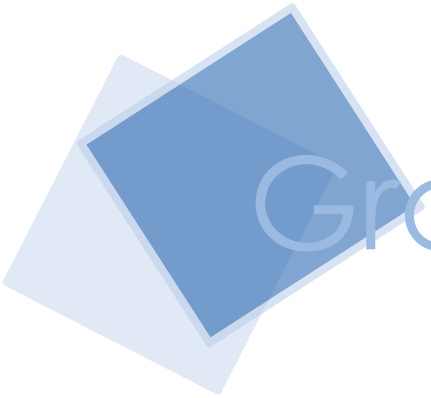

Group light-blue

Digital needle

Final report

2 June, 2003

by Patrik Olsson David Öhlin Robert Olofsson Cécile Ayrault Reine Vaerlien

Abstract

The project *"Digital Needle"*, which is described in this final report, is part of the course, 2E1366 Project Course in Signal Processing and Digital Communication.

The course was given in the Spring of 2003 by the department Signal, Sensors and Systems (S3) at the Royal Institute of Technology (KTH), Stockholm, Sweden.

The objective of the project is to create a fully automate system, which from a scanned images of a 78 rpm SP-record, can extract the present sound information. This means that from one or several readings of a record: image processing, track detection and finally sound enhancement should be performed without any human interaction.

In the final prototype, the program consists of an easy to use wizard that guides the user through the whole process, from image scanning to sound enhancement. When progressing through each phase a number of different parameters is adjustable e.g image file format, sampling frequency, various sound enhancement characteristics and number of anti-recording filters. All this to make the program adapt to user specific requirements as easily as possible.

Contents

1 Introduction

This project is part of the course, 2E1366 Project Course in Signal Processing and Digital Communication. The course was given in the Spring of 2003 by the department Signal, Sensors and Systems (S3) at the Royal Institute of Technology (KTH), Stockholm, Sweden.

1.1 Background

It is common knowledge that each time you play an old gramophone record, you also damage the vinyl grooves in which the pick-up needle runs. This causes the sound quality to slowly deteriorate.

There are a few solutions to this problem out on the market today. These products all have in common that they are fairly expensive, include highly sophisticated electrical components and utilizes advanced methods such as laser tracking.

With the equipment used in this project it will not be possible to use the program on LP- or EP-records, this because of the limited resolution capabilities of the scanner, which makes it near impossible to distinguish the track variations.

1.2 Objective

The objective is to extract the sound information from a scanned image of an SP-record. The final system shall be fully automate, meaning that from one or several readings of a record, image processing, trace detection and following, and finally sound enhancement should be performed without any human interaction.

1.3 Specifications

In the final implementation the image processing can be performed both on the host computer and the DSP board. This is due to the large image data received when extracting the images[1]. The sound enhancement and sound output on the other hand should be performed on the DSP alone.

The sound quality of the final sound output should be equal to or better than the online example [12].

1.4 Equipment

This is the equipment used for the project:

1.4.1 Hardware

- One Dell Intel OptiPlex GX400, 1.7 GHz, 512Mb RAM
- Two Dell Precision 420, Pentium III, 512Mb RAM
- One DSP-card Texas Instruments C6701 EVM
- One loudspeaker Fostex 6301B
- One scanner CanoScan 5000F 2400dpi
- A number of 78 rpm SP-records.

[1] 2400 dpi, for scanner type see 1.4.1.

1.4.2 Software

- MATLAB, version 6.5 Release 13
- Microsoft Windows 2000 Professional, version 5.0
- Microsoft Office Package
- Microsoft Visual C++, version 6.0
- Adobe Illustrator 10
- Adobe Acrobat 5.0
- Adobe Photoshop 5.0
- Code Composer Studio, version 2.10
- GoldWave 4.26
- LaTeX 2_ε [13]

2 General Premises

As mentioned earlier in section 1.1 the most constraining factor on the performance of final product is the resolution of the scanner. It will not only make it impossible to implement the program on LP- and EP-records, but also limit the image quality and thereby the bit-resolution of the sound for the SP-record.

But one must also consider, that one of the points about creating this kind of program is that you want it to work on the average desk top. So even if it were possible to get higher resolution, the result would be very large images for the computer to handle and store. This is not feasible for most users.

Since the objective is to have the DSP-card take care of the sound processing and the sound output in real-time, another constraining factor is the processing speed. This sets the boundaries for the amount of sound enhancement that can be achieved, given of course that the code is optimized to a certain extent.

3 Scanning Procedure

As input to the program, scanned images are needed. The process of retrieving the images are discussed below. The chapter includes *Multiple Images*, which describes the need for several images instead of one, and *Image-to-Noise Correlation*, a description of the relationship between the noise and the images.

3.1 Multiple Images

Often when using an ordinary scanner one is limited to the size and the resolution of the scanner. Most scanners, for natural reasons, have a width which have been adjusted to the width of an A4 paper size. Since the width of the average SP-record extends beyond this measurement, it is concluded that more than one scan of the record will be necessary in order to be able to read out a whole record.

The first thing that comes to mind is to scan half the record first and then rotate it 180° to scan the other half. The problem with this approach is that the scanner scans from the top with a fluorescent light positioned horizontally. For the record this means that tracks aligned with the light will have a much better resolution than orthogonal ones.

This can be avoided by placing the record in the middle of the scanner, instead of halfway out, and leaving out the edges. This produces two images with high resolution on all parts of the record, since the 180°-rotation. However, tests have shown that the bottom part of the record does not get the same quality in resolution as the top and therefore the only useful part is the top quarter, see fig. 1.

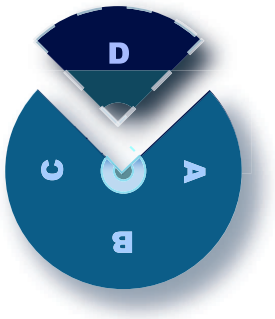

Figure 1: A record divided into 4 quarters with the top one deemed useful and extracted

The conclusion from this argument is that to be able to automatically read out useful information from the record at least 4 scans have to be made.

3.2 Image-to-Noise Correlation

When using scanned images, retrieved by the method in section 3.1, some noise will be added to the signal.

3.2.1 System noise

One source contributing to this noise is of course the bit/pixel-resolution; higher resolution gives better ability to follow the variations in the tracks, which contains the sound information. Since nothing can be done about removing this noise, other than scan with a higher resolution and better scanner, it will be referred to as system noise.

3.2.2 Process noise

Another contributing factor to the noise is the fact that there will be differences even within each of the extracted sections, see fig. 1. The tracks will be decreasing in resolution towards the edges. The noise emanating from this fact will be referred to as process noise.

The process noise has characteristics that relate to the number of images used and, as mentioned later in section 4.2 of this report, the effects of the image processing.

3.2.3 Correlation to Images

The process noise will be non-stationary and colored, but also feature a periodicity. The notation periodicity is a bit misleading since there is no actual period in a noise signal. The term in this case refers to the volume and color of the noise which exhibit periodic behavior.

The periodicity in the noise, clearly perceived when listening to the unfiltered raw data, is believed to come from the number of images used. If using 8 or 16 scans instead of the minimum of 4 scans, the result is a halved respectively a quartered period, which supports the theory.
On the other hand the noise should diminish, at least in the mean-power sense, which is either not the case or unnoticeable.

4 Image Processing

This chapter depicts probably the most crucial part in the automatization process of the program, getting standardized and easy-to-read images. The sections and their objectives included are *Orientation* - to find a fixed point reference in all images, *Transformation* - to transform the images to a more easy read form, *Cropping* - to cut away excess image material and *Formats* - to deal with the general specifics of the images.

4.1 Orientation - Finding the Center of the Record

Given an image of the upper half of the record, there exists several ways of finding the center of the record. If the border of the record can be located in the image, the center of the record can easily be found by fitting a circle to this border. If you know this circle, the rectangular to polar transformation can easily be deduced. By locating the border and then using these coordinates to fit a circle to these, the center is known.

4.1.1 Locate the border of the record

To be able to fit a circle to the border of the record the border has to be found. The coordinates of the border was acquired by searching down the columns of the image matrix, until the pixel gray-scale value reaches below a certain threshold.

The threshold was chosen based on empirical knowledge of typical values. Instead of taking advantage of all given data, that is, the whole upper border of the record, only a fraction of the points is used, e.g every hundred. This, of course, leads to noticeably less precision in the localization of the center. However, there is no reason to suspect that this would lead to any greater, negative impact on the performance of the whole system.

4.1.2 Fitting a circle to data

Given N coordinates (x_i, y_i) they all will lie on the border of the same circle if they satisfy the following equation

$$(x_i - x_c)^2 + (y_i - y_c)^2 = r^2 \quad i \in 1, 2, \ldots, N$$

for the three unknowns x_c, y_c and r where x_c and y_c represents the center of the record. The equation above can be rewritten as follows

$$\begin{aligned} x_i^2 - 2x_i x_c + x_c^2 + y_i^2 - 2y_i y_c + y_c^2 &= r^2 \\ \underbrace{r^2 - x_c^2 - y_c^2}_{a_1} + \underbrace{2x_c}_{a_2} x_i + \underbrace{2y_c}_{a_3} y_i &= x_i^2 + y_i^2 \end{aligned}$$

and substituting $a_1 = r^2 - x_c^2 - y_c^2$, $a_2 = 2x_c$ and $a_3 = 2y_c$ we get

$$a_1 + x_i a_2 + y_i a_3 = x_i^2 + y_i^2$$

which can be written in matrix form as

$$\begin{pmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ \vdots & \vdots & \vdots \\ 1 & x_N & y_N \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} x_1^2 + y_1^2 \\ x_2^2 + y_2^2 \\ \vdots \\ x_N^2 + y_N^2 \end{pmatrix}$$

with

$$\mathbf{H} = \begin{pmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ \vdots & \vdots & \vdots \\ 1 & x_N & y_N \end{pmatrix}, \mathbf{a} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}, \mathbf{h} = \begin{pmatrix} x_1^2 + y_1^2 \\ x_2^2 + y_2^2 \\ \vdots \\ x_N^2 + y_N^2 \end{pmatrix}$$

this can be written more compactly as

$$\mathbf{Ha} = \mathbf{h}$$

if there is more than three coordinates the system is over-determined, and the equality won't hold. The least squares solution can be found by solving the normal equations

$$\mathbf{H}^T\mathbf{Ha} = \mathbf{H}^T\mathbf{h}$$

which will come down to the following system of equations

$$\begin{pmatrix} N & \sum_{i=1}^N x_i & \sum_{i=1}^N y_i \\ \sum_{i=1}^N x_i & \sum_{i=1}^N x_i^2 & \sum_{i=1}^N x_i y_i \\ \sum_{i=1}^N y_i & \sum_{i=1}^N x_i y_i & \sum_{i=1}^N y_i^2 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^N (x_i^2 + y_i^2) \\ \sum_{i=1}^N x_i (x_i^2 + y_i^2) \\ \sum_{i=1}^N y_i (x_i^2 + y_i^2) \end{pmatrix}$$

this system of equations was solved by Gauss-Jordan elimination which gives

$$a = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{h}$$

after having found a_1, a_2 and a_3 the parameters of the circle are

$$\begin{cases} x_c = a_2/2 \\ y_c = a_3/2 \\ r = \sqrt{a_3 + x_c^2 + y_c^2} \end{cases}$$

These will be used in the rectangular to polar transformation.

4.2 Transformation - Rectangular to Polar Coordinates

The transformation from rectangular to polar coordinates uses the parameters of the circle found in section 4.1.

Call the center coordinates of the circle, (x_c, y_c), and the radius, r_c. This will be the center and the radius of the *record* in the image. It is that *record* that should be "straighten" out so that the relationship between a point $v(x, y)$ in the original image and a point $u(r, \theta)$ in the transformed image is:

$$u(r, \theta) = v(x_c + r\cos(\theta), y_c + r\sin(\theta))$$

Thus the value from the point $(x_c + r\cos(\theta), y_c + r\sin(\theta))$ in the original image should be placed at the point (r, θ) in the transformed image.
Eventually the transformed image will be a matrix[2] like this

$$T_{mn} = \begin{pmatrix} u(r(1),\theta(1)) & u(r(1),\theta(2)) & \cdots & u(r(1),\theta(N)) \\ u(r(2),\theta(1)) & u(r(2),\theta(2)) & \cdots & u(r(2),\theta(N)) \\ \vdots & & \ddots & \vdots \\ u(r(K),\theta(1)) & u(r(K),\theta(2)) & \cdots & u(r(K),\theta(N)) \end{pmatrix}$$

where $r(k) \in \{0, 1, 2, \ldots, r_c\}$ and $\theta(k) \in \{\frac{\pi}{4}, \frac{\pi}{4} + \delta, \frac{\pi}{4} + 2\delta, \ldots, \frac{3\pi}{4}\}$. The stepsize δ is given an appropriate value, this will determine the width of the transformed image and r_c the height, as can be seen in figure 2b).

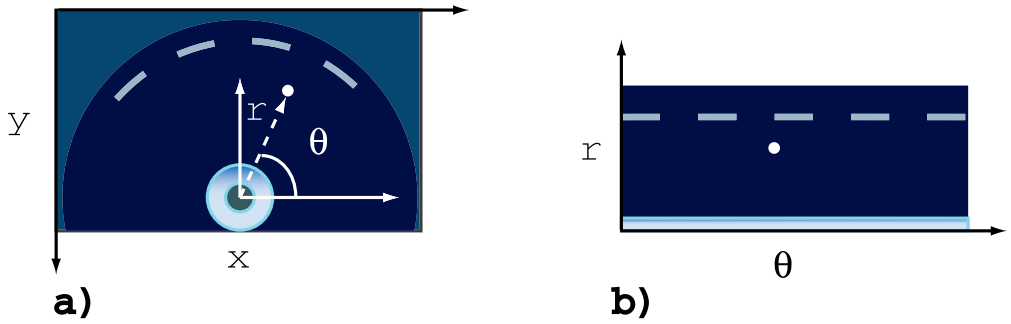

Figure 2: **a)** Original(scanned) image with a point at (r, θ). **b)** The transformed image (polar coordinates) and the transformed point.

Now the matrix element $T_{(r,\theta)}$ will contain the point at radius r making the angle θ in a polar coordinate system centered at (x_c, y_c) in the original image.

[2] se definition of image matrix.

This means that the tracks will be running in straight lines from left to right, but due to unprecise localization of the center (x_c, y_c) the transformation might not be perfect.
However, if the center is not found in the proximity (within 200 pixels $\approx$ 2 mm) of the true value, this will cause problems in the rest of the image processing. Fortunately this is very rarely the case.

4.3 Cropping

After having been converted to polar coordinates, the image contains a large amount of redundant data. In particular, the label part of the record, which accounts for almost one third of the radius, can be removed, as well as the outermost border.

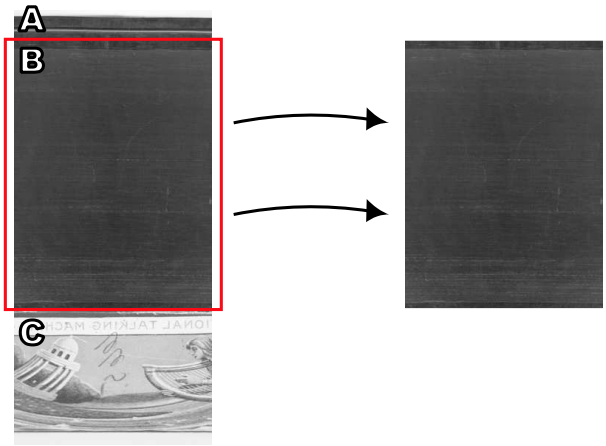

Figure 3: The cropping procedure cutting away excess material.

4.4 Saving & Formats

The images provided to the program are very big. To save hard drive space, the images may be compressed before they are processed.
Lossy compression schemes, such as JPEG, provide much higher compression rates, but, since they are destructive, they will affect the output sound quality negatively.

To accommodate to the different needs of the user, three input image formats are supported.

DIB - Uncompressed Microsoft Windows & reg; device independent bitmap

PNG - Lossless, open format, based upon Lempel-Ziv and Huffman coding

JPEG - Lossy, industry standard by the Joint Photographic Experts Group

A single scan of an SP record, at 2400 dpi, is typically larger than 200 Megapixels. If saved without compression, e.g. in the DIB format, it will be larger than 200 MB.

Without loss, the file size can easily be halved by using the PNG format. This is still a lot of data to handle, and the compression is time consuming. If some loss is acceptable, JPEG compression can be used instead. Then, compression rates of 1/10 can easily be achieved without noticeable loss of quality. The same formats are used when saving the processed images to file.

The PNG format and the JPEG compression were chosen because there are free libraries

5 Sound Extraction

After the images have been processed (see Chapter 4), it is time to start the "read-out" of information. The sound extraction described in this chapter can be divided into two parts: *Track Localization*, which finds the tracks, and *Track Following*, which runs through the tracks.

5.1 Track Localization

Since the tracks of the record runs through several images a method has to be found, which keeps tabs on where the tracks are located in the images.

5.1.1 Theory

The algorithm developed locates all the tracks on the left-hand side of the images and save the indices to these points, see fig. 4.

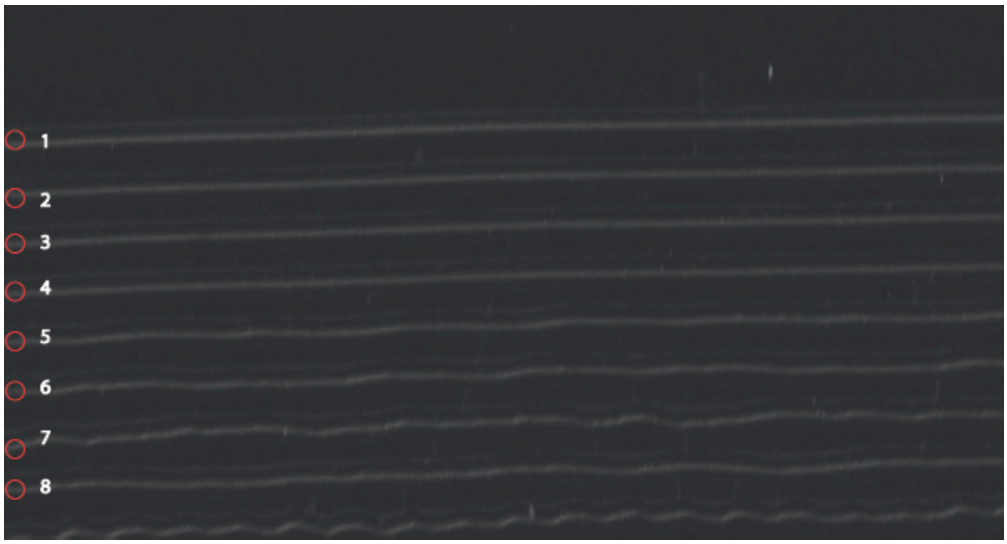

Figure 4: Indexing of starting points in an image.

The reason for this approach is that, when following a track to the border of one image you can't simply just continue at the same position in the next, because of the inter-alignment of the images is not the same. What this means is that i.e track two doesn't end in image one and start in image two at the same y-coordinate.

Another problem the algorithm had to solve was the separation between the true tracks and false positives, something erroneously indexed as a true track, since incorrect indexing would mess up the reading.

5.1.2 Algorithm

To have a good representation to work with all images are stored as matrices[3] $I(h, w)$,

$$I(h,w) = \begin{pmatrix} I(1,1) & \cdots & I(1,w) \\ \vdots & \ddots & \vdots \\ I(h,1) & \cdots & I(h,w) \end{pmatrix} \quad I(h,w) \in \{0,1,\ldots,255\} \tag{1}$$

, where h corresponds to the height of the image and w the width. Each element will then represent a pixel in the image and the value of that element is the grey-value of that pixel.

This way the pixel with coordinates[4] (x, y) is stored in the matrix I as $I(y, x)$, thus the rows correspond to the y-axis and the columns to the x-axis.

To be able to find the starting points of the tracks the algorithm utilizes the fact that the tracks produce much brighter pixels than the rest of the record. So summing over the image in the x-direction should produce prominent peaks in the y-coordinates that contains a track.

The summation is performed over 128 columns of the image matrix 1 as follows:

$$S(h) = \begin{pmatrix} I(1,1)+I(1,2)+\cdots+I(1,128) \\ I(2,1)+I(2,2)+\cdots+I(2,128) \\ I(3,1)+I(3,2)+\cdots+I(3,128) \\ \vdots \\ I(h,1)+I(h,2)+\cdots+I(h,128) \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{128} I(1,i) \\ \sum_{i=1}^{128} I(2,i) \\ \sum_{i=1}^{128} I(3,i) \\ \vdots \\ \sum_{i=1}^{128} I(h,i) \end{pmatrix} \tag{2}$$

When performing this summation for an image of height 1500 and subtracting the mean,

$$S(i) = S(i) - \overline{S}$$

the result can be seen in fig. 5 and shows that the peaks formed by the tracks are clearly identifiable.

After the summation 2, the algorithm steps through the vector S and searches for the peaks.
What is classified as a peak is very dependent on the surrounding of the classical problem of global/local minima. So a comparison with the neighboring values is therefor necessary.

In order to find a correct peak to start with, a skip of N elements into the vector S and a search for a maximum in a subsection of 60 elements is performed. Call this peak p_1.

After having found this peak, another search, commencing at the point $N - 14$, is performed. Although, this time searching for a valley in a section of 14 elements.

$$v_1 = \min_{i \in \{(N-k\cdot 14),(N-k\cdot 14)-1,\cdots,(N-k\cdot 14)-14\}} S(i)$$

then the search continues for a maximum N steps ahead

$$p_1 = \max_{i \in \{(N-k\cdot 14),(N-k\cdot 14)-1,\cdots,(N-k\cdot 14)-14\}} S(i)$$

Continuing in this fashion, alternating between maxima and minima and increasing k by 1 for each step, we will search in the grey areas shown in fig. 5. In searching for the peaks, a monitor keeps track of the mean of the previous n peaks and valleys

[3] Although the actual implementation in c/c++ is different the theory is based on this notation.

[4] The coordinate system have the origin in the upper left corner with the positive x-axis running to the right and positive y-axis downward

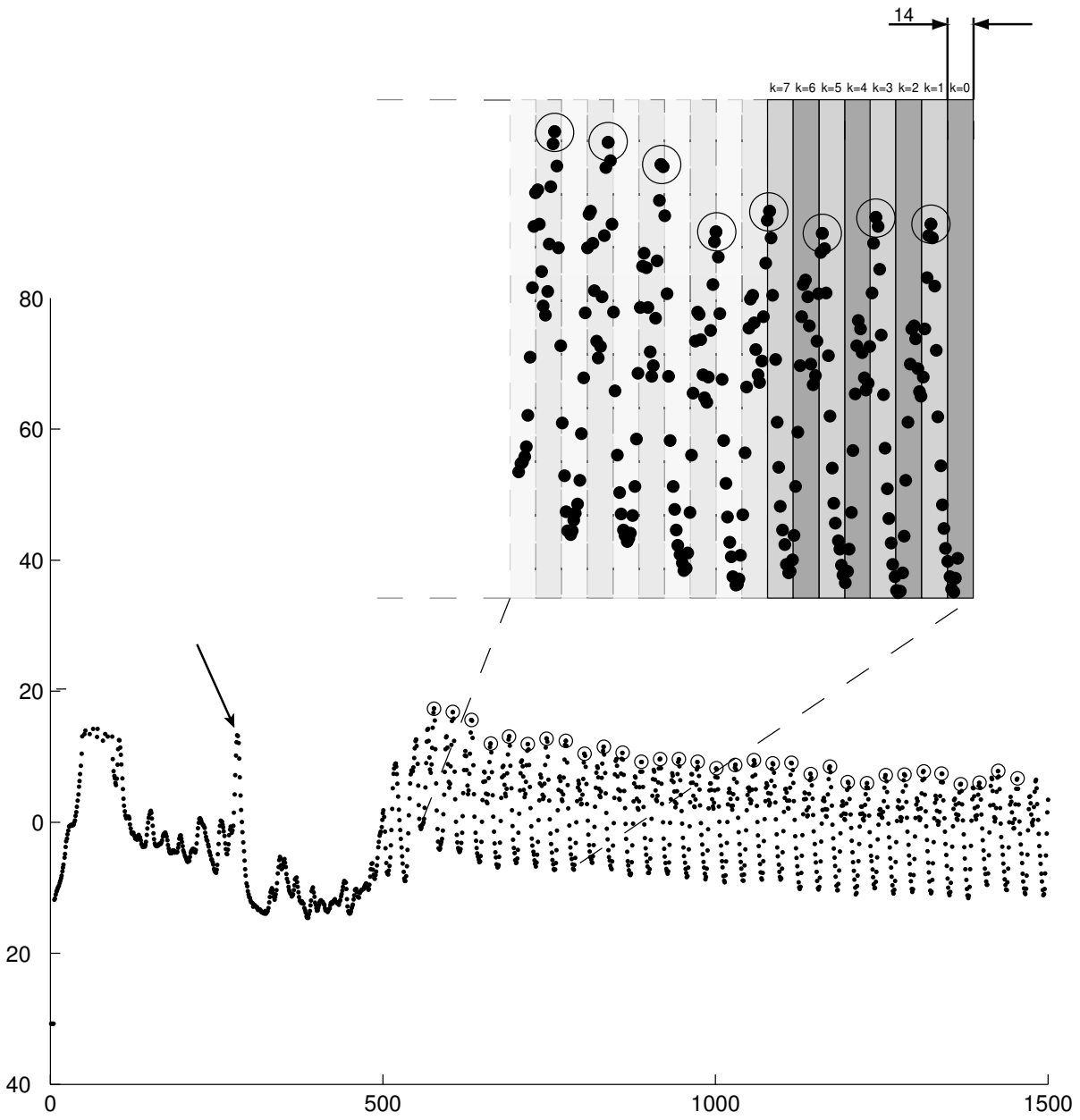

Figure 5: Peaks and valleys formed by summation of image along x-direction, vector S(h). In the bright areas a peak is searched for and in the dark a valley. Found peaks are indicated by circles.

$$\overline{p} = \frac{1}{n} \cdot \sum_{i=1}^{n} p_n$$
$$\overline{v} = \frac{1}{n} \cdot \sum_{i=1}^{n} v_n$$

If a new peak is found which has a value that is lower than $\frac{1}{2}(\overline{p} + \overline{v})$ the search is stopped. This because of the last peak in the dense area where we have all the tracks, from 500 to 1500 in fig. 5, now have been isolated.

Having found the last peak p_{last} in the dense area the first peak above this one needs to be found as this corresponds to the first track.

To find this, the rest of the vector $S(0, 1, \ldots, p_{last} - 1)$ is searched through.

The searching is performed as follows for n=$p_{last} - 1, \ldots, 1, 0$:

repeat
  if $S(p_{last} - 1 - n) > \frac{1}{2}(\overline{p} + \overline{u})$ **then**
    $p_{first} \leftarrow p_{last} - 1 - n$.
    break, we have found the first peak.
  else
    $\overline{u} \leftarrow \frac{15}{4}(\overline{p} + S(p_{last} - 1 - n))$
  end if
until n=0

now the index to the first peak can be found in p_{first}, this peak is marked with an arrow in figure 5.

What is left to do now is to search through the vector $S(i)$ for $i = (N + 1, N + 2, \ldots,$ height of image) to find the rest of the peaks. This is performed analogously to what was describe in the beginning of this section.

5.1.3 Robustness

The most sensitive and crucial part of the algorithm is the finding of starting points or indexing. If this is not done correctly the whole sound signal will be useless.
After thorough testing the algorithm proved to work very satisfactory for records with reasonable amounts of scratches and dents i.e well preserved.

Primarily the algorithm was made to be able to follow the tracks through the different images, a side effect was that the algorithm became more robust when it came to interaction between images.

An examination of what happened if the algorithm "derailed" in the middle of one image, showed that the error would be corrected when starting again in a new image. This due to the fact that even if reaching the end of one image in the *wrong* place, the starting point in the new image will have the *correct* position if indexed properly.

So if considering that the length of one image only corresponds to approximately one fifth of a second (using 4 scanned images), this "derailing" will not be particularly disturbing in the actual music if occurring only occasionally.

5.2 Track Following

The tracks in the images are created by the groove walls which reflect the light from the scanner and create intensity values in the image of the record, see fig. 6. The figure shows how a typical track on a

record might look like in the image.

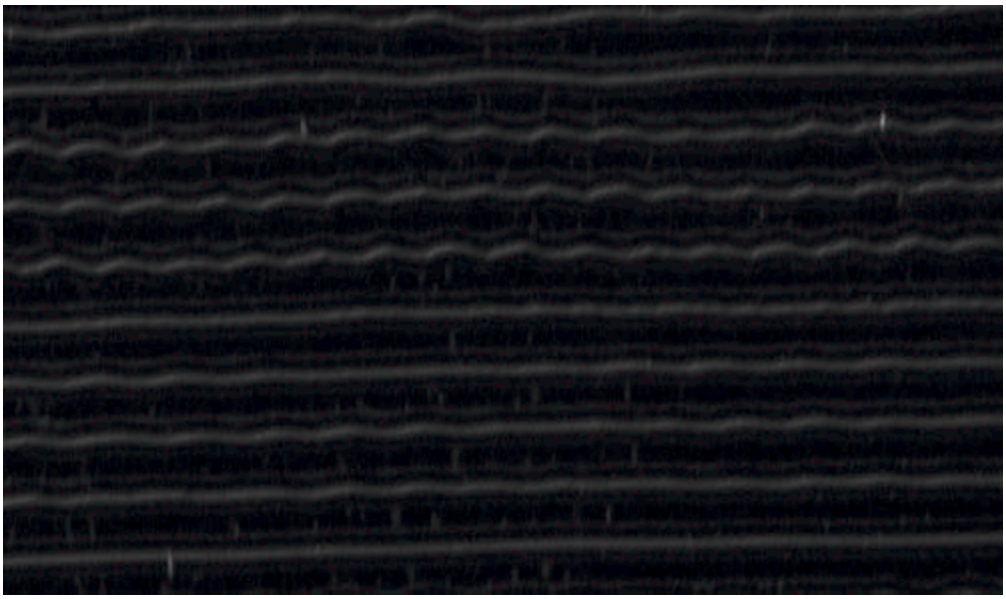

Figure 6: Closeup of tracks.

5.2.1 Theory

It is imperative that the algorithm is designed in such a way that it follows, in simple terms, those parts that a human would perceive as a track.

From the images it's quite obvious what constitutes a track and what doesn't, the hardest thing is to transfer this *knowledge* to the algorithm in such a way that it won't become unnecessarily complex.

5.2.2 Algorithm

When having a typical track as depicted in fig. 6, the main problem is to read the track from left to right in the image.

Initialization:
When extracting the wavy lines containing sound information from the image, the indexed starting points of each track are used to initialize the tracking.

A column vector of five pixels centered around the starting point y_1 in the image is extracted, see figure 7.

$$P = \begin{pmatrix} M(y_1 + r(0), c) \\ M(y_1 + r(1), c) \\ M(y_1 + r(2), c) \\ M(y_1 + r(3), c) \\ M(y_1 + r(4), c) \end{pmatrix} = \begin{pmatrix} M(y_1 - 2, 1) \\ M(y_1 - 1, 1) \\ M(y_1 \quad , 1) \\ M(y_1 + 1, 1) \\ M(y_1 + 2, 1) \end{pmatrix} = \begin{pmatrix} 0.8 \\ 0.8 \\ 0.9 \\ 0.9 \\ 0.8 \end{pmatrix}$$

These intensity (or brightness) values are then used as weights and the center of mass ρ for the vector

is calculated as follows

$$\rho(n) = \frac{\sum_{i=1}^{5} P(i) \cdot r(i)}{\sum_{i=1}^{5} P(i) \cdot c}$$

where $r(i) = \{-2, -1, 0, 1, 2\}$ and c is the column index. This value will represent where the *weight* of the track lies in that column relative to y_1.

Step 2:
In the next step $n + 1$ the column index c is increased by one and a new column vector of size five is extracted.

However, this time the column vector is centered at the row with index $y_1' = y_1 + nint(\rho(n))$ obtained in the previous step. By using the value $y_1 + nint(\rho(n))$, the search for the new vector's mass center is confined to a neighborhood of where it was located in the previous step n.

Continuing in this way the sequence of dashed circles shown in fig. 7 will be obtained.

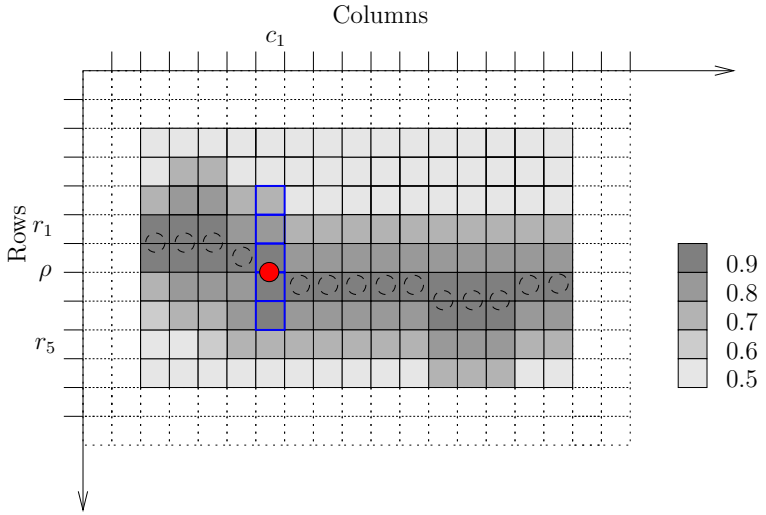

Figure 7: Schematic figure of one track, actual pixels are shown with intensity values. The red dot is the current center of mass and the dashed circles are all the calculated centers of mass along the track.

When reaching the end of one image, the new starting point in the next image will initialize a new tracking sequence and thus "revolve" around the record.

When a track has been read through all the 4 (or 8 or 16, see section 3.1) images the track number will increase by one so that no track is read twice. Eventually the whole record will be read and the centers of mass $\rho(n)$ will be the approximation of the track and incorporate the "raw track data".

5.2.3 Robustness

The algorithm presented above gives a very robust results in terms of not wandering off from the actual tracks. If the intensity of the track becomes fainter, then the algorithm will still be on track thanks to the information about where the track was in the previous column.

6 Sound Enhancement

This chapter describes the theory behind the efforts made to reduce the noise in the extracted sound. Something to have in mind is that all theories discussed in this section and later successfully simulated, have to be implemented in real-time on the DSP-card.
The first section *Data Preparation Filter*, extracts the useful signal from the "raw" data, the second *Noise Reduction*, describes two relevant methods of noise reduction, and the third *Anti-recording filter*, deals with different record characteristics.

6.1 Data Preparation Filter

The information received after the extraction process have to be pre-processed in order to be able to use it successfully.

Because the tracks spirals inwards and since we measure the track variations from the middle, the read out from our transformed images will have low frequency variations that is of no interest, see fig. 8.

Our focus is on the small modulations that represents the music. A high-pass filter would easily extract the interesting parts and get rid of the excess information.

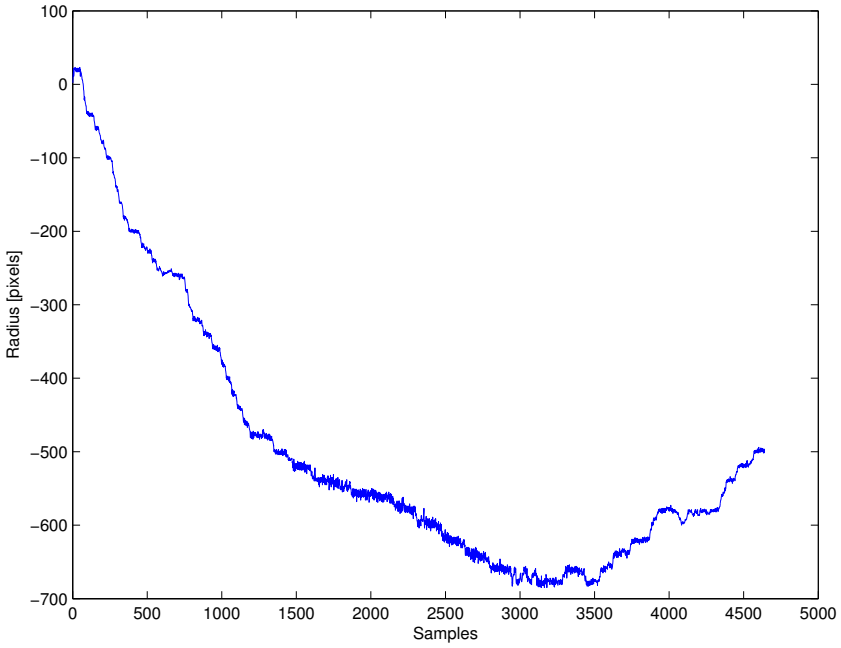

Figure 8: Graph showing the raw data read out from one of the images transformed.

Since the musical content of a 78 rpm, SP-record does not exceed 7-8 KHz a low-pass filter, which removes the out of band noise, could be useful.

The conclusion one can draw from the statements made above is that a bandpass filter would satisfy both conditions and should be implemented.

6.2 Noise Reduction

When considering how to perform noise reduction on the preprocessed sound data, there are three main things that has to be taken into account:

- The acquisition process, i.e the number of scanned images chosen, will definitely influence the signal and result in a additional periodic noise, see section 3.2.
  This will be perceived as a loud periodic "swoosh" to the human ear.
- The only signal available is the noisy signal. In these type of situations it is not possible to cancel out the random noise, but a reduction of the *average effects* of the noise may be possible to perform. This narrows the field of possible algorithms/methods applicable.
- The only way to get data for the noise estimation is to use the first part of the signal. This part exists due to that the first couple of revolutions are generally with out musical content.
  Noticeable is that an assumption that the noise is the same all through out the record has to be made. This is of course not the case, but probably the best solution under the circumstances.

From these conditions two methods of approach were deemed to be successful: *Wiener Filtering* and *Spectral Subtraction*. Other methods like a perceptual modelling, removal of certain frequencies of the sound file and softening of other depending on their significance, and Bayesian statistical restoration were contemplated, but discarded due to complexity and time consummation.
In this chapter, follows a thorough description of the two chosen methods.

6.2.1 Wiener filtering

- Problem:
  - Estimate $x(n)$ from an observation $y(n)$
  - Criterion: Minimize the Mean Square Error (MSE)
- Assumptions and notations
  - $x(n)$ and $y(n)$ are assumed to be two scalar, jointly stationary, stochastic processes, both with zero mean.
  - The covariance functions of these processes are denoted by $r_{xx}(k)$ and $r_{yy}(k)$ and their cross-covariance function by $r_{xy}(k) = E[x(n)y(n-k)]$.
- Structures of the estimator and their optimal expression
  - Finite Impulse Response (FIR) Wiener filtering:

$$\hat{x}(n) = \sum_{k=0}^{N-1} \theta(k) y(n-k) = Y^T(n)\theta \quad ; \quad N < \infty$$

where

$$\theta_{opt} = \Sigma_{YY}^{-1} \Sigma_{Yx}$$

With

$$\Sigma_{Yx} = E[Y(n)x(n)] = [r_{xy}(0) \;\; r_{xy}(1) \;\; \dots \;\; r_{xy}(N-1)]^T$$

$$\Sigma_{YY} = E[Y(n)Y^T(n)] = \begin{bmatrix} r_{yy}(0) & r_{yy}(1) & \dots & r_{yy}(N-1) \\ r_{yy}(1) & r_{yy}(0) & \dots & r_{yy}(N-2) \\ \vdots & \vdots & & \vdots \\ r_{yy}(N-1) & r_{yy}(N-2) & \dots & r_{yy}(0) \end{bmatrix}$$

- Non-causal, infinite dimensional, Wiener filtering

$$\hat{x}(n) = \sum_{-\infty}^{\infty} \theta(k) y(n-k) = H(q) y(n)$$

where $H(q)$ is the non-causal filter with transfer function

$$H(z) = \sum_{-\infty}^{\infty} \theta(k) z^{-k} = \frac{\phi_{xy}(z)}{\phi_{yy}(z)}$$

With

$$\phi_{yy}(z) = \sum_{-\infty}^{\infty} R_{yy}(k) z^{-k} \quad \text{where} \quad R_{yy}(k) = E[y(n) y^T(n-k)]$$

$$\phi_{yx}(z) = \sum_{-\infty}^{\infty} R_{yx}(k) z^{-k} \quad \text{where} \quad R_{yx}(k) = E[y(n) x^T(n-k)]$$

- Causal, infinite dimensional, Wiener filtering $\hat{x}(n) = \sum_{k=0}^{\infty} \theta(k) y(n-k) = H(q) y(n)$

$$H(z) = \sum_{-\infty}^{\infty} \theta(k) z^{-k} = \frac{1}{\phi_{yy}^{+}(z)} \left\{ \frac{z^m \phi_{xy}(z)}{\phi_{yy}^{+}(z)} \right\}$$

With

$$\phi(z) = \phi^{+}(z) \phi^{-}(z)$$

$$\phi^{+}(z) = \sigma \frac{\prod_{i=1}^{n} (z - n_i)}{\prod_{i=1}^{n} (z - p_i)}$$

  - The zeros n_i are inside the unit circle.
  - The poles p_i are strictly inside the unit circle.

6.2.2 Spectral subtraction

Spectral subtraction is non-parametric method to restore the power- or magnitude spectrum of a signal observed in additive noise.

Additive noise increases the mean and the variance of the magnitude spectrum of a signal. The effect on the variance is due to the random fluctuations of the noise and can't be remedied, but by subtracting a mean estimate of the noise spectrum the effects on the mean of the signal can be removed.

- Problem
  - In the time domain the noisy signal model can be given by

$$y(m) = x(m) + n(m)$$

where $y(m)$ is the noisy signal, $x(m)$ is the wanted signal and $n(m)$ is the additive noise. m is the discrete time index.

- Assumptions and notations
  - The signal and the noise are uncorrelated ergodic processes.
- Power spectrum subtraction
  - Division into segment of N samples length of the signal.

- Windowing of each segment, using a Hanning or a Hamming window, to alleviate the effects of the discontinuities at the endpoints of each segment.

$$y_w(m) = w(m)y(m) = w(n)[x(n) + n(m)] = x_w(m) + n_w(m)$$

- Transformation via discrete Fourier transform (DFT) to N spectral samples

$$Y_w(f) = W(f) * Y(f) = X_w(m) + N_w(m)$$

where $Y(f)$, $X(f)$ and $N(f)$ are the Fourier transforms of the noisy signal $y(m)$, the original signal $x(m)$ and the additive noise $n(m)$ respectively.
(Note: We will drop the subscript w for the windowed signal to keep it simple.)

- Reducing the noise variance
When performing spectral subtraction the process distorts the spectrum. That it why it is important to try to reduce the noise variance as much as possible.

If looking at successive blocks of spectral samples, they form a two-dimensional frequency-time matrix which can be denoted by $Y(f,t)$. t denotes the time dimension.
The signal $Y(f,t)$ can be considered to contain a time-varying signal $X(f,t)$ and a random noise component $N(f,t)$. In order to reduce the noise variations, one approach is to low-pass filter the magnitude spectrum for each frequency.

In [1] a simple recursive first order low-pass filter is given by:

$$|Y_{LP}(f,t)| = \rho|Y_{LP}(f,t-1)| + (1-\rho)|Y(f,t)|$$

where the subscript LP is the resulted signal low-pass filtered, and ρ is the smoothing coefficient, which controls the bandwidth and the time constant of the low-pass filter.

- The spectral subtraction is defined by:

$$|\hat{X}(f)|^b = |Y(f)|^b - \alpha|\overline{N(f)}|^b$$

where $|\hat{X}(f)|^b$ denotes the estimate of the original signal spectrum $|X(f)|^b$ and $|\overline{N(f)}|^b$ the time-averaged noise spectra. The parameter α is used to control the amount of noise subtracted from the noisy signal,

$$\alpha = 1 \;\Rightarrow \text{full subtraction}$$

$$\alpha > 1 \;\Rightarrow \text{over-subtraction}$$

To obtain the time-averaged noise spectra, the noise part of the signal, part where the signal is assumed to be only noise. Thus,

$$|\overline{N(f)}|^b = \frac{1}{K}\sum_{i=0}^{K-1}|N_i(f)|^b$$

$|N_i(f)|^b$ is the i^{th} noise frame and K is the number of frames in the noise part.

- Over-subtraction

  The expression of the spectral subtraction illustrates the improvement due to over-subtraction:

$$\begin{aligned} |\hat{X}(f)| &= |Y(f)| - |\overline{N(f)}| \\ |\hat{X}(f)| &\approx |X(f)| + |N(f)| - |\overline{N(f)}| \\ |\hat{X}(f)| &\approx |X(f)| + V_N(f) \end{aligned}$$

  where $V_N(f)$ is the zero-mean random component of the noise spectrum.

  If $V_N(f)$ is well above the signal $X(f)$ then the signal may be considered as lost in noise. In this case, over-subtraction will result in a higher overall attenuation of the noise.

- The transformation into the time-domain is performed using the inverse discrete Fourier transform by combining the magnitude spectrum estimate $|\hat{X}(f)|^b$ with the phase of the noisy signal as

$$\hat{x}(m) = \sum_{k=0}^{N-1} |\hat{X}(k)| e^{j\theta_Y(k)} e^{-j\frac{2\pi}{N}km}$$

  $\theta_Y(k)$ denotes the phase of the noisy signal frequency $Y(k)$; we assume that the audible noise is mainly due to the distortion of the magnitude spectrum and that the phase distortion is largely inaudible [1].

- Negative estimates

  Spectral subtraction may result in negative estimates of the power due to the variations of the noise spectrum. Thus, we need to post-processed the estimated signal using a mapping function T[.] of the form

$$T[|\hat{X}(f)|] = \begin{cases} |\hat{X}(f)|, & \text{if } |\hat{X}(f)| > \beta|Y(f)| \\ func[|Y(f)|], & \text{otherwise} \end{cases}$$

- Processing distortion

  The non-linear processing due to the variation of the noise spectrum is the main problem of spectral subtraction. By expanding the equation of the power spectrum subtraction, we have:

$$\begin{aligned} |\hat{X}(f)|^2 &= |Y(f)|^2 - |\overline{N(f)}|^2 \\ |\hat{X}(f)|^2 &= |X(f)|^2 + \underbrace{(|N(f)|^2 - |\overline{N(f)}|^2)}_{\text{Noise variations}} + \underbrace{X^*(f)N(f) + X(f)N^*(f)}_{\text{Cross products}} \end{aligned}$$

  Since the magnitude spectrum must have non-negative values and according to the previous equation, the following distortions can be identified:

  1. The variations of the instantaneous noise power spectrum about the mean.
  2. The signal and noise cross-product terms
  3. The non-linear mapping of the spectral estimates that fall below a threshold.

  The dominant distortion is due to the non-linear mapping of the negative, or small valued, spectral estimates. This kind of distortion results in a metallic sounding noise named "musical tone noise" due to their narrow band spectrum and the tin-like sound.

  The efficiency of the spectral subtraction depends on the reduction of the noise variations and on the removal of the processing distortions.

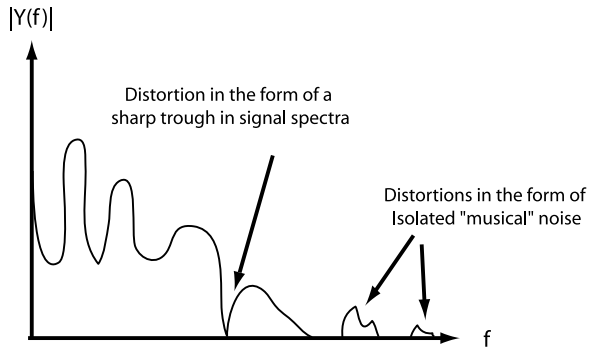

Figure 9: Illustration of distortion characteristics that can occur when spectral subtracting.

The residual often have the following form:
A sharp trough or peak in the signal spectra and often constrained to isolated narrow bands of frequencies.

- Removal of musical tone noise
  When examining the variations of musical noise in the time and frequency domain, we notice that musical noise tends to be relatively short-lived random isolated bursts of narrow band signals, with relatively small amplitudes. Thus, a way for the identification of musical noise is illustrated bellow.

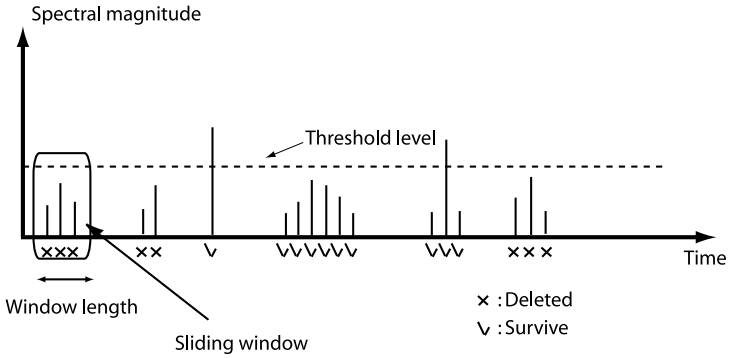

Figure 10: Illustration showing how to filter out the processing distortions known as "musical noise".

Each DFT channel is examined to identify short-lived frequency events. If a frequency component

1. has a duration shorter than a pre-selected time window
2. has an amplitude smaller than the threshold
3. is not masked by signals components in the adjacent frequency bins.

Then it is classified as distortion and deleted.

6.3 Anti-recording Filter

In order to understand what is meant by "anti-recording" or "cutting equalization", one naturally have to understand how the record was recorded [14].

6.3.1 Theory

Basically when recording on a vinyl disk, a cutter-head having a stylus connected to a coil placed in the field of a strong magnet, engraves a grove on the blank disk.

Because of the rate at which the stylus moves translates the original signal amplitude swing's, the low-frequency signals will have a much larger swing than the high frequency signals even though both have the same original amplitude. In order to keep the modulations the same over all frequencies a circuit introduce a 6 dB/octave cut as the frequency decreases.

This gives that when playing the record with this normalized grove, the bass will be low and the treble high. Therefor a reversal of the recording circuit would be called for, i.e 6 dB/octave cut as the frequency increases.

But early cutter heads were highly inefficient, which caused the modulation to decreased above the mid frequencies. Hence to compensate for that the reversal was left flat for frequencies over 200 Hz.

When improvements in cutter heads was introduced, it was possible to fit in more treble on the records. This led to the use of further equalization and a 6 dB/octave cut above a certain turnover frequency. The turnover frequency, however, vary between 3.4 and 6 kHz *depending* on the production label.
Flattening of the very lowest frequencies was also common to reduce rumbling coming from the turntable, see fig. 11.

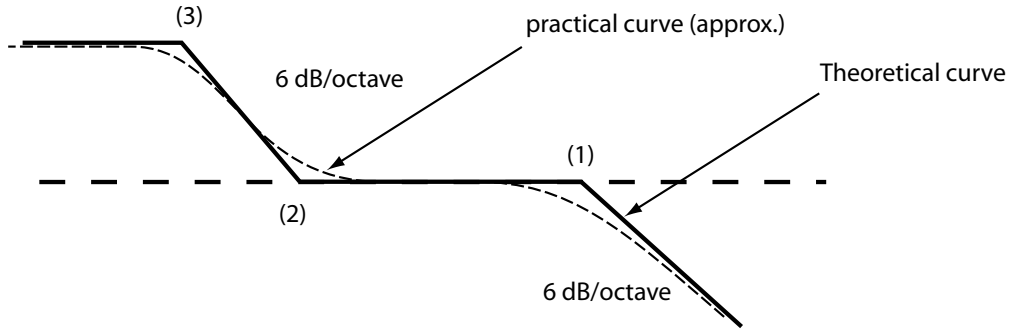

Figure 11: Reproduction curve for vinyl recordings.
1) treble turnover (3 dB point)
2) bass turnover (3 dB point)
3) low bass tip
(Note: The *recording* curve would be the inverse.)

6.3.2 Specifications

So when playing the extracted music a "anti-recording" filter has to be implemented to compensate for the methods used when pressing the record. As mentioned earlier the technics used differed from one label to another, which means that several different filters have to be created.

The table below shows the specific label filters we chose to implement including our own generic filter based on the average values of the others.

Label	*Treble turnover*	*Bass turnover*	*Low bass tip*
DECCA 78	3.4 kHz	150 Hz	20 Hz
ffrr 78	6.36 kHz	250 Hz	40 Hz
BSI 78	3.18 kHz	353 Hz	50 Hz
Generic filter	3.1 kHz	250 Hz	50 Hz

7 MATLAB Implementation

Most of the development was done in MATLAB and this Chapter depicts the structure and framework used therein.

The algorithms described in sections **4.Image Processing** and **5.Sound Extraction** were developed in MATLAB and then transferred in to C/C++-code. The simulation was carried out with ordinary MATLAB-commands [5] using the toolboxes included in the available version, see section 1.4.2.

7.1 Sound Enhancement

The data preparation band pass filter was easily implemented using second order butterworth filters with cutting frequencies at 40 Hz (high-pass) and 8 kHz (low-pass).
The anti-recording filter was never implemented as a simulation in MATLAB.

7.1.1 Wiener filtering

Different versions of the Wiener filter was implemented, see section 6.2.1. They all share the following principle:

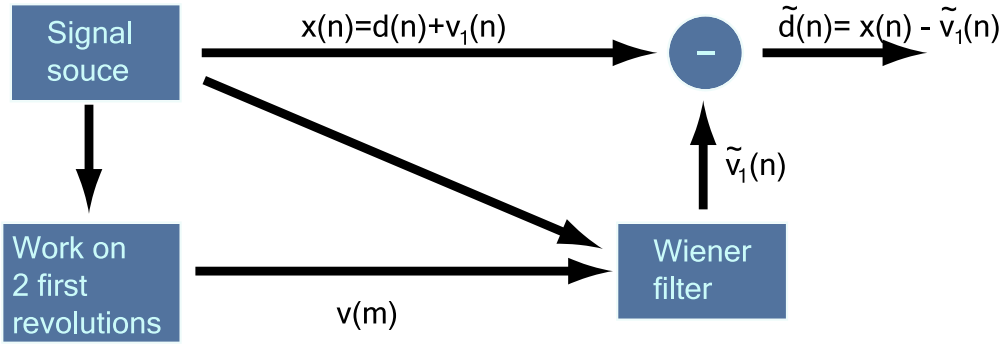

Figure 12: Flow-chart for Wiener filter implementation.
$d(n)$: sound signal
$v_1(n)$: additive noise
$v(m)$: noise part

The implementation consists of:

- Division of the noisy signal into segments of N samples, during which stationarity is assumed. The segments range from approximately 40 to 100 ms in length.
- Computation of the autocorrelation matrix on the noise-only part of the signal i.e the beginning of the record.
- Average of the autocorrelation matrix on the noise-only part of the signal. The averaging was done in several manners. We considered:
  - Averaging over the whole noise-only part of the signal resulting in an autocorrelation matrix of the required size.

[5] For code and details, check out the www-archive at [11]

  - Averaging over one revolution. We consider the different parts of one revolution (4, 8 or 16 depending on the number of scans) and make the average over the number of revolutions the noise-only part include. Generally the noise-only part holds at least 3 revolutions, but for stability reasons only 2 are used.
  - Averaging over the length of one scanned image to keep the periodicity[6] of the noise.

- Computation of the cross-correlation matrix for each part of the noisy signal by taking into account the way the average was done.

- Estimation of the noise for each part of the noisy signal and removal of it from the original signal.

For this kind of implementation, the FIR Wiener filter was used.
The causal and non-causal Wiener filter were also tried by modelling the noise part of the signal and the noisy signal. AR-modelling were implemented.

A *adaptive* FIR Wiener filter was also created. The principle is the same, but the difference lies in that the taps are continuously updated.
The implementation assumes that the input to the filter is a stream of sound samples, rather than blocks of image data.

- First, a model of the noise is created. For each sample in the noise, an estimation of the current auto-correlation is created by calculating:

$$r_{nn}(k) = \frac{1}{N} \sum_{l=-\frac{N}{2}}^{\frac{N}{2}-1} y(m+l)y(m+l-k)$$

  for the input signal $y(m)$, for each m in one frame with only noise.

- The same procedure is repeated to create a model of the noisy signal, i.e.

$$r_{yy}(k) = \frac{1}{N} \sum_{l=-\frac{N}{2}}^{\frac{N}{2}-1} x(m+l)x(m+l-k)$$

  for the input signal $y(m)$.

- For each sample in the sound signal, an FIR Wiener filter is calculated using these auto-correlation estimations as follows:

$$\theta = \begin{bmatrix} r_{yy}(0) & r_{yy}(1) & \dots & r_{yy}(K-1) \\ r_{yy}(1) & r_{yy}(0) & \dots & r_{yy}(K-2) \\ \vdots & \vdots & \ddots & \vdots \\ r_{yy}(K-1) & r_{yy}(K-2) & \dots & r_{yy}(0) \end{bmatrix} \begin{bmatrix} r_{nn}(1) \\ r_{nn}(2) \\ \vdots \\ r_{nn}(K) \end{bmatrix}$$

  Since the 1/N's cancel each other, they are discarded.

- The filter is then applied.

$$\hat{x}(m) = \sum_{l=0}^{K-1} y(m-l)\theta(l)$$

  where $\hat{x}(m)$ is the output signal.

[6] Note the definition in section 3.2

One unexpected problem appeared during the implementation. The covariance matrix R_{yy} would sometimes have very low rank, due to an almost flat auto-correlation function, spawning incorrect filter-taps θ, and leading to peaks in the amplitude, i.e. clicks in the sound.
Two ways of avoiding this were developed: *Smoothing of filter taps* and *Peak detection/removal*.

By using a simple auto-regressive low-pass filter on the form

$$\theta(n) = \rho\ \theta(n-1) + (1-\rho)\ \theta_{new}$$

the disturbing θ's are abated. However, they are not completely removed, and clicks still occur.
The clicks differ from the real noise by having a much larger amplitude. Therefore, they can be detected and removed by simply putting a limit on the noise amplitude.
A combination of the two was also implemented, and proved to give the best result.

7.1.2 Spectral Subtraction

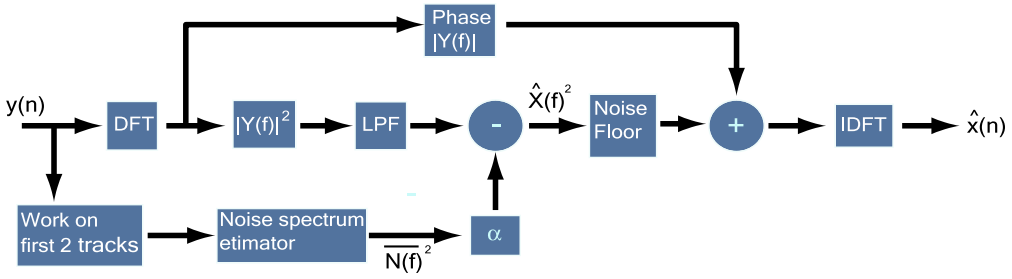

Figure 13: Flow-chart for Spectral Subtraction implementation.

The implementation of the spectral subtraction consist of:

- Division of the original sound file into noise-only part and noisy signal. Division into segments of N samples. The segments range from approximately 40 to 100 ms in length.
  The division includes an overlap of about 50% and was performed considering each part as one vector.
- Transformation via a discrete Fourier transform (DFT) of the time domain signal to the frequency domain of both the noise-only part and the noisy signal.
- Backup of the phase of the noisy signal and treatment of the amplitude.
- Low pass filtering to reduce the noise variance in the noisy signal.
  ρ - deciding the amount of filtering performed - is typically set between 0.85 and 0.99.
- Averaging of the noise-only part of the signal. The averaging was done over the length of one scanned image to keep the periodicity of the noise.
- Power spectrum subtraction defined by:

$$|\hat{X}(f)|^2 = |Y(f)|^2 - \alpha|\overline{N(f)}|^2$$

  α - noise reduction parameter (typically $1 \leq \alpha \leq 6$).

- Removal of some of the processing distortions, introduced by the spectral subtraction, using a mapping function T [.] of the form:

$$T[|\hat{X}(f)|] = \begin{cases} |\hat{X}(f)|, & \text{if } |\hat{X}(f)| > \beta|Y(f)| \\ func[|Y(f)|], & \text{otherwise} \end{cases}$$

  β - noise floor parameter.
  $\beta = 0.01$ represents a difference larger than 40 dB and is a good initial value.

- Transformation via an inverse discrete Fourier Transform (IDFT) of the processed signal to the time domain using the phase backed up.

Note 1:
In section 6.2 one of the steps describing the spectral subtraction was the windowing of each segment using a Hanning or Hamming window. This type of windowing requires a 50% overlap.
Due to the image block size being different each time depending on the sampling frequency, the overlapping should be variable. Therefore a new way to achieve approximately the same results as if windowing where created.
This new method is called *Fading* and is described in section 7.1.3.

Note 2:
The task of suppressing the "musical noise", described in theory in section 6.2, could not be implemented in an effective way. The main problem associated with this theory is that it would be very difficult to create an effective suppression system on the DSP.
Searching through several FFT's and correctly separating the "musical noise" from the genuine signal frequencies, while maintaining the real-time specifications for the sound processing.

7.1.3 Fading

This method accomplishes similar results as windowing, but also has the bonus effect of making the concatenation of the blocks an easy task. This implementation feature is depicted in fig. 14.

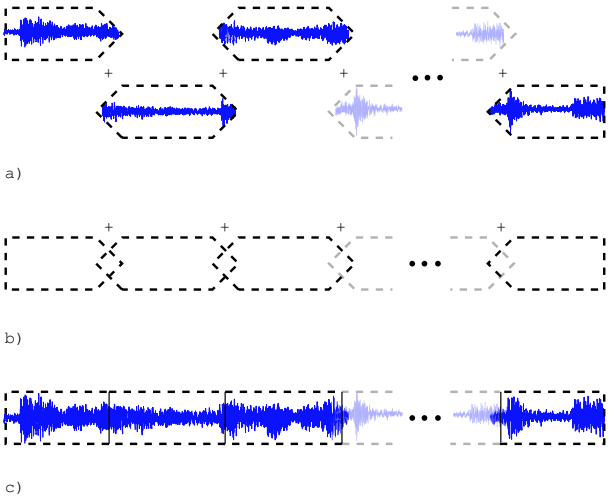

Figure 14: Illustration of the fading feature over one image block.
(a) The image block is divided in to blocks of N samples and faded over the overlap. The blocks are overlapped (b) and summed together (c).

8 Simulation Results

This Chapter deals with and discusses the results from the MATLAB-simulations.

8.1 Sound Enhancement

The process of figuring out which method of noise reduction to use in the final version proved to be easier than first imagined. This because of the simple reason that the implementation of the Wiener filters resulted in a worse final sound than the spectral subtraction. However, in order to be a little more analytical; here follows a short discussion of why this is the case.

Neither the causal nor the non-causal Wiener filter are implementable in any real-time solution and this of course puts them out of the picture. It can also be added that those methods are not optimized for blocks of data.

Of the two implementations of the FIR Wiener filter, the ordinary filter which uses the method described in section 6.2 were not very successful, but the adaptive filter described in section 7.1 was able to remove the periodic "swoosh" or the "swoosh-train" from the signal. Although, it still left a bit of noise left.

The spectral subtraction has a lot of benefits compared to the adaptive FIR Wiener filter.

- It is more robust since the adaptive filter uses inverse matrix operations, which makes it more sensitive to bad autocorrelation estimations causing low rank matrices.
- It utilizes spectral noise subtraction in a mean power sense, which is more intuitive than the adaptive filter's MSE minimization.
- Furthermore, the algorithm of the Wiener filtering is computationally heavy, while the one of spectral subtraction is relatively inexpensive in terms of computational complexity.

Thus only the spectral subtraction was implemented on the DSP and in the prototype.

9 Host/GUI Implementation

9.1 Introduction

This section gives a brief introduction to the 78rpm-program. Before you can use the program you need to scan 4,8 or 16 images as described in the section on scanning procedures. When this is done you are ready to use the program.

9.2 78rpm program

The first screen that appears when the 78rpm program is opened is figure 15. Here you can either choose

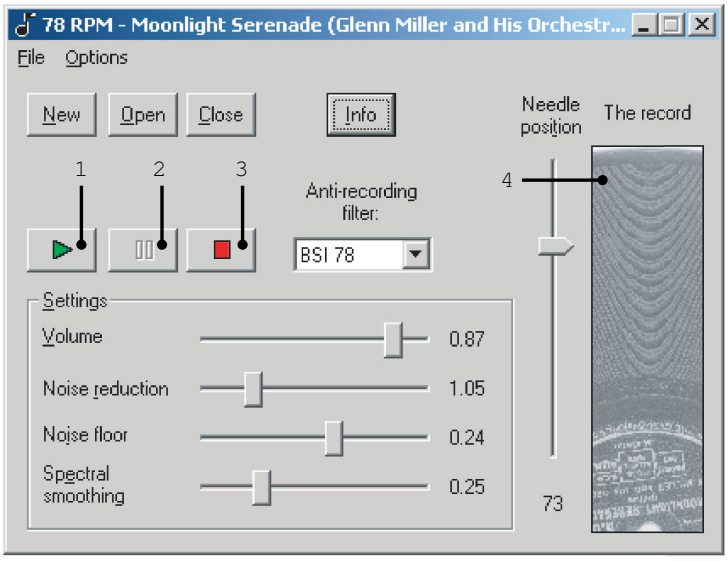

Figure 15: The main interface.

to create a new 78rpm-file from scratch by selecting NEW or OPEN a previously created file. If you select NEW, skip to section 9.2.1 or OPEN, then skip to section 9.2.2.

9.2.1 Create new 78rpm file

If you chose to create a new 78rpm file figure 16[7] appears, here you can choose how many images you already have scanned[8] (4,6,8 or 16), choose the appropriate number and click NEXT.
This will bring-up the figure 17 where you can add the images you have scanned by pressing ADD. Pressing ADD will bring-up a file dialog where you can select the images. Having selected the images and pressing OPEN will make their names appear in the area 1 shown in figure 17.

[7] In this screen the option to scan (in position 2) appears but this hasn't been implemented.
[8] If you havn't scanned any images you need to do this before continuing see the section on scanning procedures 3.

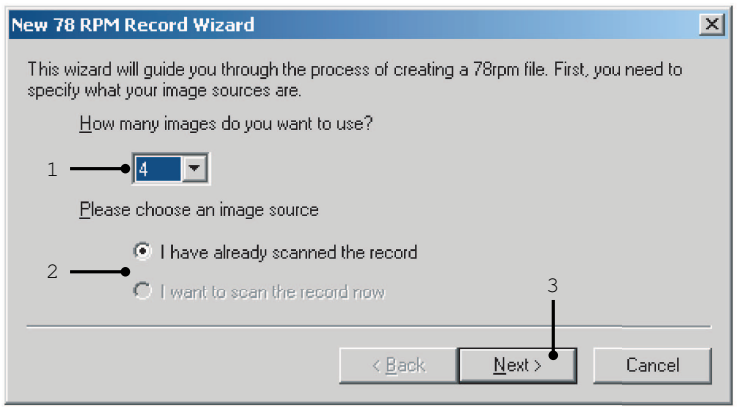

Figure 16: Step 1 - Scanning the record

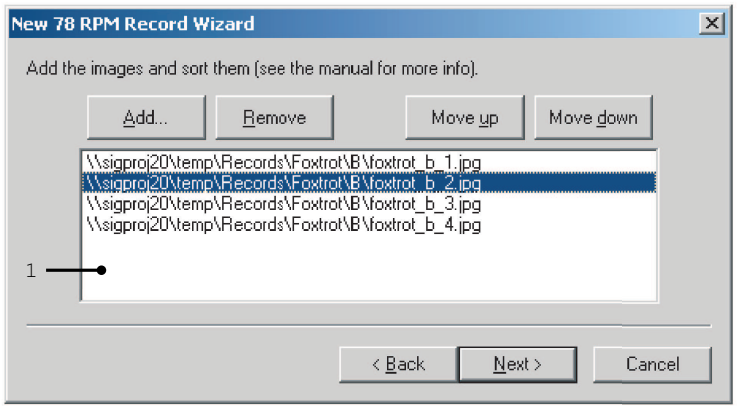

Figure 17: Step 2 - Adding the scans to the program.

By selecting one of the filenames and pressing REMOVE,MOVE UP or MOVE DOWN will either remove the selected file, move it up in the list or move it down in the list. Use these buttons to sort the images in the right order i.e you need to make sure that the images are ordered such that the tracks on the first image continues on the second and so on.

When you have ordered the images and want to continue, press NEXT this will bring up figure 18.

Here you can specify how the images should be stored in the 78rpm format. The compression can be set to one of JPG, PNG or NONE. The different compression schemes will affect the size of the 78rpm file, *png* and *none* will NOT affect the sound quality but *jpg* will. If you chose to use jpg compression

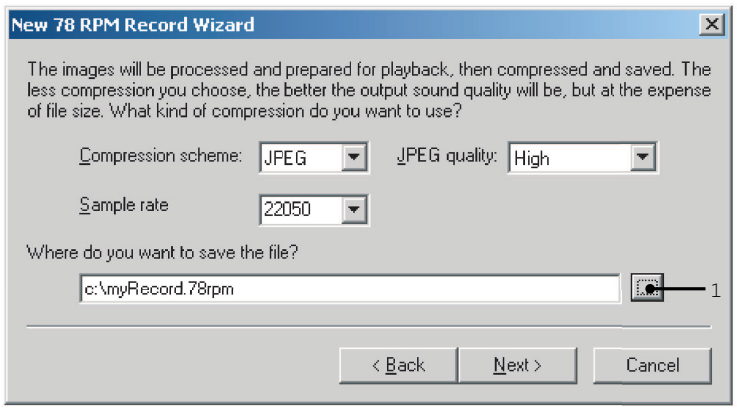

Figure 18: Step 3 - Saving location and format selection.

you also have the option to set the JPEG quality (compression level). Select the wanted SAMPLE RATE also. When you are satisfied press 1 to be able to name the file and choose where it will be saved. Then press NEXT which will bring up figure 19.

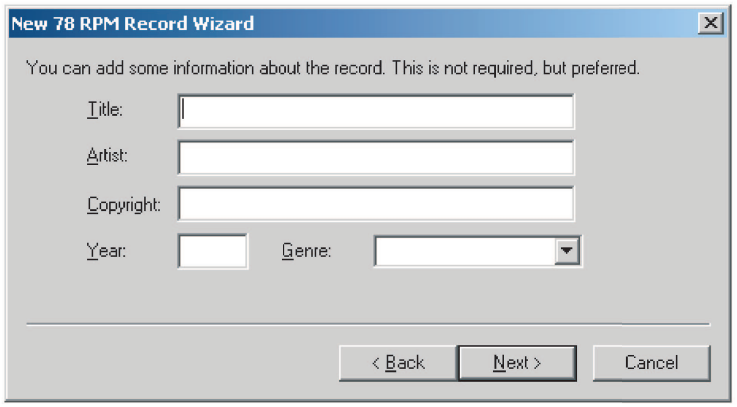

Figure 19: Step 4 - Adding preferred record information.

This screen lets you enter information about the record, that will be embedded in the 78rpm-file, this information will later be retrievable when you play the record. Enter the information you want (you don't need to enter it though) and press NEXT. After you press the NEXT-button a screen will appear asking you if you would like to change anything, if there is something you want to change simply press BACK and redo that step otherwise press NEXT and the conversion will commence. After a while you get back to figure 15. Now the 78rpm-file will have been saved.

9.2.2 Open existing 78rpm file

If you chose to OPEN an existing 78rpm file, a file dialog will appear asking you to select a previously created 78rpm file, select a file and press OPEN. Now figure 20 appears. This screen will let you

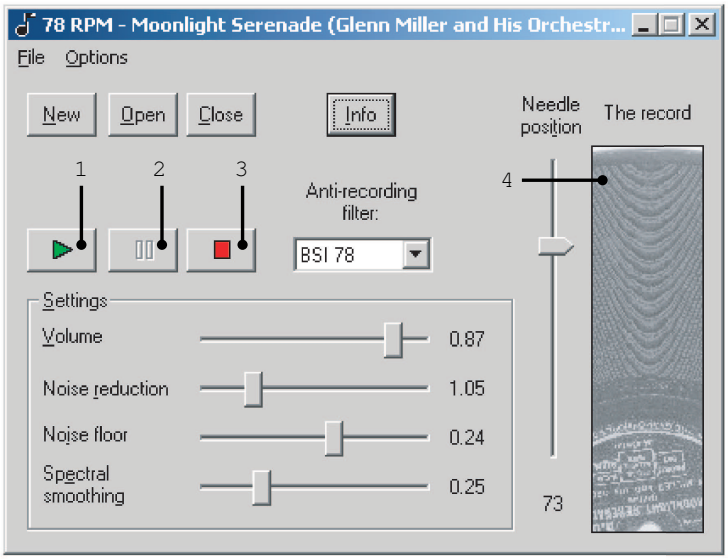

Figure 20: Step 5 - Back to the Main interface.

interactively control the sound playback. By pressing 1 the system will start playing the sound, you can pause the playback or stopping it anytime by pressing 2 or 3 respectively. To see the information about the record press INFO this will bring up info about this record (if it was entered).

In position 4 you can see an excerpt of the record you are playing, next to this image you can see a meter that continuously show the radial position of the needle. If you at anytime want to skip ahead in the record (or back), simply slide the needle to the right track.

To control the noise reduction various options are at your disposal. First of all you can select the type of Anti-recording filter to be used by selecting ANTI-RECORDING FILTER, this will let you choose among various predefined filters. If you want a more fine control of the type of noise reduction to be performed you can resort to the *Settings* section. Here you find the VOLUME, NOISE REDUCTION, NOISE FLOOR and SPECTRAL SMOOTHING-sliders. These sliders control the parameters of the spectral smoothing filter described in the theory section. Feel free to experiment with different combinations of the sliders. Note that you *can* do this while the sound is playing!

10 DSP Implementation

In the specifications section 1.3 it is stated that the DSP that should take care of all the signal processing. This chapter describes how the DSP was programmed.

10.1 Background

The Host, the PC, sends several blocks of data to the DSP that has been extracted from the SP record. The DSP should process the data in real time and play it on the speaker with good sound quality [9].

Real-time in this case means that when the host starts sending data containing music the DSP should play it on the speakers with out any interruptions or delays.

10.2 Survey of the DSP-process

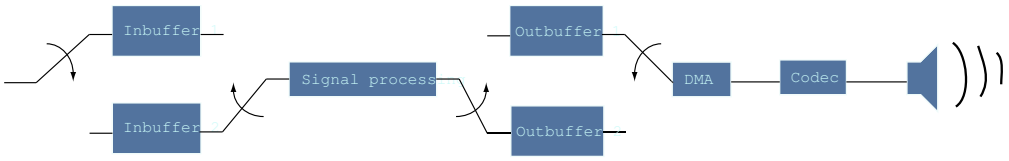

Figure 21: The buffer scheme that was used.

To be able to solve the real-time problem two in-buffers, where the host can input the data, and two out-buffers, that the DMA can read from before passing it on to the Codec, were created.
When the host puts data in one of the in-buffers the DMA reads from the corresponding out-buffer while the data in the second in buffer is being signal processed and passed to the second out-buffer. When the DMA is ready with a buffer it creates an interrupt and the buffers are changed so the host sends data to the second in buffer and the DMA reads from the second out-buffer while the data in the first in-buffer is processed and sent to the first out-buffer (see fig. 21).

Before the host starts to send the data with the music it sends just noise to the DSP so that a power spectrum of the noise can be created and used later in the spectral subtraction. The signal-processing block starts with a band pass filter, the data preparation filter, then the power spectrum of the filtered signal is created.

After that, the spectral subtraction takes place, subtracting the noise spectrum from the signal spectrum. The signal is then transformed back to the time-domain and the phase is added.

The "anti-recording" filter is then applied.

The signal is made into a two-channel mono signal and then put in the out-buffer that the DMA isn't reading from at that moment. When done reading the other buffer the DMA then switches and starts to read the newly processed data and send it on to the Codec.

At the same time as the DMA sends the data to the Codec, which plays the signal on the speakers, the host reads the current out-buffer data as well. By doing this the host can save the processed data on the hard drive so the user can listen to it later without using the DSP.

[9] See the specifications 1.3

10.3 Specifics

The DMA is set up so it has a fixed destination address and a changing source address. When the DMA has read from the source buffer and comes to the last bit, it generates an interrupt and changes the source address.

The interrupt routine starts the signal processing and updates the flags for the host where to read from and where to send data. If all the blocks that should be sent is sent, the interrupt routine also stops the DMA and sets a flag to indicate for the host that the DSP is done. This also happens if the user has requested a stop.

The routine also clears the interrupt bit so a new interrupt can be generated.

The main loop is just a loop that doesn't do anything except waiting for the DSP to be stopped by the user or to send all the blocks of music.

When all blocks are sent or the user choose to stop the program it stops the DMA and sets a flag to indicate for the host that the DSP is done. It works the same way as the interrupt routine does when all the buffers are played or the user wants to stop. This is to prevent getting an infinite loop between the host and the DSP.

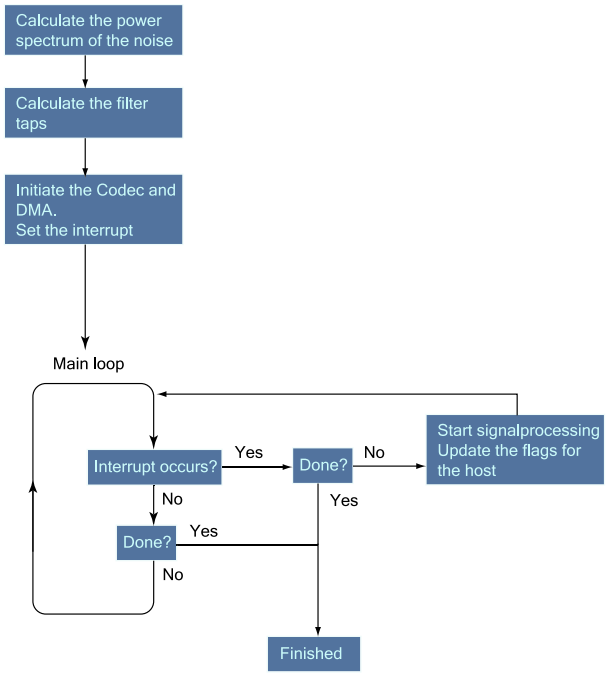

Figure 22: A flow-chart for the program.

10.4 Data Preparation Filter

The second order band pass filter used to prepare the data for spectral subtraction is implemented in the following fashion.

First order low-pass:

$$H_L = \frac{1}{1 + \frac{s}{\omega_1}} \qquad \omega_1 = 2\pi f_1$$

First order high-pass:

$$H_H = \frac{\frac{s}{\omega_2}}{1+\frac{s}{\omega_2}} \qquad \omega_2 = 2\pi f_2$$

Together the form a first order band pass:

$$H_B = H_H \cdot H_L$$

This gives the second order bandpass:

$$H_B^2 = (H_H \cdot H_L)^2 = \frac{(\frac{s}{\omega_2})^2}{(1+\frac{s}{\omega_1})^2(1+\frac{s}{w_2})} =$$

$$= \frac{\frac{1}{\omega_2^2 s^2}}{1+2(\frac{1}{\omega_1}+\frac{1}{\omega_2})s+((\frac{1}{\omega_1}+\frac{1}{\omega_2})^2+\frac{2}{\omega_1\omega_2})s^2+2(\frac{1}{\omega_1}+\frac{1}{\omega_2})\frac{1}{\omega_1\omega_2}s^3+(\frac{1}{\omega_1\omega_2})^2 s^4}$$

$$= \frac{b_2 s^2}{a_0+a_1 s+a_2 s^2+a_3 s^3+a_4 s^4}$$

The filter can be rewritten in the time-domain as:

$$Y(s) = H_B^2(s)X(s)$$

$$a_0 y + a_1\frac{dy}{dt} + a_2\frac{d^2y}{dt^2} + a_3\frac{d^3y}{dt^3} + a_4\frac{d^4y}{dt^4} = b_2\frac{d^2y}{dt^2}$$

This can be translated into discrete form as:

$$y \to y_n, \quad \frac{dy}{dt} \to f_s(y_n - y_{n-1}), \quad \frac{d^2y}{dt^2} \to f_s^2(y_n - 2y_{n-1} + y_{n-2})$$

$$\frac{d^3y}{dt^3} = f_s^3(y_n - 3y_{n-1} + 3y_{n-2} - y_{n-3}), \quad \frac{d^4y}{dt^4} = f_s^4(y_n - 4y_{n-1} + 6y_{n-2} - 4y_{n-3} + y_{n-4})$$

$$\begin{aligned}\implies \quad & (a_0 + f_s a_1 + f_s^2 a_2 + f_s^2 a_2 + f_s^3 a_3 + f_s^4 a_4)y_n + (-f_s a_1 - 2f_s^2 a_2 - 3f_s^3 a_3 - 4f_s^4 a_4)y_{n-1} + \\ & (f_s^2 a_2 + 3f_s^3 a_3 + 6f_s^4 a_4)y_{n-2} + (-f_s^3 a_3 - 4f_s^4 a_4)y_{n-3} + (f_s^4 a_4)y_{n-4} \\ = {} & f_s^2 b_2 x_n - 2f_s^2 b_2 x_{n-1} + f_s^2 b_2 x_{n-2} \quad \implies \quad \text{Discrete second-order band pass filter}\end{aligned}$$

Since the data preparation filtering is done in discrete time these last ones are the filter taps implemented.

10.5 Noise Reduction

10.5.1 Noise Estimation

The noise estimation is implemented in a pretty straightforward manner.

The two first revolutions of the record are saved in separate buffers. The data is then divided in to small overlapping buffers and Fourier-transformed. The data is now on complex form.

The complex data is squared and saved in a buffer that have the same size as the music data buffers, later sent to the DSP. This buffer size is a bit bigger than the size of one revolution divided with the number of images.

The values that are saved are the means of the values of the corresponding buffers of the two revolutions. Because the power spectrum is mirrored around zero only half of the spectra is used.

10.5.2 Subtraction

The first bit of the signal processing is solved like the noise estimation, but instead of just noise the input is music and noise.

After the values are squared and saved, they are also low pass-filtered before the corresponding bit of the noise-estimation is subtracted. The values can not be negative so they must be tested before anything else is done. If the values are negative, they are set according to a user chosen variable. It is then squared and filtered.

The data is now transformed from the power spectrum to the frequency spectrum and the phase is added.

The processed spectrum, being only one half, is now copied so a whole spectrum is made. The spectrum is inverse Fourier-transformed back to the time domain.

The blocks now be merged together. This is accomplished by fading, see section 7.1.3.

At the same time as the blocks are merged, the music is filtered with the "anti-recording" filter.

10.5.3 Specifics

The buffer sizes are restricted to 512 or 1024 samples. This because both the processing speed of the DSP and the compromise between time- and spectral resolution are somewhat optimized for these sizes.

The noise reduction are only an option when using the sampling frequencies, $f_s = 11025$ or 22050 kHz, if $f_s = 44100$ is chosen the noise reduction feature will not be available.

10.6 Anti-recording Filter

The "anti-recording" filter is implemented as an ARMA-process with five taps that is calculated in the beginning of the program. The difference in the filters is the turnover frequencies, see section 6.3. The turnover frequencies are stored in the program and the host sets a flag to indicate which filter the user wants.

The main program calls a function that takes the sampling frequency and the wanted turnover frequencies and then uses them to calculate the filter taps.

10.7 Two-channel mono signal

After the "anti-recording" filter the signal is finally processed, but there is still one more thing to do and that is to make the signal that now is a one-channel mono signal to a two-channel mono signal. This is done by just copying the processed channel to the other channel before passing it to the DMA.

11 Implementation Results

The final prototype has the following features:

- Play-back of scanned SP-records at 11025, 22050 or 44100 kHz.
- A user-friendly interface that lets the user
  - Convert images
  - Play them through the DSP.
  - Alter the following noise reduction parameters in real-time.
    The amount of *Noise Reduction*, the *Noise Floor* level and the level of *Spectral Smoothing*. The *Volume* is a parameter as well.

- The ability to perform noise reduction through spectral subtraction.
- A fully automatic track searching algorithm.
- Conversion and Compression of images for efficient storage and later play-back.
- Possibilities for saving the extracted sound on hard drive.

12 Conclusions

The objective of the project was to create a fully automate system, which from a SP-record could extract,process and output sound information. The specifications required that all sound processing be performed on the DSP.

These goals have all been fulfilled and the work well documented in this report.

The project has proved that it is possible to create a user-friendly system, adapted to run on the average home desk top with a DSP-card, which can, from images using a ordinary retail scanner, automatically retrieve good quality sound information. The features presented in the Host/GUI-section (see 9) have also proven that the system is adaptive and real-time oriented towards its user.

These are the qualities that often today separates a good, working system from an excellent and impressive system.

12.1 Improvement Possibilities

- *Create a "musical noise" suppression system.*
  Since the only real remnant of the noise, when the signal has undergone the sound enhancement, is the metallic low-volume clicks referred to as "musical noise", a system that can correctly identify and remove these process distortions would greatly benefit the output sound quality.
- *Develop a more robust track indexing routine.*
  The indexing of the tracks could be done in a more robust way. One possibility could be to do some kind of spatial separation check so that a missed track could be spotted and re-indexed.
- *Develop a system which is not so sensitive to faulty indexing.* As the system is designed now too much depends on the track indexing, one faulty index will render the rest of the extracted tracks useless. So it would be good to improve the track indexing or even better design an algorithm that accepts some faulty indices and compensates for it.

References

[1] Saeed V. Vaseghi, *Advanced digital processing and noise reduction*, 2^{nd} edition. John Wiley & sons, Ltd. Chichester, 2000.

[2] Monson H. Hayes, *Statistical digital signal processing and modeling*, John Wiley & sons, Inc. New York, 1996.

[3] Simon J. Godsill and Peter J.W.Rayner, *Digital Audio Restoration*, Springer-Verlag, London, 1998.

[4] Brian W.Kernighan and Dennis M.Ritchie, *The C programming language second edition*, Prentice hall, Upper Saddle River, New Jersy, 1988.

[5] Davis Chapman and Jeff Heaton, *Sams Teach Yourself Visual C++ 6 in 21 days, Professional Reference Edition*, Sams Publishing, Indianapolis, 1999.

[6] Stanley B.Lippman and Josée Lajoie, *C++ Primer third edition*, Addison-Wesley longman, Inc., Massachusetts, 2000.

[7] Naim Dahnoun, *Digital Signal Processing Implementation using the TMS320C6000 DSP platform*,Pearson Education Limited, Harlow, 2000.

[8] Nasser Kehtarnavaz and Burc Simsek, *C6x-Based Digital Signal Processing*,Prentice Hall, Inc., New Jersey, 2000.

[9] *TMS320C6000 Assembly Language Tools User's Guide*,Texas Instruments, 1999.

[10] Håkan Hjalmarsson and Björn Ottersten, *Lecture notes In Adaptive Signal Processing 2E1350*, Royal Institute of Technology, 2002.

[11] Internet reference, *Group Light-blue - Digital Needle*, 2E1366 Project Course in Signal Processing and Digital Communication. KTH, Stockholm, Sweden <http://www.s3.kth.se/signal/edu/projekt/students/03/lightblue/>, Updated June 2, 2003.

[12] Internet reference, *Digital Needle - A Virtual Grammophone*, <http://www.cs.huji.ac.il/ springer/>, Retrieved June 2, 2003.

[13] Tobias Oetiker et al. *The Not So Short Introduction to LaTeX 2ε* [10].

[14] Internet reference, *78 RPM and RIAA Record Preamplifier*, <http://www.vadlyd.dk/English/RIAA_and_78_RPM_preamp.html>, Retrieved June 2, 2003.

[10] `www.nada.kth.se/datorer/tex/doc/latex/general/lshort.pdf`