and compare them in order to know which version should be used in which case. In both cases, the tests will show if some parameters need to be adjusted by using formulas or by analyzing the input data.

Before beginning with tests, one should know what happened during the project. The lens of the Precitec has been changed. The problem was discovered before this bachelor project by the Irene team by making scanning with different exposure times. They came to the conclusion that 1 milliseconds give bad results. Therefore a few discs have been scanned with an exposure time of 2 milliseconds or more. During this project, it was necessary to scan new discs. The Irene team decided that the exposure time of 1.6 milliseconds gave good results. More than a dozen scans were carried out with this value. Subsequently, the lens was changed and it became possible to carry out scanning with an exposure time of 1 milliseconds. This had two major effects on the data. First, the scan rate could be increased because each scan was faster due to the decreasing of the exposure time. Second, the data is also less noisy. The assumption that can be made is that the old lens was dirty. No matter if this assumption is true, it led to rescan some discs. This means that the tests will be based on the discs that have been scanned twice in order to make good comparisons.

5.1 Conventions and lenses

During these tests, two conventions have been used that will make the understanding easier. The first thing is the word **disc** that can be misunderstood. Disc does not mean the entire disc but one side. For instance, "disc 4147" does not mean the disc 4147 but the side 4147. On the other side of the same disc, there is the 4148. In order to make it easier to understand, disc 4147 and disc 4148 are not related even if it is not exactly true.

Another convention that has been used is the use of the exposure time to differentiate two scannings of a same disc. Usually, two scanning have been applied on each disc, one with 1 ms exposure time and one with 1.6 ms exposure time. In fact, the main difference between these two scannings is not that they have been scanned with a different exposure time but that the Precitec had a different lens between the two scanning. The difference in the exposure time is a consequence of the changing of the lens.

The tests contain personal judgments on the results. The idea is not to focus the attention on my opinions but the reality is that some tests needed to be conducted by myself. Without tools and without adequate virtual support, the only way to qualify the quality of the sound by listening to it is to mention my judgment that tried to be objective as much as possible .

Serial number of the lenses The scanning that were carried out with 1 ms exposure time have used the new lens and the scanning that were carried out with 1.6 ms or 3 ms exposure time have used the old lens. The serial number of each of the lenses are the following:

Table 5.1: Lenses used for the project

	Serial number
Old Lens	15.07.0199
New Lens	15.39.0239

5.2 Noise detection

These tests will validate the quality of the universal shape detect the noise. For these tests, different tiff file will be tested:

- disc 4147 | exposure time of 1 milliseconds | 2 sub pass | 0.8 step size | new lens
- disc 4148 | exposure time of 1 milliseconds | 2 sub pass | 0.8 step size | new lens
- disc 68 | exposure time of 1 milliseconds | 2 sub pass | 0.8 step size | new lens
- disc 70 | exposure time of 1 milliseconds | 2 sub pass | 0.8 step size | new lens
- disc 858 | exposure time of 1 milliseconds | 2 sub pass | 0.8 step size | new lens
- disc 7137 | exposure time of 1 milliseconds | 2 sub pass | 0.8 step size | new lens
- disc 4147 | exposure time of 3 milliseconds | 1 sub pass | 0.96 step size | old lens
- disc 4147 | exposure time of 1.6 milliseconds | 2 sub pass | 0.8 step size | old lens
- disc 4148 | exposure time of 1.6 milliseconds | 2 sub pass | 0.8 step size | old lens
- disc 68 | exposure time of 1.6 milliseconds | 2 sub pass | 0.8 step size | old lens
- disc 70 | exposure time of 1.6 milliseconds | 2 sub pass | 0.8 step size | old lens
- disc 858 | exposure time of 1.6 milliseconds | 2 sub pass | 0.8 step size | old lens
- disc 7137 | exposure time of 1.6 milliseconds | 2 sub pass | 0.8 step size | old lens

5.2.1 Overview of the tests

Figure 5.1 presents the plugin used during these tests.

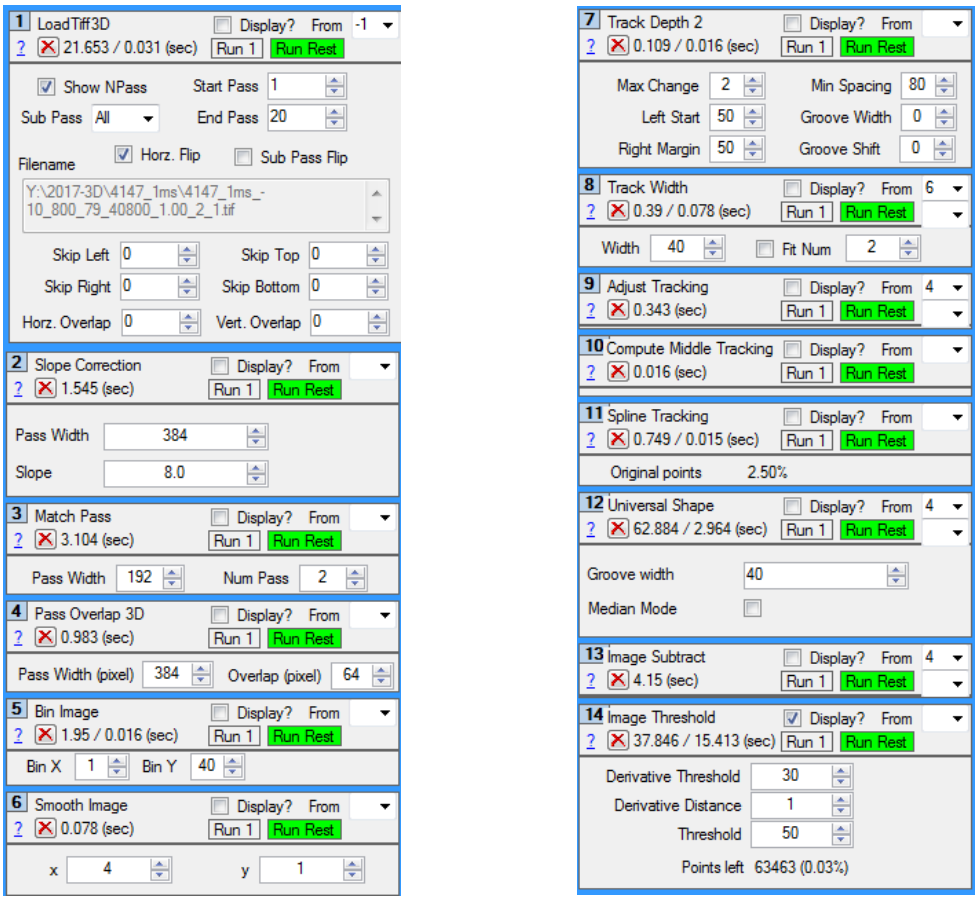


Figure 5.1: Plugin used for the tests "Noise detection"

The tests use 14 plugins. The plugins have parameters that must be decided once for all tests or once per disc. The values for each plugin are the described below.

1. LoadTiff3D: **Start Pass** will always be 1 because it is sometimes possible that the scanning has been started too early and that there isn't any data on the first pass. **End Pass** will be 20 because it is not necessary for this test to validate on the entire disc. This represents more-less a quarter of the disc. Sometimes, the start pass and the end pass may change because it was not possible to find an interesting point in these passes.

Horz Flip needs to be applied if the scanning has been processed with 2 sub-passes.

2. Slope Correction *Pass Width* is n times 192. N is the number of sub passes. 192 is the number of points that the Precitec can scan at once. If the number of sub-passes is two, this number is 382. *Slope* needs to be changed for every disc. As mentioned in the analysis, this value represent the highest difference between the bottom of two grooves in each pass.

3. Match pass *Pass Width* is the width of the Precitec pass. It will always be 192 if the scanning has been made with the Precitec. *Num Pass* is the number of sub-passes. In this test, it will be two or one. If it is one, this plugin is not necessary.

4. Pass Overlap 3D This plugin is needed if an overlap exists between two passes. In other words, if the size of a step is smaller than the size of the precitec range. This plugin will be needed to manage the overlaps. In this tests, the size of the step is either 0.8 millimeters or 0.96 millimeters. If the step size is 0.96 millimeters, this plugin is not

necessary. If the step size is 0.8 millimeters, the size of the overlaps can be computed by using the rule of three formula (the number of points in a size is 384):

$$384 - 384 * 0.96/0.8 = 64 \tag{5.1}$$

The two parameters will always be 384 and 64 for each of the discs that have been used for the tests. However, if the number of sub-passes is 2, this plugin is not necessary.

5. Bin Image The binning in X is 1 otherwise the motion of the data would be lost. It has been decided than the binning in Y will always be 40. If this number is bigger, it might be that the groove resulted by the universal shape will be too different then the original groove. If this number is too small, it might be that the universal shape is badly influenced by the noise. The number 40 seemed to be working well during the entire development process and was therefore adopted.

6. Smooth Image This plugin helps to smooth the shape of the groove. It is required to erase the small irregularities in the groove's shape. This number cannot be too big because it would modify the shape of the groove. This number has to be modified for each disc but it will be around 4.

7. Track Depth 2 The following parameter are fixed for each disc. One parameter might change if the disc has been scanned with one pass: *Min Spacing* would be 40. For the other scans, this parameters stays 80.

8. Track Width The width of the groove is 40 for the discs scanned with two passes.

9. Adjust Tracking No parameter. It adjusts the tracking to the original image. In this case, the original image has to be the result of the pass overlay if the scan took more than 1 pass. If the scan used only one pass, this plugin will use the original image produced by the first plugin.

10. Compute Middle Tracking This plugin takes no parameter and compute the middle of the tracking if the tracking has for each tracking point different x values. In these tests, it will always be the case.

11. Spline Tracking The tracking has been applied on a binned image. Once adjusted with the plugin "Adjust Tracking", the tracking will not contain a tracking point for each y value. That is not acceptable and the missing points have to be found by using an interpolation. For these tests, the spline interpolation will be used but a linear interpolation could also work. The spline has more chances to be more representative than the linear one.

12. Universal Shape This plugin takes two parameters that can easily be chosen. The first **Groove width** is the width of the groove that is 40 for the two-passes-scans and 20 for the for the one-pass-scans. The second parameter allows you to choose between median and average. In these tests, this parameter will always be average (not checked) because it seemed that it did not make any difference between the two options.

13. Image Subtract This plugin will subtract the universal shape to the original image. In this case, the original image has to be the result of the pass overlay if the scan took more than 1 pass. If the scan used only one pass, this plugin will use the original image produced by the first plugin.

14. Image Threshold This plugin has three parameters that need to be tested in order to see what gives the best noise detection. These are highly depending on the data and the values are indicated in the following tests. The parameter derivative distance will not be tested because this parameter is not useful in this process.

Summary In the tests, three parameters will be tested over 13 tiff files. The 13 tiff files are mentioned in the introduction of this section. In addition, some parameters needs to be decided for each disc. The three parameters that have to be decided and that can change the result are present in the plugin "Image Threshold". The parameter that are depending on the discs are the ones that concern the width of the pass and the slope adjustment. These tests have been conducted by processing the plugins and by looking on the original image. The idea is to see if the process finds the blobs and if it finds points that should not be considered as blobs.

5.2.2 Goal of the tests

The following pages present the results of the tests as well as the parameter used. The idea of these tests is to see if it is possible to find an optimal set of values for the two following parameters:

- Plugin "Image Threshold" - Threshold
- Plugin "Image Threshold" - Derivative Threshold

The first parameter is very easy to find and is presented in the first test. The second parameter is presented in the conclusion part of this section.

Some parameters change depending on the disk being tested. These are the following:

- Plugin "LoadTiff3D" - Start Pass(usually 1 but might change)
- Plugin "LoadTiff3D" - End Pass(usually 20 but might change)
- Plugin "SlopeCorrection" - Slope
- Plugin "SlopeCorrection" - Pass Width (384 for 2 sub-passes, 192 for 1 sub-pass)
- Plugin "TrackDepth2" - Min Spacing (80 for 2 sub-passes, 40 for 1-sub-pass)
- Plugin "TrackWidth" - Width (40 for 2 sub-passes, 20 for 1 sub-passes)

Because these values are specific to the discs and to the scanning, they are presented in the section 5.4.

5.2.3 Test 1: Parameter Threshold

This test will found the best value for the parameter *Threshold* for the plugin "ImageThreshold". To find this value, it is very easy. One should use the process that has been described in the overview. 13 plugins are necessary. The last plugin on which the test will focus on is the plugin "Subtraction". This plugin creates a sub-flat image containing noise as well as some audio points. The GUI allows to click on this image and to see a function of the depth. By clicking almost anywhere on the image, the function should be similar to the function that can be seen on figure 5.2:

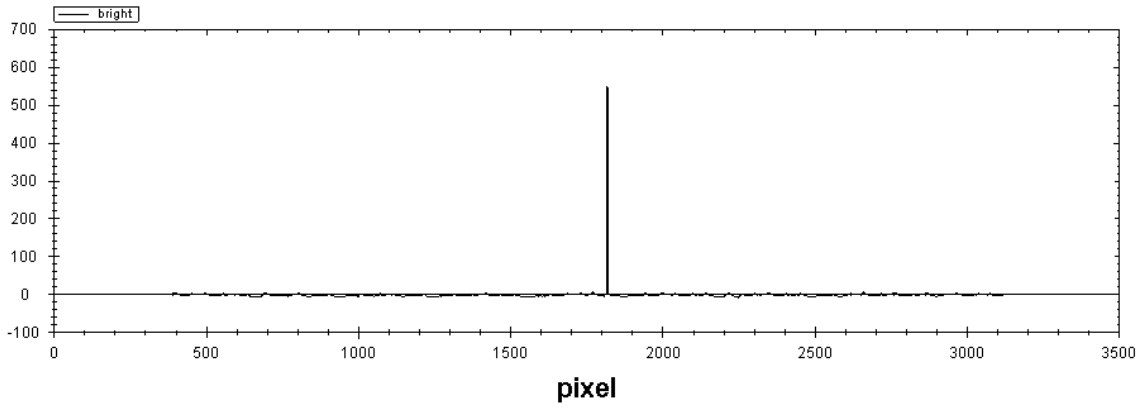


Figure 5.2: Function of the depth after subtraction

The peak that can be seen on figure 5.2 represents a blob that is really lightweight (like dust). This represents one part of the blobs that can be found by changing the parameter "threshold" of the plugin "Image Threshold". This can only be seen after the subtraction processed by the plugin "Image Subtract". However this value can not be too small and not too big. In this particular case, the value can not be higher than 500. This feature can be observed on any disc at almost any location. If the disc is clean, this will not be the case.

So which value should be used? 500 is too big, and the value can also not be too small. 50 has been chosen and will be used for the next tests.

5.2.4 Procedure for the threshold parameter

The next tests will be very similar. This section describes the procedure that has been used to test the noise detection.

Six discs have been tested but there is a difference between the tests applied on the discs 4147, 4148 and 60 and the tests applied on the discs 70, 858, 7137. On the first three discs, two locations were found that will be compared between the different scannings. Each disc has been scanned with 1 milliseconds exposure time and with 1.6 milliseconds exposure times. The idea is to compare the original images of these two scanning together as well as the threshold-images. Each threshold-image has been processed by changing the parameter "Derivative Threshold" in the plugin "Image Threshold". One image represents a bad choice for the parameter. A bad choice is a choice where the threshold plugin filters on the blobs but also on the movement of the groove. A second image represents a good choice. It means that only the blobs have been found. The third image represents an alternative to the second image. This image shows only the blobs after the threshold but they are bigger than with the second image. It is important to understand that the value of the parameters are not always similar. The idea was to find three solutions that give similar results. This might be subjective from a certain point of view but using the same values would have led to non-comparable results.

One the three last discs (70, 858, 7137), the procedure is similar but only one location has been used.

Therefore, it means that the tests will show four images for each location. One representing the original image and three representing the results after threshold. The value of the parameter "derivative threshold" is indicated in the label of each image.

The tests also indicate the x-y coordinates of any location on every file. This could be useful if the tests were to be carried out again. Each location is represented by two x-y

coordinates: top-left point and bottom-right point. To find these points again, the value of the parameters presented in the section 5.4 have to be used.

5.2.5 Test 2: disc 4147

5.2.5.1 First location

For this test, only one threshold image has been processed for the 1-milliseconds-scanning. The reason will be explained in the conclusion of these tests (see section 5.2.11).

Version scanned with 1 milliseconds exposure time:
The x-y coordinates of the top-left point are (2898, 18474). The x-y coordinates of the bottom-right point are (6068, 19670).

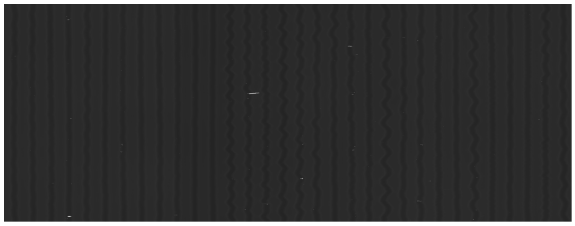


Figure 5.3: 1 ms-scanning-image

Figure 5.4: Derivative Threshold: 20

Version scanned with 1.6 milliseconds exposure time:
The x-y coordinates of the top-left point are (2808, 32988). The x-y coordinates of the bottom-right point are (5958, 34136).

Figure 5.5: 1.6 ms-scanning-image

Figure 5.6: Derivative Threshold: 15

Figure 5.7: Derivative Threshold: 20

Figure 5.8: Derivative Threshold: 30

Version scanned with 3 milliseconds exposure time:
The x-y coordinates of the top-left point are (1335, 8759). The x-y coordinates of the bottom-right point are (2919, 9333).

Figure 5.9: 3 ms-scanning-image

Figure 5.10: Derivative Threshold: 15

Figure 5.11: Derivative Threshold: 20

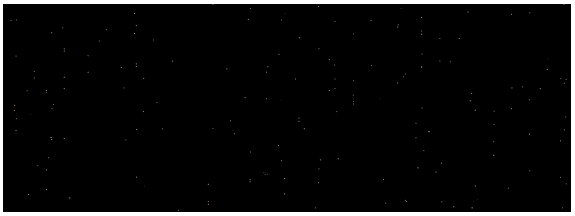


Figure 5.12: Derivative Threshold: 30

Analysis of the test: By looking at the original images (figures 5.3, 5.5 and 5.9), it is possible to see that the 1-milliseconds image is less noisy than the other two. Another thing that can be seen is that the noise is not similar to the three images. On figure 5.3, it is possible to see a blob on the middle-left part of the image. This blob has not been detected by the two other scanning. This might be interpreted in different ways. The first idea could be that the dust has been inserted during the process by human interaction. Another idea could be that the dust moves on the discs. The last idea is that the lens of the precitec is not defective. The conclusion will discuss about these three hypotheses (see Section 5.2.11).

Figure 5.4 shows the result of the derivative threshold 20. This has the feature that of showing the noise without showing other elements such as the groove. This is exactly what a noise detection should do.

Figure 5.6 is way to sensitive. This could not be used properly because one might see the border of the grooves. This can only be seen on the figure 5.10.

Figures 5.7 and 5.11 shows more than figures 5.8 and 5.12. The difference is not huge but the noise is bigger on the first ones than on the second ones. At this point, it is hard to say which one is better. The conclusion will detail the difference between these two solutions (see section 5.2.11).

5.2.5.2 Second location

This second location is very interesting because a big scratch can be seen on the original image.

Version scanned with 1 milliseconds exposure time:
The x-y coordinates of the top-left point are (2672, 28320). The x-y coordinates of the bottom-right point are (5830, 29518).

Figure 5.13: 1 ms-scanning-image

Figure 5.14: Derivative Threshold: 8

Figure 5.15: Derivative Threshold: 10

Figure 5.16: Derivative Threshold: 30

Version scanned with 1.6 milliseconds exposure time:
The x-y coordinates of the top-left point are (2574, 2766). The x-y coordinates of the bottom-right point are (5728, 3908).

Figure 5.17: 1.6 ms-scanning-image

Figure 5.18: Derivative Threshold: 15

Figure 5.19: Derivative Threshold: 20

Figure 5.20: Derivative Threshold: 30

Version scanned with 3 milliseconds exposure time:
The x-y coordinates of the top-left point are (1318, 17460). The x-y coordinates of the bottom-right point are (2897, 18029).

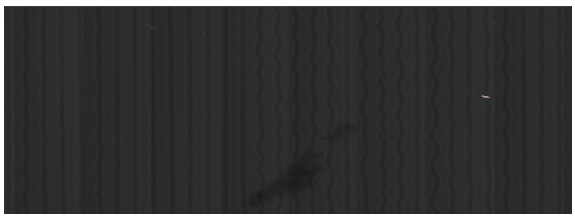


Figure 5.21: 1.6 ms-scanning-image

Figure 5.22: Derivative Threshold: 15

Figure 5.23: Derivative Threshold: 20

Figure 5.24: Derivative Threshold: 30

Analysis of the test: Same verdict as for the first location, the 1 ms scanning image is less noisy than the two others(the 1.6 ms is especially noisier). For a moment, let us ignore the big scratch. The comparison of the figures 5.14, 5.18 and 5.22 shows that the three thresholds are too sensitive. Indeed, the groove can be seen. The comparison of the figures 5.15, 5.19 and 5.23 shows that the threshold are better. It is still possible to see the groove but by looking at the original data, it is possible to realize that it is indeed noise. The three last figures 5.16, 5.20 and 5.24 are not sensitive enough but would still be useful.

One might definitely not ignore the big scratch too long. The two first thresholds of the 1-milliseconds scanning shows the contours of the scratch. In the last threshold, it is possible to distinguish the scratch. The same verdict can be made for the 1.6-milliseconds scanning as well as for the 3-milliseconds scanning. Distinguishing the contours of the scratch might not be enough. This phenomenon is explained in the conclusion part of these tests.

5.2.6 Test 3: disc 4148

5.2.6.1 First location

Version scanned with 1 milliseconds exposure time:
The x-y coordinates of the top-left point are (772, 11266). The x-y coordinates of the bottom-right point are (3922, 12414).

Figure 5.25: 1 ms-scanning-image

Figure 5.26: Derivative Threshold: 10

Figure 5.27: Derivative Threshold: 15

Figure 5.28: Derivative Threshold: 20

Version scanned with 1.6 milliseconds exposure time:
The x-y coordinates of the top-left point are (750, 11050). The x-y coordinates of the bottom-right point are (3864, 12120).

Figure 5.29: 1.6 ms-scanning-image

Figure 5.30: Derivative Threshold: 10

Figure 5.31: Derivative Threshold: 15

Figure 5.32: Derivative Threshold: 20

Analysis of the test: It is possible to see that the blobs on the two scanning are not related. This leads to the same hypotheses mentioned in the previous test. In the figures 5.26 and 5.30, it is possible to see the groove. It is more extreme in the 1.6-milliseconds scanning. In both cases, the threshold 15 and threshold 20 are very similar but the threshold 15 might be more sensitive. The threshold 10 is the only one that shows more information than blobs. The other one are really good.

5.2.6.2 Second location

Version scanned with 1 milliseconds exposure time:
The x-y coordinates of the top-left point are (2986, 29500). The x-y coordinates of the bottom-right point are (6138, 30640).

Figure 5.33: 1 ms-scanning-image

Figure 5.34: Derivative Threshold: 10

Figure 5.35: Derivative Threshold: 15

Figure 5.36: Derivative Threshold: 20

Version scanned with 1.6 milliseconds exposure time:
The x-y coordinates of the top-left point are (2926, 29364). The x-y coordinates of the bottom-right point are (6084, 30502).

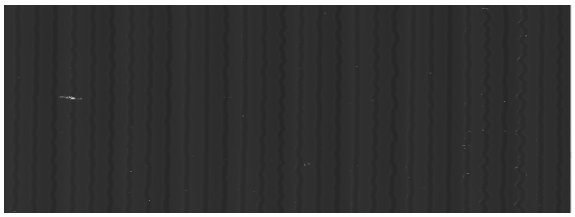


Figure 5.37: 1.6 ms-scanning-image

Figure 5.38: Derivative Threshold: 15

Figure 5.39: Derivative Threshold: 20

Figure 5.40: Derivative Threshold: 30

Analysis of the test: For this test, the 1-milliseconds results are way much better than the 1.6-milliseconds results. In the 1.6-milliseconds results, the groove can be seen on any results (right part of the image). In fact, noise is present on the groove and has been detected during the thresholding. For the 1-milliseconds, the threshold 15 is more sensitive and seems more useful than the 20. The threshold 10 might also work even if it is possible to distinguish the groove on the right side of the image 5.34. However, the threshold 20 is also good. For the 1.6-milliseconds, the threshold 15, 20 and 30 could work. The treshold 15 is maybe too sensitive.

5.2.7 Test 4: disc 68

It seems that this disc has more dust than the previous two discs 4147 and 4148. It has however no more scratches than the previous discs. In this case, the values of the threshold are far apart (5, 15 and 30). This is not the case for the previous tests. The reason is that 15 and 20 gave almost the same results.

5.2.7.1 First location

Version scanned with 1 milliseconds exposure time:
The x-y coordinates of the top-left point are (568, 23986). The x-y coordinates of the bottom-right point are (3270, 25092).

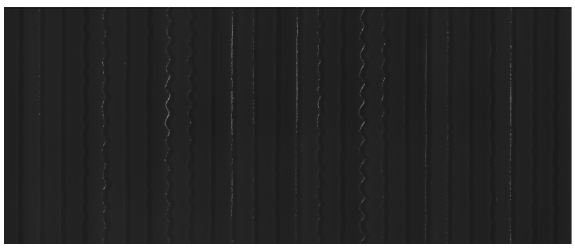


Figure 5.41: 1 ms-scanning-image

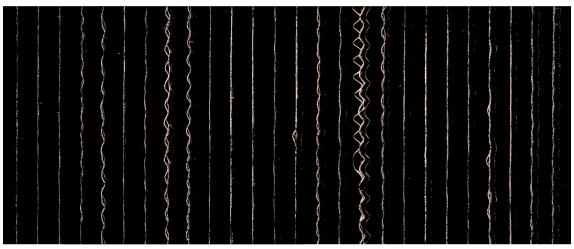


Figure 5.42: Derivative Threshold: 5

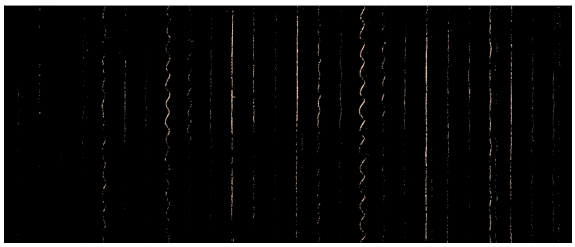


Figure 5.43: Derivative Threshold: 15

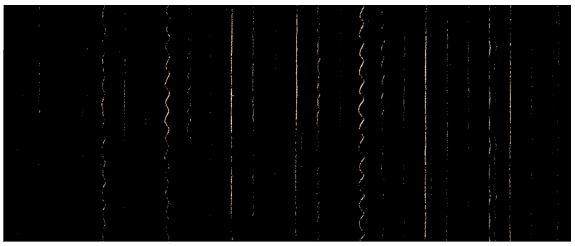


Figure 5.44: Derivative Threshold: 30

Version scanned with 1.6 milliseconds exposure time:
The x-y coordinates of the top-left point are (525, 18898). The x-y coordinates of the bottom-right point are (3280, 19750).

Figure 5.45: 1.6 ms-scanning-image

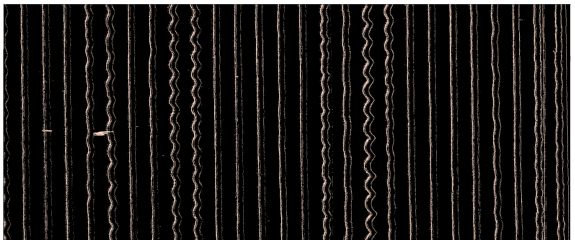


Figure 5.46: Derivative Threshold: 5

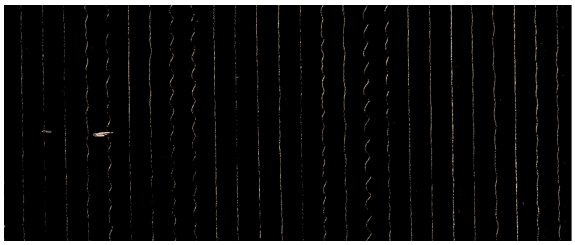


Figure 5.47: Derivative Threshold: 15

Figure 5.48: Derivative Threshold: 30

Analysis of the test: A first remark that can be made is that there are many blobs on the edges of the grooves. The threshold 5 (figures 5.42 and 5.46) is way too sensitive because it is possible sometimes to see the two sides of the groove. The threshold 15 (figures 5.43 and 5.47) is in both cases acceptable. Indeed, the noise is perfectly detected. The threshold 30 (figures 5.44 and 5.48) does not reveal all the blobs.

5.2.7.2 Second location

At this location, there is a big scratch visible on both scanning.

Version scanned with 1 milliseconds exposure time:
The x-y coordinates of the top-left point are (0, 234). The x-y coordinates of the bottom-right point are (2694, 1332).

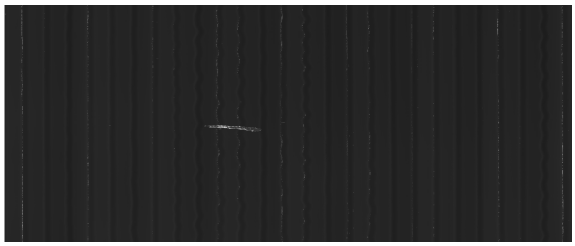


Figure 5.49: 1 ms-scanning-image

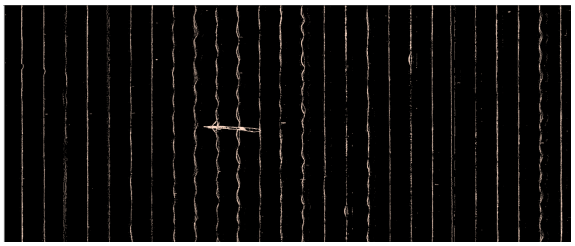


Figure 5.50: Derivative Threshold: 5

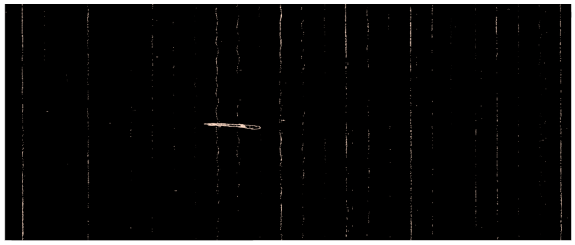


Figure 5.51: Derivative Threshold: 15

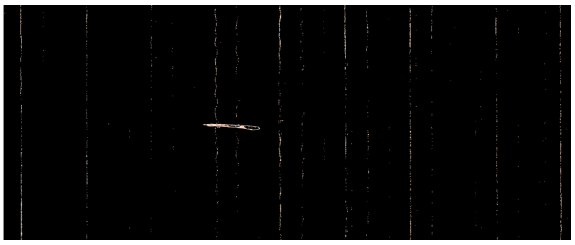


Figure 5.52: Derivative Threshold: 30

Version scanned with 1.6 milliseconds exposure time:
The x-y coordinates of the top-left point are (0, 32858). The x-y coordinates of the bottom-right point are (2694, 33976).

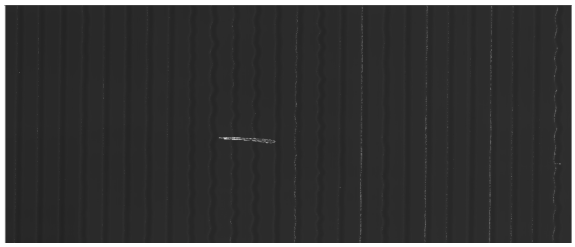


Figure 5.53: 1.6 ms-scanning-image

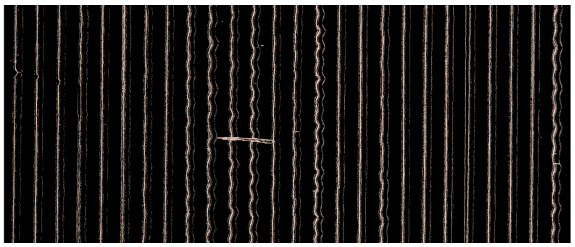


Figure 5.54: Derivative Threshold: 5

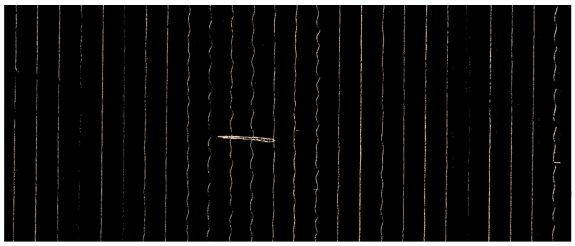


Figure 5.55: Derivative Threshold: 15

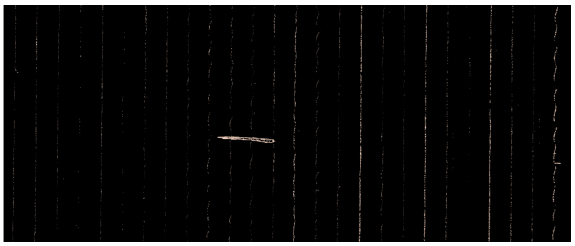


Figure 5.56: Derivative Threshold: 30

Analysis of the test: The verdict is the same as for the first location. The threshold 5(figures 5.50 and 5.54) is in both cases too sensitive. The threshold 15(figures 5.51 and 5.55 works better then the threshold 30(figures 5.52 and 5.56) than is not sensitive enough. One remark that can be made is that the big scratch is always detected.

5.2.8 Test 5: disc 70

Version scanned with 1 milliseconds exposure time:
The x-y coordinates of the top-left point are (0, 3960). The x-y coordinates of the bottom-

right point are (2694, 5080).

Figure 5.57: 1 ms-scanning-image

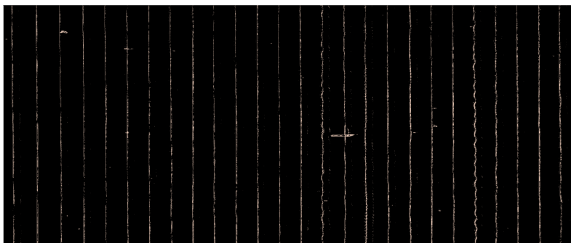


Figure 5.58: Derivative Threshold: 5

Figure 5.59: Derivative Threshold: 15

Figure 5.60: Derivative Threshold: 30

Version scanned with 1.6 milliseconds exposure time:
The x-y coordinates of the top-left point are (0, 3264). The x-y coordinates of the bottom-right point are (2690, 33744).

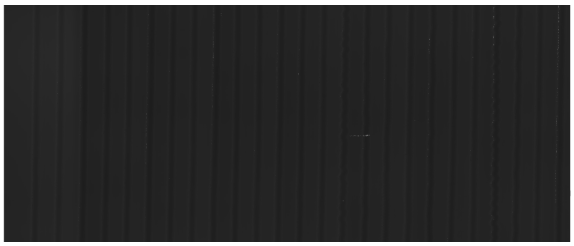


Figure 5.61: 1.6 ms-scanning-image

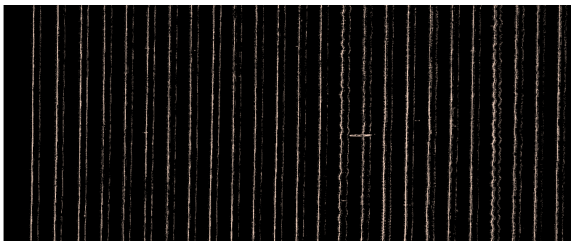


Figure 5.62: Derivative Threshold: 5

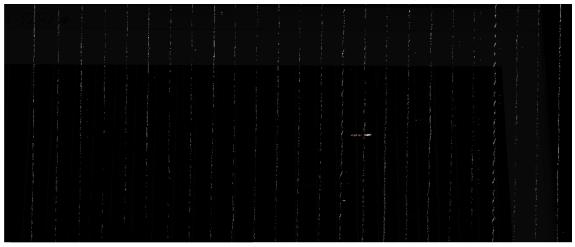


Figure 5.63: Derivative Threshold: 15

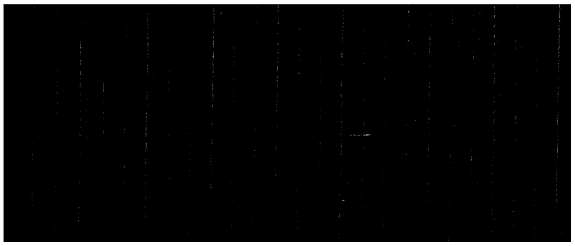


Figure 5.64: Derivative Threshold: 30

Analysis of the test: In this case, the values of the threshold are far apart again (5, 15 and 30). The reason is that 15 and 20 gave almost the same results.

The 1.6-milliseconds scanning has again more dust than the 1-milliseconds scanning. But in both cases, the threshold 15(figures 5.59 and 5.63) is the value that works the best, the value 30(figures 5.60 and 5.64) is not sensitive enough and the value 5(figures 5.58 and 5.62) is too sensitive.

5.2.9 Test 6: disc 858

Version scanned with 1 milliseconds exposure time:
The x-y coordinates of the top-left point are (4832, 39064). The x-y coordinates of the bottom-right point are (7524, 40164).

Figure 5.65: 1 ms-scanning-image

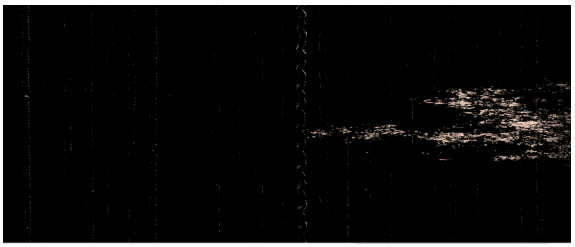


Figure 5.66: Derivative Threshold: 5

Figure 5.67: Derivative Threshold: 10

Figure 5.68: Derivative Threshold: 20

Version scanned with 1.6 milliseconds exposure time:
The x-y coordinates of the top-left point are (5044, 29830). The x-y coordinates of the bottom-right point are (7732, 30954).

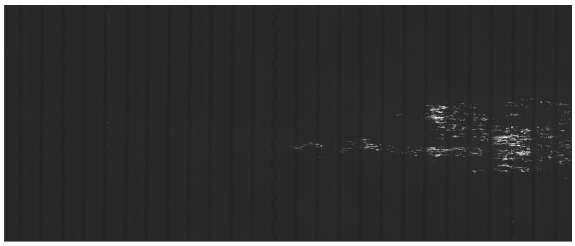


Figure 5.69: 1.6 ms-scanning-image

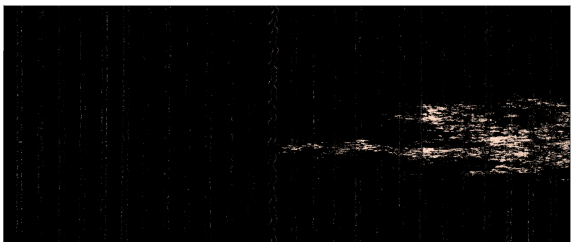


Figure 5.70: Derivative Threshold: 10

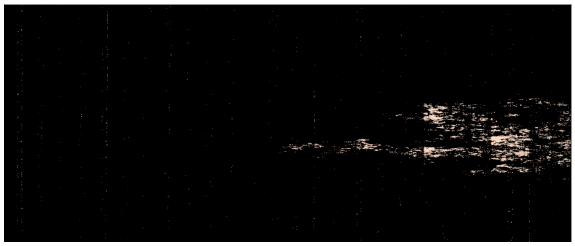


Figure 5.71: Derivative Threshold: 15

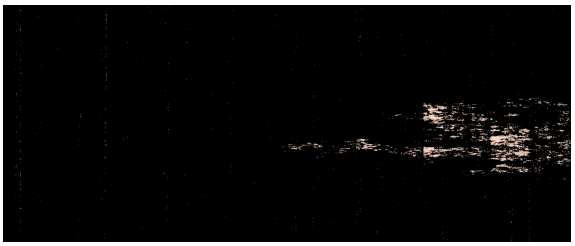


Figure 5.72: Derivative Threshold: 20

Analysis of the test: This disc does not have a lot of dust but this location of the disc shows a big dust. Moreover, this dust can be seen on both scanning. This is one of

the first times that a specific blob can be seen on both scanning (1 milliseconds and 1.6 milliseconds).

In the first scanning, the threshold 5 is too sensitive. Indeed, the groove is not as dirty as what suggest the figure 5.66. The thresholds 10(figure 5.67) is really good because it really shows the dust and nothing more. The threshold 20(figure 5.68) is not sensitive enough. The same analysis can be made on the 1.6 milliseconds scanning.

In both cases, the big dust is always detected and that is good. Sometimes, that could be a little better.

5.2.10 Test 7: disc 7137

Version scanned with 1 milliseconds exposure time:
The x-y coordinates of the top-left point are (476, 19762). The x-y coordinates of the bottom-right point are (3166, 20868).

Figure 5.73: 1 ms-scanning-image

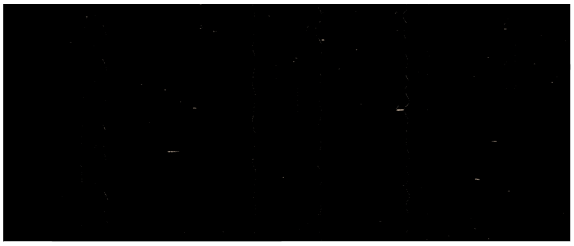


Figure 5.74: Derivative Threshold: 5

Figure 5.75: Derivative Threshold: 10

Figure 5.76: Derivative Threshold: 20

Version scanned with 1.6 milliseconds exposure time:
The x-y coordinates of the top-left point are (390, 32690). The x-y coordinates of the bottom-right point are (3470, 35784).

Figure 5.77: 1.6 ms-scanning-image

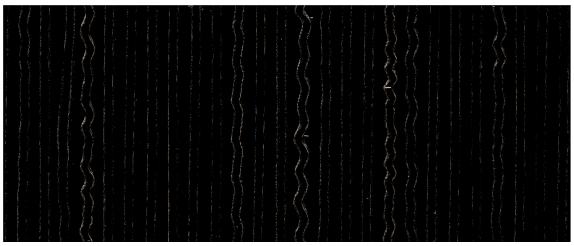


Figure 5.78: Derivative Threshold: 5

Figure 5.79: Derivative Threshold: 10

Figure 5.80: Derivative Threshold: 20

Analysis of the test: This record is relatively clean and it was difficult to find a location that allows to compare and analyze. The threshold 5 is too sensitive for the 1.6 milliseconds scanning(figure 5.78) but is not that bad for the 1 milliseconds scanning (figure 5.74). The threshold 10 and 20 are in both cases very similar. It is hard to say that the threshold 10 is better than the threshold 20.

5.2.11 Conclusion of the tests

In the tests 2-7, different values of threshold have been tested in order to find three solutions for each disc for each scanning. The values for the derivative threshold are between 5 and 30. It has shown many times that the value 5 is way too sensitive and that the value 30 is not sensitive enough. In most of the cases the values 15 and 20 gave similar results. Sometimes, 15 was a little bit too sensitive and 20 was sometimes not sensitive enough. It depends on which discs, the value is tested.

The tests have also shown that the scanning of 1 milliseconds was better than the scanning of 1.6 milliseconds. The scanning with 1-millisecond exposure time had fewer blobs than the scanning with 1.6 milliseconds exposure time. The scratches could be found on both scanning and that is normal. There are many answers to the following question: why has the scanning of 1 milliseconds less blobs than the other scanning? For that, three reasons exist. The first one is that the dust is on the lens of the Precitec. There are many reasons to think that this is not the right answer. One should know that the lens of the Precitec has recently been changed(few days before the scanning) and that all the scanning with 1 millisecond exposure time have been taken with the new lens. This new lens should be very clean. But test 4147 showed that a drop was only found on the scanning with the new lens (1 ms scanning). The second reason is that the dust is moving around the disc. What does it mean? The discs are stored in a transparent tote. It may be that the dust moves every time the disc is removed or put back into the tote. A third reason could be that the human interaction with the disc has changed the dust present on the disc. So, it could be that dust may be deposited on the disc during the processor. It is impossible in this project to know exactly the reasons: these are hypotheses.

The tests have also shown an improvement potential. The second location of the disc 4147(see 5.2.5.2) contains a big scratch that can also be seen in the following figure 5.81.

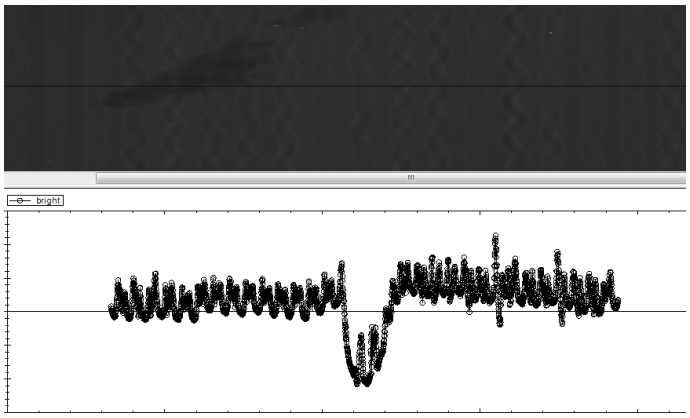


Figure 5.81: Function of the depth on the big scratch on the disc 4147 (the line is 0)

The top of the figure shows the results of the process after subtraction of the original image with the universal shape. The scratch that was very deep is still very deep. The values of the points are below zero(after subtraction). For the moment, it is impossible to detect that. The threshold plugin can find the contours by using the "derivative threshold" parameter but cannot find the inside of the blobs. The parameter "threshold" can only see the dust that is light. An improvement could be to add a parameter that thresholds below a certain value. However, no one can say(before testing) that it will help to produce better sound, but it will certainly help to improve the noise detection

To conclude: The best value for the "derivative threshold" parameter is between 15 and 20. 15 might be more sensitive than 20. Another reason why 15 might be a good value is that 20 might be really too bad on the entire image. The tests have only shown a part of the disc. Sometimes, 20 is really too big. For future uses, it can be asserted that 15 is the right value. In this way, it would simplify the choice of the user. It has also shown in the test 1 that 50 was a good value for the parameter "threshold". There are many improvements that can be made in future projects.

5.3 Polynomial fitting

This section will test the most important part of this project: the creation of the sound based on TIFF files. For these tests, six files have been tested with the four different algorithms. Each algorithm has a different number of parameters. Some of them are specific to the TIFF file and some of them can be tuned. In these tests ,many different parameters have been tested. Sometimes, it does not change the results, sometimes it does. In this section, the purpose was to investigate in order to find the best algorithm, if there is one.

First, the different tiff files that have been tested are the following ones:

- disc 4147 | exposure time of 1 milliseconds | 2 sub pass | 0.8 step size | new lens
- disc 4148 | exposure time of 1 milliseconds | 2 sub pass | 0.8 step size | new lens
- disc 858 | exposure time of 1 milliseconds | 2 sub pass | 0.8 step size | new lens

And the algorithm that have been implemented and are being tested in this section are:

- Polynomial fitting over one line (PolyFit Blob)

- Iterative Polynomial fitting over one line with outlier parameter (Polyfit Outlier Blob)
- Polynomial fitting by using the noise of two lines with outlier parameter (Polyfit Outlier 2 Blob)
- Polynomial fitting 3D over many lines (3D Polyfit Blob)

The test can be divided into two parts. One part has tested the four solutions over the five discs with five different values for the parameters. The binning of the image that has been applied was 40. This seemed to be a reasonable value at the beginning of the project and has never been changed. The test 1 to 4 give the results for a binning of 40. The second part will contains some tests that have been carried out on the discs 4147 with four different binning (5,10,20,40). The tests 5 and 6 contains many tests about the change of the binning.

5.3.1 Overview of the tests

A part of the plugins used for these tests have already been presented in section 5.2.1. To these plugins, four plugins needs to be added to the process. The last plugin will create the audio file.

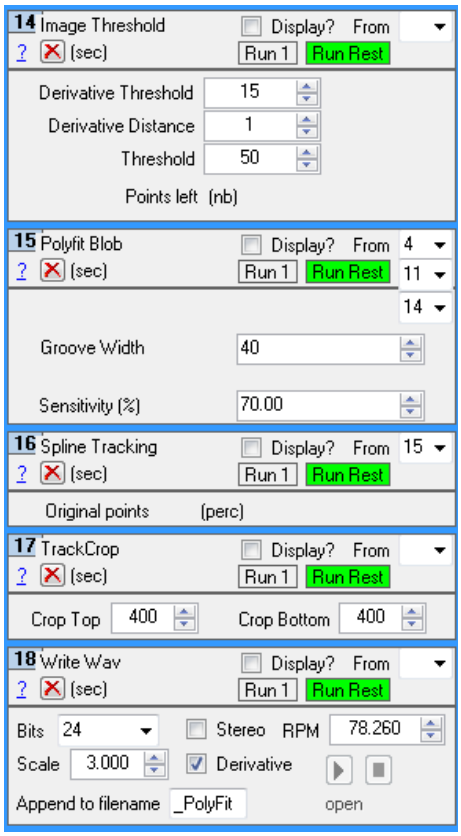


Figure 5.82: Parameter used for the tests "Polynomial fitting"

The tests use 18 plugins. The plugin 14 "Image Threshold" has been tested in the previous section. The plugin 15 "Polyfit Blob" is one of the four plugins that are beeing tested. Its parameters are explained in the goal part of this section. For the three last plugins, the next lines will give an answer on how to use them.

16. Spline Tracking: Each of the four solutions have a chance to produce a tracking that is not complete. It means that it is likely possible that some points are missing.

However, in order to produce the sound, the list of tracking point have too contain Y values that are continuous. The spline interpolation will solve the problem. A linear interpolation may also work. Usually, an interesting groove is not linear but rather looks like a sinusoidal curve. A spline helps to have a final curve that is nearer to the reality. This plugin takes no parameters.

17. TrackCrop Before starting the scanning, it is possible to choose two parameters "Scan Start" and "Scan End". These parameters represents an angle. All the scanning that are used for these tests have been scanned with the value 0 for "Scan Start" and the value 367.2 for "Scan End". Another parameter in LabView define the sample rate. For these scanning, the value was 40'800. The goal is to sample one rotation with 40'000. Where does the 800 come from? This has already been explained in the analysis. The idea is to use a rule of three to find 800.

$$40'000 = 360 \tag{5.2}$$

$$? = 7.2 \tag{5.3}$$

The answer is 800. It means that 800 extra points have been scanned one each rotation. Those points have to be taken away. Otherwise, some points will be added twice to the audio file. This plugins takes two parameters that allow to take away a certain amount of points on the start of the rotation(top) or at the end of the rotation(bottom). It has been decided to take away half of the points on the top(400) and half of the points on the bottom(400).

18. Write Wav The values for this plugin are always similar. The aluminum discs of the Milman Parry collection run with more-less 78 RPM. The number of bits texitBits is 24 in order to have more different numbers. This number could also be 32 or 16. If the memory allows it, it is better to choose a big number for this parameter. It has been decided to choose 24. That gives more than 16 million different values. The scale is a parameter that will chance the volume of the audio sound. It could be useful to change this parameter if two audio files don't have the same volume. In these tests, the scale will always be 3 for each solution. *Stereo* is useful if the motion of the groove is different on the two sides of the groove. This is not the case for the discs that have been tested. This parameter will not be checked. *Derivative* has to be checked. Each polynomial fitting solution gives a tracking that cannot directly been used to create the sound. The sound is the derivative between two neighbors values. It is important to check this box. Otherwise, the resulting audio file will have an indescribable sound.

5.3.2 Goal of the tests

The goal is to test the four solutions and to find for each solution the best values for the parameters. This section will also answer the two following questions: which algorithm is the best or should a solution be improved to have better results?

The four plugins that are tested are presented on the figures 5.83 and 5.84.

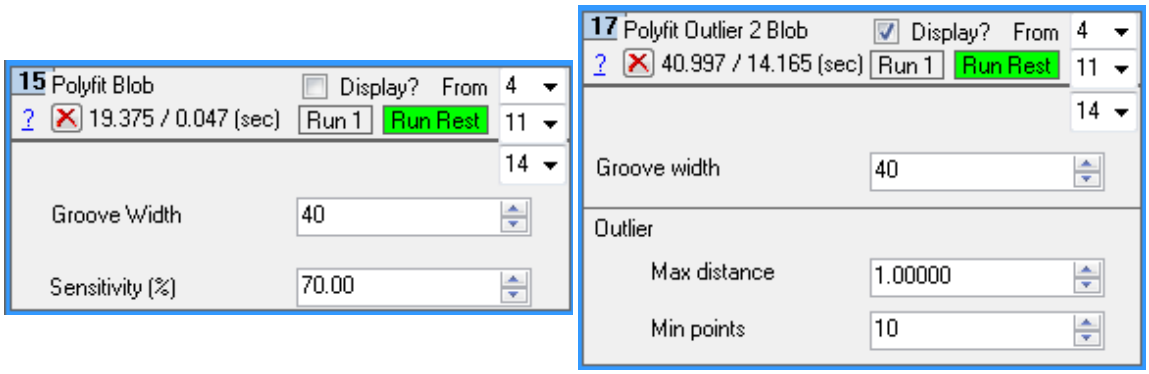


Figure 5.83: Interfaces for the four "polynomial fitting" solutions (1)

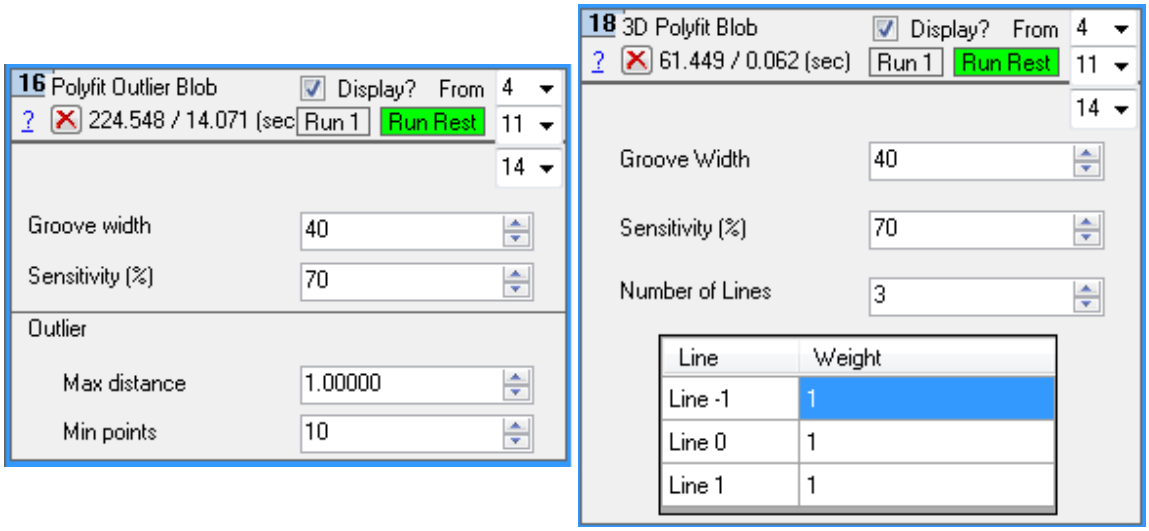


Figure 5.84: Interfaces for the four "polynomial fitting" solutions (2)

On each of these plugins, the parameter "Groove Width" is fixed to 40 because this is the width of the groove. Moreover, this is consistent with the global parameters presented in section 5.4. It means that this parameter will not be tested. The parameters that have been tested are for each plugin the following ones:

- Plugin "Polyfit Blob" - Sensitivity
- Plugin "Polyfit Outlier Blob" - Sensitivity
- Plugin "Polyfit Outlier Blob" - Max distance
- Plugin "Polyfit Outlier Blob" - Min points
- Plugin "Polyfit Outlier 2 Blob" - Max distance
- Plugin "Polyfit Outlier 2 Blob" - Min points
- Plugin "3D Polyfit Blob" - "Line-weight" with different number of lines

The sensitivity in the plugin "3D Polyfit Blob will not be tested because it will only make it worst. In this plugin, this parameter can be used to be more restrictive. If this number is too high, there is a lot of chances that no tracking will be made. If this number is too low, the resulting fitting will not be representative enough of the reality. 70 is a choice that has been made and the tests will use this value. Moreover, the interesting parameters for this plugins are the weights for each lines that can be modified.

5.3.3 Procedure of the tests

The procedure for the tests is the following one: different sets of values for the parameters have been tested on the six discs mentioned in the introduction of this section (see 5.3). The different values that have been tested are presented in the introduction of each test. For the first solution "Polyfit Blob", three values have been tested for the parameter "Sensitivity". For the three other solutions, four solutions have been tested. The results of these tests shows the spectrum at a high frequency, a low frequency. The different spectrum for each solution are presented in the results of each test.

Beside that, a subjective opinion from myself will also be presented in the analysis of the test's results. The reason is that the results can also be heard via headphones and this is very subjective, especially because the author of this document, Matthias Christe, is not an audio scientist. For this opinion, a filter ReaFir has been used on the audio file.

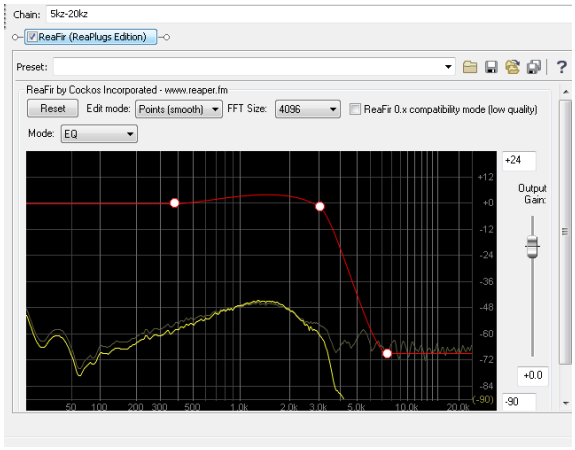


Figure 5.85: Use of ReaFir in SoundForge

In these tests, this plugin filters and changes the high frequencies.

The title of each test will include the name of the disc as well the exposure time that has been used during the scanning. It also contains the name of the algorithm that is being tested.

For the test, three discs have been tested on low and high frequencies parts. The portion of the disc 858 did not have any high frequencies. The reason is that someone is speaking. For the discs 4147 and 4148, it is possible to hear someone singing.

5.3.4 Test 1: Polyfit

This test will compare the different parameters that have been chosen for the test. In this test, it will not validate the performances of the polynomial fitting over another idea but validate the best value for the parameters.

It has already been mentioned but the plugin **Polyfit** only have one interesting parameter: *Sensitivity*. For this problem the three values that have been tested are the following:

Table 5.2: Value of the parameters - 3 Tests for the Polyfit idea

	Test 1	Test 2	Test 3
Sensitivity	70	80	60

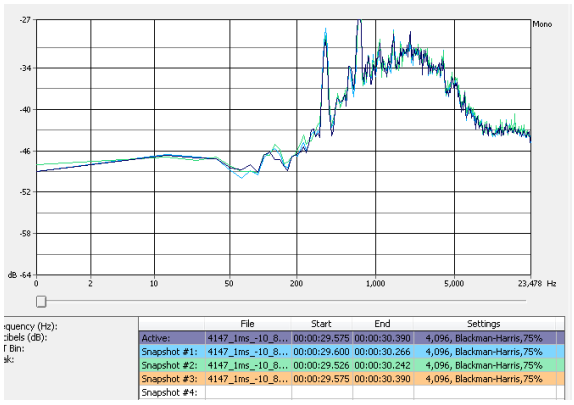


Figure 5.86: Disc 4147 - Spectrum of the 3 polyfit versions - high frequencies

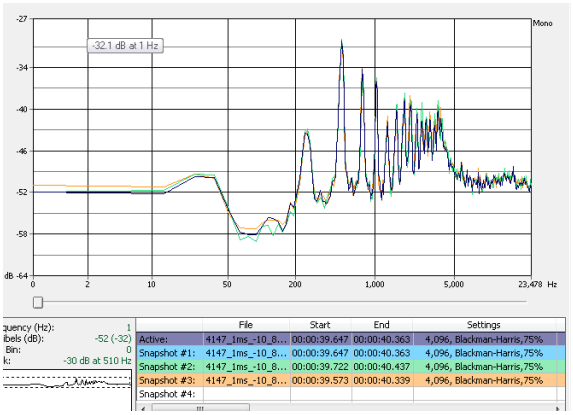


Figure 5.87: Disc 4147 - Spectrum of the 3 polyfit versions - low frequencies

Figure 5.86: the colors for each test are the following: test 1 = blue, test 2 = green, test 3 = black.

Figure 5.87: the colors for each test are the following: test 1 = black, test 2 = green, test 3 = yellow/orange.

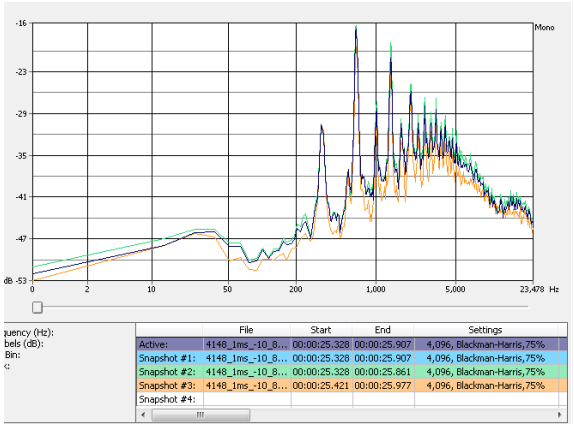


Figure 5.88: Disc 4148 - Spectrum of the 3 polyfit versions - high frequencies

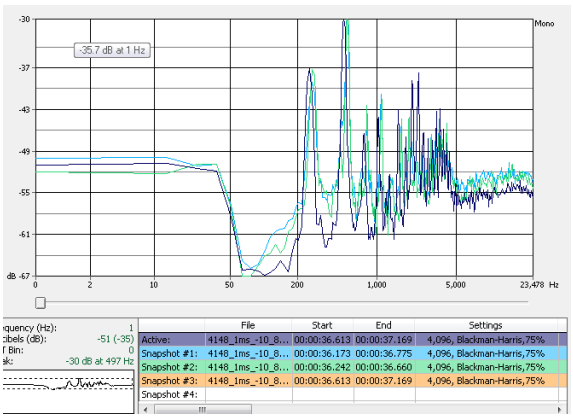


Figure 5.89: Disc 4148 - Spectrum of the 3 polyfit versions - low frequencies

Figure 5.88: the colors for each test are the following: test 1 = black, test 2 = green, test 3 = yellow/orange.

Figure 5.89: the colors for each test are the following: test 1 = blue, test 2 = green, test 3 = black.

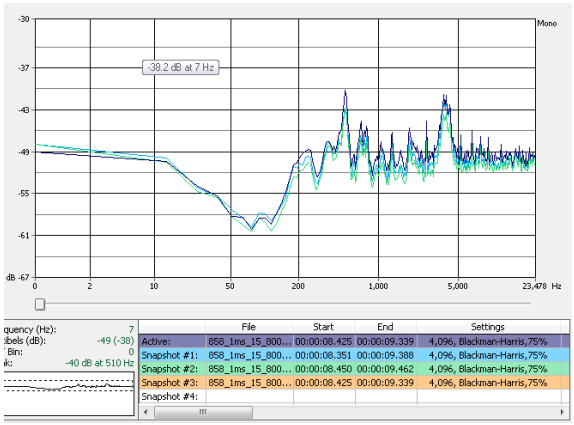


Figure 5.90: Disc 858 - Spectrum of the 3 polyfit versions - low frequencies

Figure 5.90: the colors for each test are the following: test 1 = blue, test 2 = green, test 3 = black.

Observations: By listening to the audio files, it was clear that the test 2 was the most successful. The noise was less disturbing. The test 2 has the value 80 for the parameter *sensitivity*. This means that more points were ignored and needed to be added with a spline. This spline has smoothed the final result a little bit. In the low frequencies, it can clearly be seen that the test 3(sensitivity 60) is noisier than the two others. The difference between a peak and a local minimum are more accentuated. However, the difference between 70 and 80 is not huge. In addition, the audio is still not good, especially in the high frequencies.

5.3.5 Test 2: Iterative polyfit with outlier

The plugin **Polyfit Outlier Blob** has two parameters that can be tested: *Max distance* and *Min points*. For this problem the five pair of values that have been tested are the following:

Table 5.3: Value of the parameters - 3 Tests for the iterative polynomial fitting idea

	Test 1	Test 2	Test 3	Test 4	Test 5
Max distance	1	1	0.5	0.2	0.25
Min points	10	20	20	10	10

In this test, the color always mean the same tests. Here are the meaning of each test for each each color: test 1 = Black, test 2 = Blue, test 3 = Green, test 4 = Yellow/Orange, test 5 = purple.

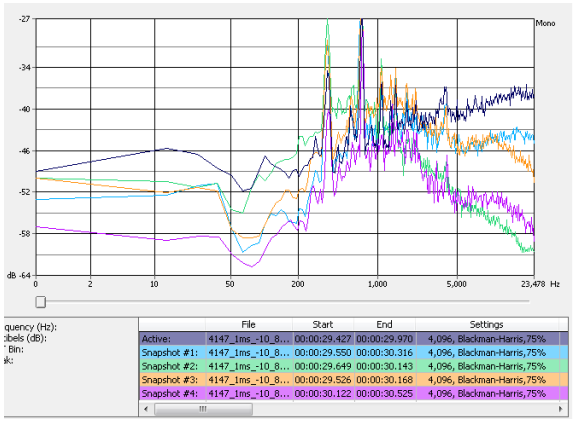


Figure 5.91: Disc 4147 - Spectrum of the 5 polyOut1 versions - high frequencies

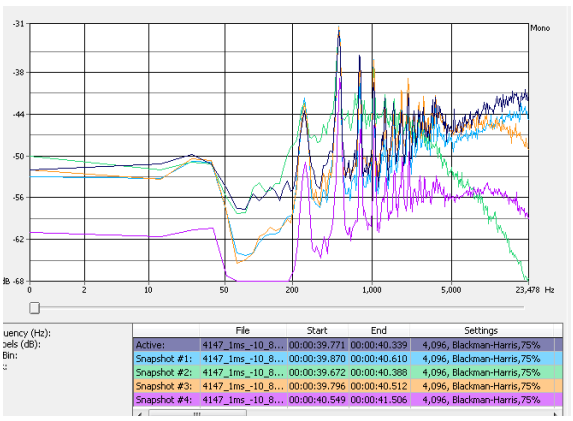


Figure 5.92: Disc 4147 - Spectrum of the 5 polyOut1 versions - low frequencies

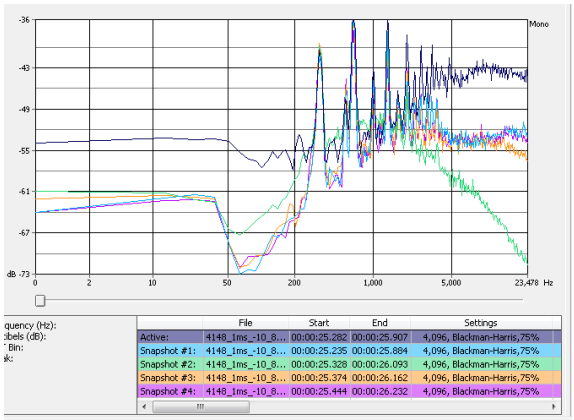


Figure 5.93: Disc 4148 - Spectrum of the 5 polyOut1 versions - high frequencies

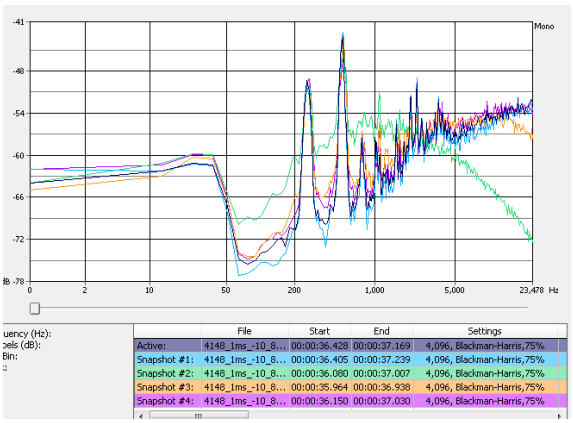


Figure 5.94: Disc 4148 - Spectrum of the 5 polyOut1 versions - low frequencies

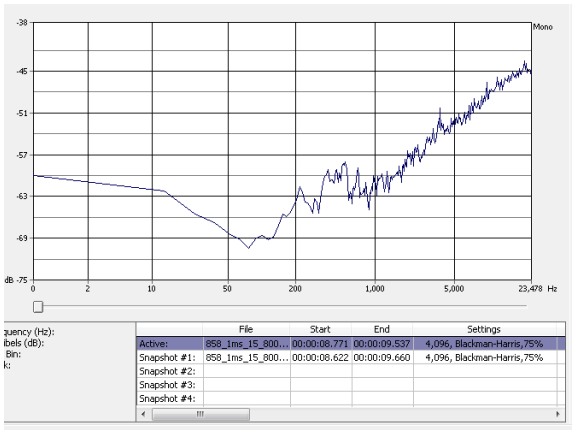


Figure 5.95: Disc 858 - Spectrum of the 5 polyOut1 versions - low frequencies

Observations: The parameters of test 3 were always very bad. This can be seen one the five figures 5.91,5.92, 5.93, 5.94 and 5.95 where the green function is falling down in the frequencies above 1kHz. The test 1 is the only version on which there are no roll-off in the high frequencies. By listening to the sound, the version 1 was always the best. For the disc 858, the other tests give results that could not be used for any comparison. For

the disc 858, the test 1 is the only test that give some results. The other four tests gave only bad results that could never be compared with the test 1. The verdict is that the parameters (Max distance=1 and Min points=10) give the best results.

After discussion with Carl and Earl, the results are not logical at all. The outlier version should not give bad results like the previous one. It might be that higher parameter than 1(for max distance) should have been used. It could also mean that an error in the implementation has screwed the results. Due to lack of time, further tests could not be carried out, but it would be worth investing in future projects.

5.3.6 Test 3: Polyfit over two lines with outlier

The plugin **Polyfit Outlier 2 Blob** has two parameters that can be tested: *Max distance* and *Min points*. For this problem the five pairs of values that have been tested are the following:

Table 5.4: Value of the parameters - 3 Tests for the iterative polynomial fitting idea

	Test 1	Test 2	Test 3	Test 4	Test 5
Max distance	1	1	0.5	0.2	0.2
Min points	10	20	20	10	10

In this test, the color always mean the same tests. Here are the meaning of each test for each each color: test 1 = Black, test 2 = Blue, test 3 = Green, test 4 = Yellow/Orange, test 5 = purple.

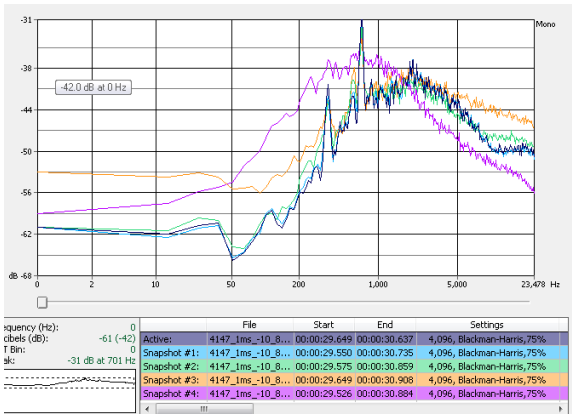


Figure 5.96: Disc 4147 - Spectrum of the 5 polyOut2 versions - high frequencies

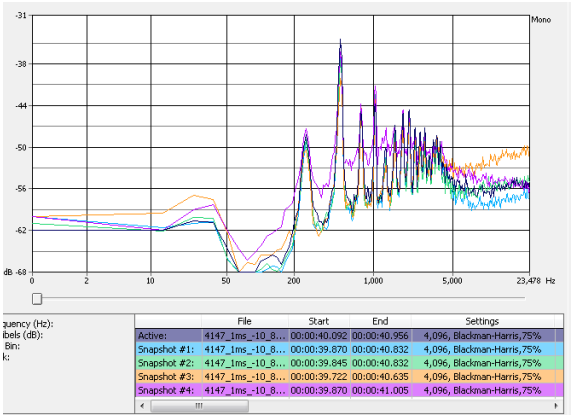


Figure 5.97: Disc 4147 - Spectrum of the 5 polyOut2 versions - low frequencies

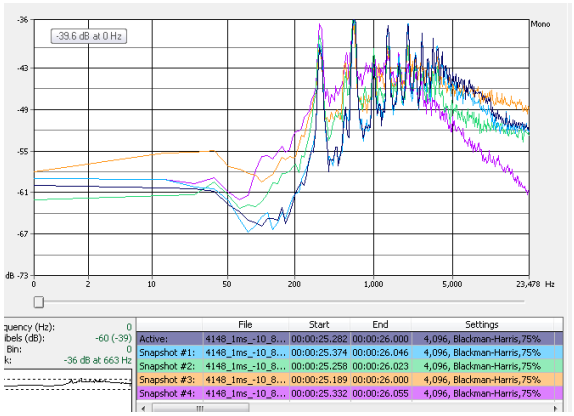


Figure 5.98: Disc 4148 - Spectrum of the 5 polyOut2 versions - high frequencies

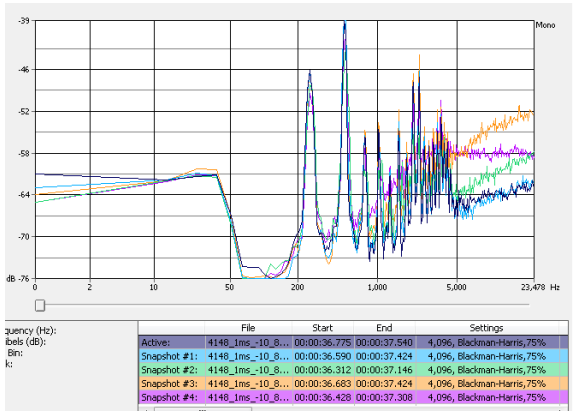


Figure 5.99: Disc 4148 - Spectrum of the 5 polyOut2 versions - low frequencies

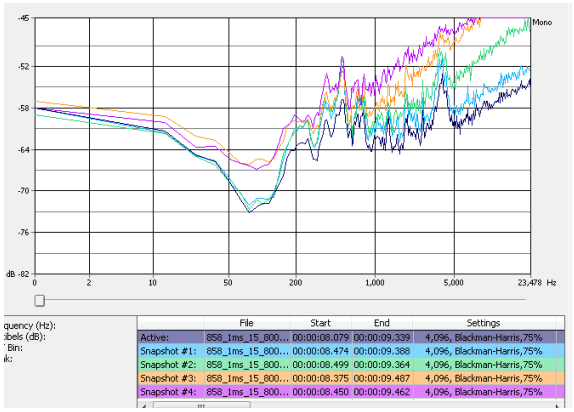


Figure 5.100: Disc 858 - Spectrum of the 5 polyOut2 versions - low frequencies

Observations: The tests 4 and 5 don't give any good results. The test 5 gives a lot of noise in the medium frequencies. In figure 5.98, it is possible to see a big roll-off in the high frequencies for the test 5. Indeed, the sound was very bad in the high frequencies for the test 5. By listening to the sound, the tests 4 and 5 were generally bad. The tests 1,2 and 3 were very similar with a slightly preference for the results of test 3. The test 3 was also better in the silence parts.

5.3.7 Test 4: 3D Polyfit over many lines

The plugin **3D PolyfitBlob** has one parameter to set up the number of lines. If this parameter is changed, the array below it will also be changed. This array makes it possible to change the weight for each line. Five tuples containing the weight for each line have been tested. The tuples are the following:

Table 5.5: Value of the parameters - 3 Tests for the iterative polynomial fitting idea

	Test 1	Test 2	Test 3	Test 4	Test 5
Number of lines	3	3	3	5	5
Weight Line -2	-	-	-	1	100
Weight Line -1	1	10	100	1	10
Weight Line 0	1	1	1	1	1
Weight Line 1	1	10	100	1	10
Weight Line 2	-	-	-	1	100

In this test, the color always mean the same tests. Here are the meaning of each test for each each color: test 1 = Black, test 2 = Blue, test 3 = Green, test 4 = Yellow/Orange, test 5 = purple.

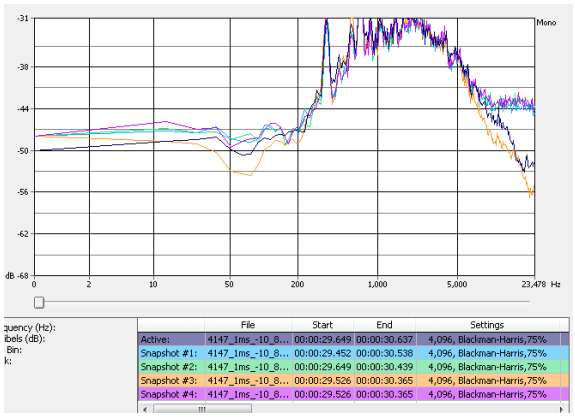


Figure 5.101: Disc 4147 - Spectrum of the 5 polyFit3D versions - high frequencies

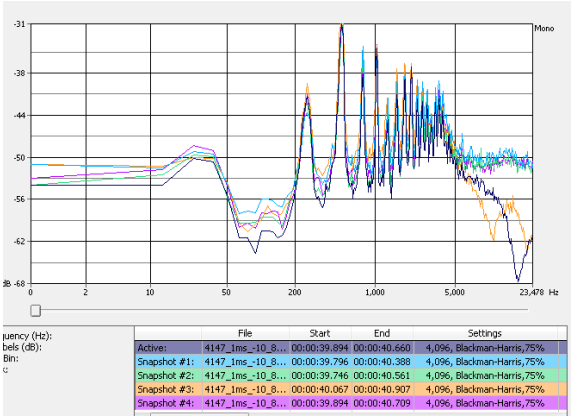


Figure 5.102: Disc 4147 - Spectrum of the 5 polyFit3D versions - low frequencies

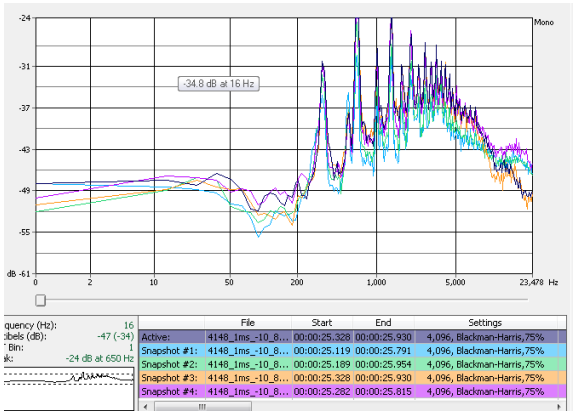


Figure 5.103: Disc 4148 - Spectrum of the 5 polyFit3D versions - high frequencies

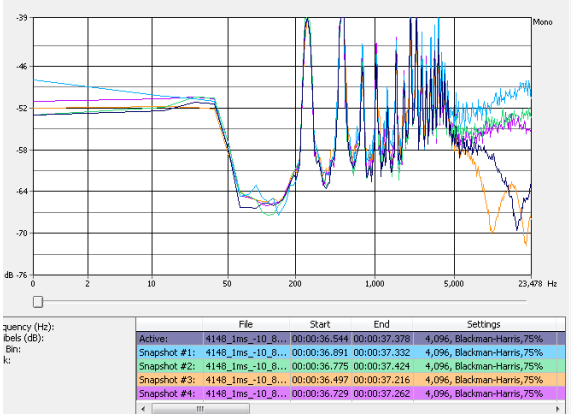


Figure 5.104: Disc 4148 - Spectrum of the 5 polyFit3D versions - low frequencies

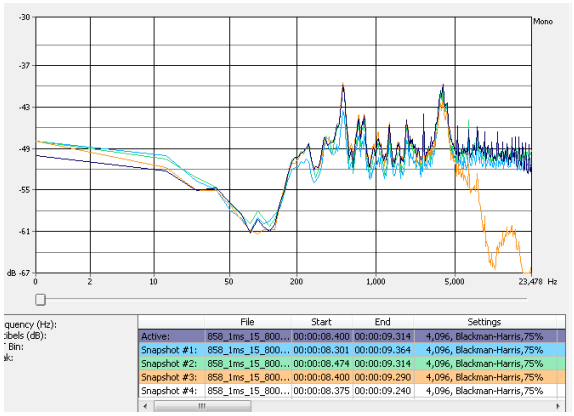


Figure 5.105: Disc 858 - Spectrum of the 5 polyFit3D versions - low frequencies

Observations: A first observation that can be made is that the tests 4 and 5 display a roll-off in the high frequencies (orange and black). That had to be excepted and has also been mentioned in the analysis phase. The high frequencies are dropping down, especially in the low frequencies. Another observation that can be made is that the test 3 gives similar results to a normal polynomial fitting and that can also be excepted. For this test, the weight of the preceding and following line are so big that they are almost ignored during the process. The algorithm therefore makes a fitting over one line. In a general way, the test 1 and 4 gave the best results but it was hard to say that the test 2 and 5 were really worst. The test 3 was clearly less good. It is also normal that the test 5 and 2 give similar results because the weight et for the second-preceding lines and second-following lines are really big(100). This test seemed to give the best results by comparison with the four previous tests.

5.3.8 Test 5: Change of the binning

During a discussion, it appears that part of the noise that can be heard is due to an error in the choice of the binning. Before this discussion, a binning of 40 has always been used. The intuition appeared because the noise in the sound was much more present in the high frequencies. If the binning is too big, the first tracking that is made on the binned image will not reflect the reality. Therefore, it required to change the value of the binning and to test again. Due to a lack of time, the tests have only be carried out on the disc 4147 and the conclusions will be reached by looking at these results.

Why a binning of 40 During the earliest development, the binning of 40 seemed to give good results and to not change the tracking to much. Because the discs have been scanned again and that the data is much more precise than before, the binning of 40 was not enough anymore. Unfortunately, it has been discovered towards the end of the project and the time did not allow anymore to make a lot of tests.

This tests will compare the four solutions with different binnings (5,10,20 and 40) together. Here again, the high frequencies and the low frequencies are separated. The parameters of the four solutions are the one that gave the best results for the test one to four (see corresponding section in the tests for more details).

Explanation of the colors: Unless, it is explicitly mentioned, the colors are always similar. Blue means a binning of 5, Green means a binning of 10. Yellow/Orange means a binning of 20 and Purple means a binning of 40.

Polynomial fitting

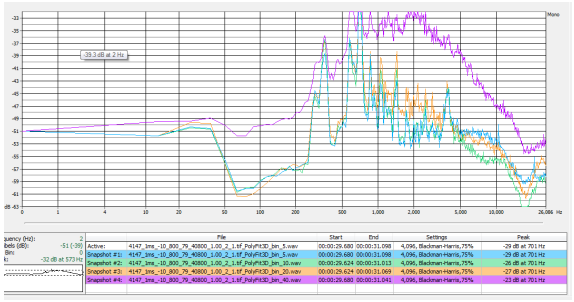


Figure 5.106: Disc 4147 - Spectrum of the 4 versions - high frequencies

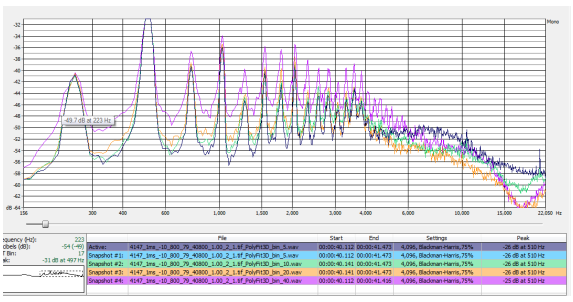


Figure 5.107: Disc 4147 - Spectrum of the 4 versions - low frequencies

Note: on figure 5.107, black means a binning of five instead of blue.

Iterative polynomial fitting with outlier

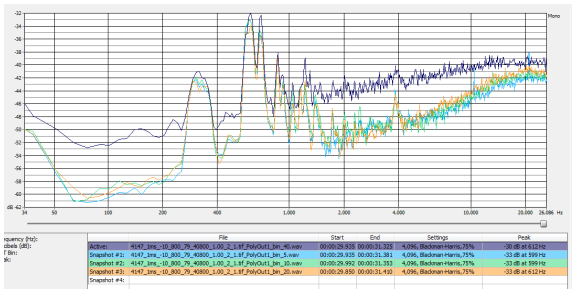


Figure 5.108: Disc 4147 - Spectrum of the 4 versions - high frequencies

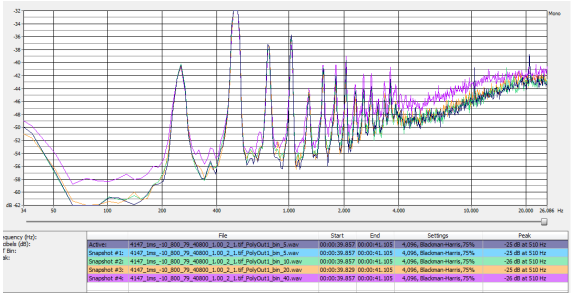


Figure 5.109: Disc 4147 - Spectrum of the 4 versions - low frequencies

Note: on figure 5.108, black means a binning of 40 instead of purple. on figure 5.109, black means a binning of 5 instead of blue.

Polynomial fitting by correlation with outlier

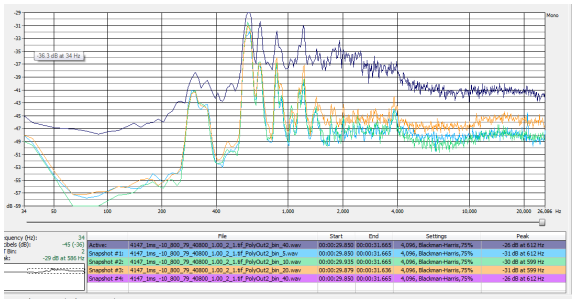


Figure 5.110: Disc 4147 - Spectrum of the 4 versions - high frequencies

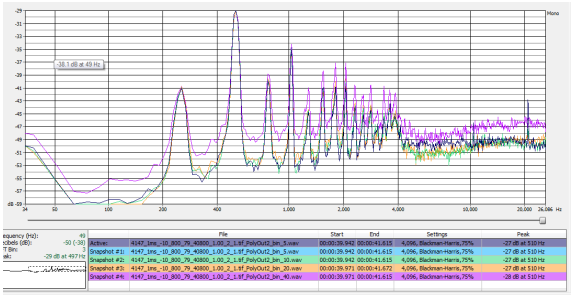


Figure 5.111: Disc 4147 - Spectrum of the 4 versions - low frequencies

Note: on figure 5.110, black means a binning of 40 instead of purple. on figure 5.111, black means a binning of 5 instead of blue.

3D polynomial fitting

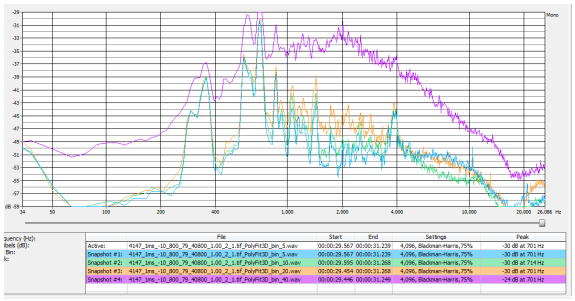


Figure 5.112: Disc 4147 - Spectrum of the 4 versions - high frequencies

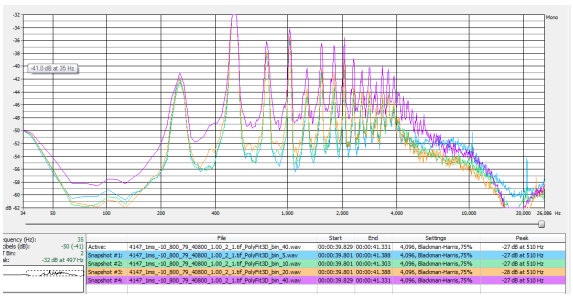


Figure 5.113: Disc 4147 - Spectrum of the 4 versions - low frequencies

Observations: A first observation that can be made is the difference of the binning of 40 in the high frequencies. The spectrum is shifted a lot even if it should not be. This is a clear factor explaining why the binning of 40 is worst. If there are more decibels, the noise is also more extreme. A second observation that can be made by listening to the audio files is that the binning of 10 is always better. This can not be well seen on the spectrums. The only thing that can be seen is the high variance in the middle frequencies for the other binning. However, this is hard to see.

5.3.9 Test 6: Binning of 10

The test 5 give the answer that the binning of 10 was the best binning. Than the question might be, which solution is the best solution?

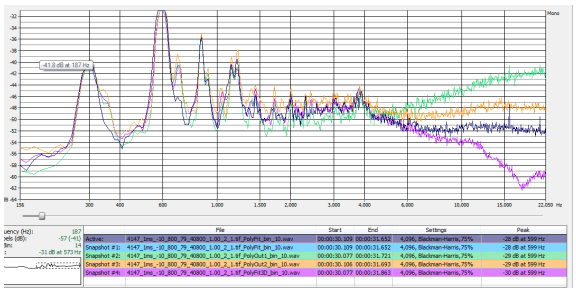


Figure 5.114: Disc 4147 - Spectrum of the 4 versions - high frequencies

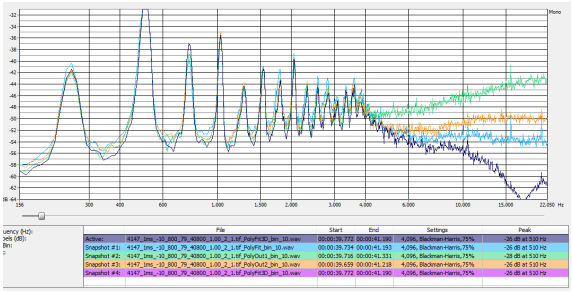


Figure 5.115: Disc 4147 - Spectrum of the 4 versions - low frequencies

Explanation of the color for each figure:
figure 5.114 Black = polyfit, Green = iterative polyfit, Yellow/Orange = polyfit with correlation with outlier, Purple = 3D polyfit.

Figure 5.115 Blue = polyfit, Green = iterative polyfit, Yellow/Orange = polyfit with correlation with outlier, Black = 3D polyfit.

Observations: A clear observation that can be made is that there is a split in the high frequencies (> 4kHz). The 3D polynomial fitting is as excepted the one that has the biggest roll-off. The iterative outlier version is the one with the biggest noise(Green). This verdict can also be made by listening to the sound, the iterative outlier version is not really good. The best version is the 3D polynomial fitting followed by the polynomial fitting. Another observation that can be made is that a binning of 10 is really better than the previous tests. With this version, the sound becomes pleasant to listen. There are no

huge differences between the high frequencies and the low frequencies. The only remaining problem is the presence of clicks and cracks in the sound. Future work should investigate the presence of these clicks.

5.3.10 Conclusion of the tests

The tests of the polynomial solutions have shown many things. The first thing is that a binning of 40 does not give any satisfactory results. The high frequencies are always too noisy. For the discs 858, the noise seems to be not too noisy. Unfortunately, more than half of the tests that have been presented in the present document have used a binning of 40. This has been discovered in the latest stage of the project. Changing the binning had the effect to obtain an audible sound. It is pleasant but still contains a lot of clicks and cracks. The time was too short to investigate the reasons of these clicks and cracks. One assumption that can be made is that the noise is not well detected but that cannot be true, the first tests proved it. The noise is detected but badly handled.

Another strange observation that the tests put forward is that the results given by iterative polynomial version do not make any sense. There is either an error in the implementation of the plugin "Polyfit Outlier Blob", or it would be necessary to test more. According to the IRENE team, something does not look good.

Another observation that has generally been made is that the 3D polynomial fitting is not bad but has a roll-off in the high frequencies. However, it seemed not to be a problem with the binning of 40. The test with the binning of 40 has shown that the 3D polynomial fitting is the most promising solution.

5.4 Parameters for each disc

This section describe the values that have been used for each parameter during these tests. These values are specific to the tiff files and changing them would not be useful. For some of these parameters, it would even make it worst.

Plugins / Discs	LoadTiff3D		Slope Correction		TrackDepth2	TrackWidth
	StartPass	EndPass	PassWidth	Slope	MinSpacing	Width
4147 1ms	1	20	384	8	80	40
4147 1.6ms	1	20	384	8	60	40
4147 3ms	1	20	192	12	40	20
4148 1ms	1	20	384	8	80	40
4148 1.6ms	1	20	384	8	80	40
68 1ms	1	20	384	-4	80	40
68 1.6ms	1	20	384	-4	80	40
70 1ms	1	20	384	4	80	40
70 1.6ms	1	20	384	4	80	40
858 1ms	1	20	384	8	80	40
858 1.6ms	1	20	384	8	80	40
7137 1ms	40	60	384	-10	80	40
7137 1.6ms	40	60	384	-10	80	40

5.5 Conclusion

The two tests parts have shown that the project has created a powerful solution. The noise is well detected. When one zooms into the original image to find the different scratches and dust particles, it is possible to see that these are the same points that have appeared after the noise detection. This noise detection is good but could still be improved. The inside of the big scratches are not always clearly detected. Fortunately, this could easily be solved by adding a new parameter to the plugin "ImageThreshold". This parameter would threshold over the values that are lower a certain value (example: $<$ -10). Indeed, the sub-flat image has the effect to contain values that are really low.

The four polynomial solutions have at the beginning of the tests shown that they were not satisfactory. However, a light has emerged which has shown that these solutions clearly deserve to be taken into consideration. A change in the binning process made it possible. This is a good warning for the future of the project, never take certain parameters for granted. Although this is easy to say, this is more difficult to achieve. There are more than 40 parameters in the entire process of the sound reproduction. Of course, the values of some parameters are easy to find as the width of the groove. But other parameters are very quickly less obvious.

Next chapter "Future work and improvement" will detail the possibilities that could be explored in future work as well as the improvements that could be made. The various improvements have already been presented in this chapter but will be approached from another angle: how to proceed?

Chapter 6

Future work and Improvements

This section presents six different improvements that could be made in order to improve the results. Some will be easy to do and some would be more challenging. Some might also give work for an entire project.

6.1 Noise detection

It has already been presented in the tests, but a small improvement could be made to the noise detection. Fortunately, this is very easy to do but due to a lack of time, this has not be done yet.

The noise detection perfectly detects the dust particles. This has been achieved by applying a threshold over the sub-flat image at the end of the universal shape process. This threshold is a value that filters all the points that are lighter than a certain value. The plugin "Image Threshold" has a parameter "threshold" that does that.

The scratches are also detected during the process. This has been achieved by looking at the derivative of the points. If the difference between two nearby points is high, it means that this is a scratch. Indeed, the plugin "Image Threshold" has a parameter "derivative Threshold" that does that. This parameter can be tuned in order to find the scratches. However, if this parameter is too low, it might be that the threshold also filters the grooves. Fortunately, this has been tested and a good parameter appeared to give very good results. So, this parameter can find the difference between two points. That means that the threshold plugins filter the border of the scratches that are really deep. The inside of the scratches are not filtered. The reason is that the scratches are very deep. After subtraction, they are even deeper. For the moment, the threshold does not allow to filter points that are lower than a certain value. This has been shown in the conclusion of the noise detection tests (see section 5.2.11 for more details regarding this phenomenon). Therefore, the plugin "Image threshold" should contain an additional parameter "Lower Threshold" that will filter the values lower than a certain value. In this way, it will be distinguished from the actual parameter "Threshold" that could be renamed "Higher Threshold".

6.2 Problem in the iterative outlier version

The iterative outlier version is an idea that consists of applying many fitting over one same line by removing the worst point after each iteration. This plugin allows a maximal number of iterations. If it is reached, the plugin uses the latest fitting and continues to the next point. The tests have shown that the results given by this implementation do not make any sense. This version should be better than the basic polynomial fitting. Two

hypotheses have been imagined. One hypothesis is that the implementation does not agree with the analysis and the design. It might be that there is a mistake. For instance, it could be that the algorithm does not stop after the given number of iterations. It could also be that the outlier parameter is not correctly used. The second hypothesis might be that the parameters that have been used during the tests are not good. The case of bad settings for the binning (40 instead of 10) is an example where a bad choice has given poor results. It is not excluded that this occurred during these tests. One should definitively investigate because this solution has high potential.

For instance, it could be good to test the iterative outlier version with different values for the parameter "Max Distance". Even more so now that the change of binning brings improvement.

6.3 Clicks and Cracks

Even with the best version that has been found during the tests, the sound is still not perfect. A hiss can still be heard in every part of the audio file: silence, high frequencies and low frequencies. It is not disturbing anymore like it used to be but it is still annoying. This hiss can be seen on the sound wave of the audio file. Figure 6.1 displays a silent part of the discs 4147 that have been processed with a binning of 40 and the 3D polynomial fitting solution.

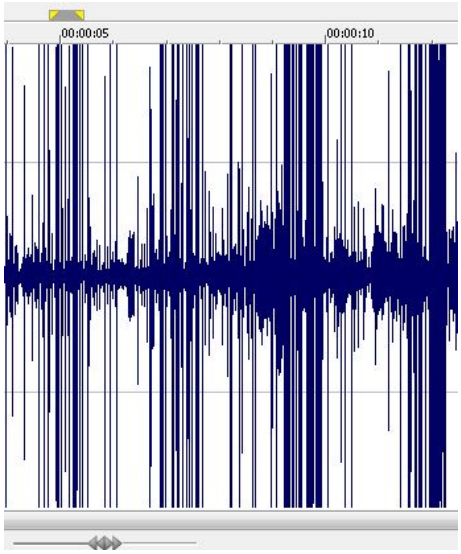


Figure 6.1: Sound wave of a silent part in the sound

The peaks that can be seen in figure 6.1 represents the clicks and cracks that have been mentioned below. These clicks and cracks can also be heard through a hiss. SoundForge give the user the possibility to remove them but that is not the idea. The idea would be to remove them during the process. Unfortunately, the time was too short to understand the reasons of these clicks and cracks. Is it due to noise that has not been found or is it due to the algorithms? It seems that the noise has been found. Therefore, there is something going on during the polynomial fitting process that deserves an investigation.

6.4 Quadratic fitting

For the moment, the polynomial fitting is always applied on the original image by using the original groove. This idea that has been proposed by Earl Cornell from Lawrence Berkeley

National Laboratory consists of forcing the shape of the groove to be polynomial and to be similar for each place of the groove. In other words, the a term of the following function would be forced and would be similar to different consecutive places of the groove. It does not have to be the same for each point on the entire disc. b and c would still be specific for each groove.

$$a * x^2 + b * x + c \tag{6.1}$$

A student of UC Berkeley is currently working on this idea. The good thing is that the result of this forcing could be used as input for the polynomial solutions.

6.5 Rough tracking

The entire process is based on a rough tracking that has been made. In this report, the trackDepth2 plugin has been used. This finds the middle of the groove by assuming that the deepest point is in the middle. This tracking is good but sometimes lead to points that are not in the middle of the groove. One solution, that has been developed by Earl Cornell, finds two points on each side of the groove at each tracking point. After that, the middle between these two points can be computed by using another plugin that has been implemented: *ComputeMiddle*. The problem is that some points are still not in the middle. If a point is not in the middle, the universal shape will be screwed by the shape based on this point. The polynomial fitting might also fails. A considerable improvement to the process would be to improve the rough tracking.

6.6 Change of sampling rate

The sampling rate for each disc that has been tested during this project is 40'000. This number has been chosen to ensure the minimum of 48 kHz that is required in order to play the audio again. In fact, this number is better than 48 kHz, it gives 56 kHz. The real number that the aluminum discs need is 96 kHz. Even if 56 kHz is good, a bigger number than 40'000 will give a higher frequency. In addition to getting closer to the original frequency, the increase of the sampling would have another impact. The high frequencies would have more data. For example, this number could be increased to 80'000 that gives us a frequency of 104 kHz. Moreover, the binning would less affect the wave of the sound, especially in the high frequencies.

Chapter 7

Conclusion

This chapter links the work that has been done with the goal formulated in the introduction. A personal conclusion is also presented.

7.1 Comparison with the objective

The goal of this project is achieved. The noise detection works fine. It consisted in following the tracking and computing the universal shape. A universal shape is created by averaging the values of many shapes. After that, the universal shape is placed at any place of the tracking. The result is a flat image which contains the groove without blobs and scratches. Therefore, it can be subtracted from the original image in order to find the blobs. The result is that the dust particles and the borders of the scratches are always detected. Sometimes, the inside of the scratches is not detected. Fortunately, a small change could be made to improve the noise detection. It means that the universal shape idea is a good idea.

Both the processing of the sound and the sound creation by polynomial fitting work. Four solutions have been found. The first idea, that is very basic, applies a polynomial fitting over the shape of the groove at each point of tracking. This second idea also applies a polynomial fitting but goes beyond. It tries to find out if some points can be removed because they are not close enough to the fitting. This second idea should have been an improvement but the tests have shown that something might not work properly. Applying a polynomial fitting over one line is good but ignores the fact that there are some correlations between two consecutive lines of the image. The third idea was to check if there is noise on both lines and if the next line does not present too many differences after being subtracted from the first line. This version was not bad but did not improve the sound. The last version that seems to work better consists in applying a polynomial fitting over many lines. Indeed, this improves the sound.

The sound that is produced by one of the different solutions is fine. There is a hiss that can be heard but is not disturbing. Moreover, the tests have shown that one single parameter can make a big difference. This proves that the solutions that have been implemented might give better results once parameters are adjusted. The results are encouraging. Improvements still need to be done in order to reach the quality of the stylus versions. Finally, the different solutions can be a good basis for future work.

7.2 Personal conclusion

I really enjoyed working on this project. There are two reasons for that. The first reason is that this project was very different from any projects I have worked on in the past. A

lot of mathematical notions had to be learned. Everything I heard from Carl and Earl was new to me and it was really exciting. The second reason is that I was able to produce a version that gave an audible sound. That is really encouraging because it could have been totally different.

However, I regret two things. One is that I did not had the time to reprocess some tests in order to better verify my solutions. I wish I would have had the time to understand the presence of clicks and cracks in my solutions.

I really appreciate having acquired a lot of knowledge that I can use in the future.

I am also aware that I am a lucky man. Indeed, I was lucky enough to be able to do my bachelor project in the United States at LBNL. I had the chance to see what it is like to work in a laboratory and to speak with incredible people. Everyone is really motivated and friendly. In addition, I had the chance to visit another country where the sun is always shining (if you ignore the fog of San Francisco).

7.3 Acknowledgments

This project would not have been possible without the help of professors of the HEIA-FR and the people of the IRENE-team.
I would like to thank particularly:

Dr. Carl Haber for giving me the opportunity to realize my bachelor project at the Lawrence Berkeley National Laboratory. His knowledge, his valuable advice and his remarks allowed me to conduct the project in the best conditions. I also received very good advice regarding the report and the presentation.

Earl Cornell for his precious time explaining how to scan the discs and how to process the TIFF files on the IRENE-plugin system. He has also given a lot of advice about the idea of the project. His comments regarding the report and the presentation were also really appreciated.

Frédéric Bapst for his supervision and his support and for the time he took to discuss the ideas of the project and the report. His precious comments that he gave were really helpful.

Nicolas Schroeter for the time he took to discuss the different ideas of the project. His advice concerning signal processing were also helpful for the understanding.

Noe Lutz for his comments, remarks and valuable advice in the middle of the project. Doing a polynomial fitting in 3 dimensions was really a good suggestion as it is the version that works best.

Finally, I would like to thank the College of Engineering and Architecture of Fribourg that gave me the opportunity to do my bachelor's work abroad. I can only recommend this experience to anyone and I hope that the school will continue to propose this opportunity to future students.

Glossary

RPM revolutions per minute

GUI graphical user interface

IDE integrated development environment

RMS root mean square

LBNL Lawrence Berkeley National Laboratory

Bibliography

[1] Alain Benninger. *2D and 3D Scanning of Metallic records and Masters*. Bachelor thesis, Haute école d'ingénierie et d'architecture de Fribourg, 2012.

[2] Romain Crausaz. *Bellrecords - Analysis of Historical Sound Recordings*. Bachelor thesis, Haute école d'ingénierie et d'architecture de Fribourg, 2013.

[3] Microsoft. Controlcollection class. `https://msdn.microsoft.com/en-us/library/system.web.ui.controlcollection(v=vs.110).aspx`. [Consulted: 2017-06-12].

[4] Visual paradigm. Media kit. `https://www.visual-paradigm.com/aboutus/mediakit.jsp`. [Consulted: 2017-04-16].

[5] G.Cowan (revised). 38.probability. Technical report, Lawrence Berkeley National Laboratory, 2015.

[6] G.Cowan (revised). 39. statistics. Technical report, Lawrence Berkeley National Laboratory, 2015.

[7] Ryan Seghers. C sharp cubic spline interpolation. `https://www.codeproject.com/Articles/560163/Csharp-Cubic-Spline-Interpolation`. [Consulted: 2017-07-02].

[8] Softexia. Sony sound forge pro. `https://www.softexia.com/wp-content/uploads/2015/01/SONY-Sound-Forge.png`. [Consulted: 2017-06-28].

[9] Irene Project Team. Source material audio. Technical report, Lawrence Berkeley National Laboratory, 2015.

[10] Wikipedia. Latex. `https://en.wikipedia.org/wiki/LaTeX`. [Consulted: 2017-06-28].

[11] Wikipedia. Linear interpolation. `https://en.wikipedia.org/wiki/Linear_interpolation`. [Consulted: 2017-06-28].

[12] Wikipedia. Microsoft project. `https://en.wikipedia.org/wiki/Microsoft_Project`. [Consulted: 2017-06-28].

[13] Wikipedia. Microsoft visio. `https://en.wikipedia.org/wiki/Microsoft_Visio`. [Consulted: 2017-06-28].

[14] Wikipedia. Microsoft visual studio. `https://en.wikipedia.org/wiki/Microsoft_Visual_Studio#/media/File:Visual_Studio_2017_logo_and_wordmark.svg`. [Consulted: 2017-06-28].

[15] Wikipedia. Polynomial interpolation. `https://en.wikipedia.org/wiki/Polynomial_interpolation`. [Consulted: 2017-06-28].

[16] Wikipedia. Runge's phenomenon. `https://en.wikipedia.org/wiki/Runge%27s_phenomenon`. [Consulted: 2017-06-28].

[17] Wikipedia. Spline interpolation. https://en.wikipedia.org/wiki/Spline_interpolation. [Consulted: 2017-06-28].

[18] WolframMathWorld. Least squares fitting. http://mathworld.wolfram.com/LeastSquaresFitting.html. [Consulted: 2017-08-01].

Declaration of good faith

I, Matthias Christe, declare under the penalty of perjury that the work performed for this project is my own. I did not copy or use anyone else's published or unpublished results other than those that are clearly stated and attributed to their rightful owners.

Used softwares and versions

The software used during this project are described in the following appendix.

SoundForge

SoundForge is a audio editing and analyzing tool developed by Magix Software GmbH. It can be used by professional and semi-professional experts. It was used to test the results and analyze with other audio files. It has helped to understand the noise problem. The version used for this project is Sound Forge Pro Version 10.0e.

Figure 7.1: Sound Forge Logo[8]

Microsoft Visual Studio

Visual Studio is integrated development environment (IDE) originally released in 1997. It allow to develop application for Microsoft Windows systems. Visual Studio was used to develop the Irene project as well as the contribution of this project. The version of Visual Studio used for this project is Express 2013.

Figure 7.2: Visual Studio Logo[14]

LaTeX

Latex, based on a macro-command language, is a tool for writing professional documents. The idea is to write the report and to worry about the visual result once it has been written. It was used to write the main report.

Figure 7.3: LaTeX Logo [14]

Visual Paradigm

Visual Paradigm is a program that allows you to create UML diagrams. The version used for this project is 13.1. This software was used during the analysis phase of the software development life cycle.

Figure 7.4: Visual Paradigm Logo [4]

Microsoft Office

Microsoft Office 2013 is an office suite of application that helps to write different documents. Among this suite, 2 softwares were used:

- **Microsoft Word** is a program for writing documents. It offers the possibility to visualize in real time the final result of the document. It was used to write the weekly minutes.
- **Microsoft Visio** made it possible to realize the large part of the schemes presented in the report.

Figure 7.5: Microsoft Word Logo[12]

Figure 7.6: Microsoft Visio Logo[13]