# Abstract

Various efforts have been made in the past to restore impairments of old motion pictures, such as bleaching, dust and scratches by digital image processing means. However, these efforts were confined to the picture frames.

Restoration of movie sound is currently being performed in the digital domain, too, but on the audio data delivered by a reproduction unit. In the process of converting the optical sound tracks to audio, much of the additional information visibly present on the sound track is buried in the audio signal.

The approach taken in this diploma thesis aims at performing restoration tasks already on the digital image, where defects are more clearly identifiable. This also provides a means for discerning defects that were already recorded onto the sound track and others that were introduced later. The original characteristic of a sound track is thus easier to restore.

Algorithms for dust removal, film grain reduction and azimuth correction were implemented in a C++ framework, allowing the accommodation of a new restoration task by simply adding a new module.

The software could be tried on an actual example, and the processing has led to a considerable decrease in noise and improved speech clarity. Defects present on the original recording sometimes appeared more pronounced.

# Table of Contents

# *Preface*

I chose this diploma thesis not only because of the interesting technical field of digital image processing, but also because it promised insight into something I hadn't heard of before: Optical movie sound tracks and their history. Later, I have once again realized how much work and research is behind something we usually take for granted.

In the course of the project, I had the opportunity to learn not only something about the exciting history — technical and non-technical — of film in general and of film sound in particular, but I have also gotten the chance to learn C++, Matlab, digital audio processing basics and a lot more.

Patrick Streule, March 1999

## *Acknowledgments*

Besides my professor, Prof. Dr. A. Gunzinger and my supervisors Armin Wittmann and Dr. Werner Graff, I would like to thank the following persons and institutions who supported me during this project.

Reto Kromer, Cinémathèque Suisse
Christian Beusch, Magnetix
Pascal Suter, ETHZ
Clive Tobin, Tobin Cinema Systems, Inc.
SwissEffects, Zürich
Egli Film, Zürich
Audio-Ciné Walter Voigt AG, Unterlunkhofen

And, of course, my parents and friends.

CHAPTER 1

# Introduction

Attempts to record sound optically have been made as early as 1857, not with the application of film sound in mind, however. The first photographic recordings comparable to the standard optical sound tracks used later stem from 1886. Standardized optical movie sound tracks started to be popular in the 1920s, with the first major sound pictures appearing between 1925 and 1930.

As its name implies, the optical — or photographic — sound track is recorded onto the film by the same means as the picture, and is therefore subject to the same deteriorations occurring over time, such as scratches, dust, shrinkage due to the nitrate film base and many others.

Up to now, restoration of old worn-out copies has usually been achieved by reproducing the film on a standard projector and digitizing its audio output. Audio restoration algorithms could subsequently be applied to the sound data.

In this thesis, a new approach was chosen to restore deteriorated analogue optical movie sound tracks. The basic idea was to perform the restoration already on the digital image representation of the sound track, before converting it to audio. This approach has a couple of advantages over the traditional method.

1. Advantage can be taken of the additional dimension of the image. For instance, dust can be recognized as such and is not merely a variation in the audio signal.
2. Consequently, it is possible to discern between noise that was already present on the original recording — and is thus part of the recorded signal — and noise introduced later by scratches and dust, for example. This is an important property if the aim is to restore the original characteristic and not just to make it sound better.
3. It does not rely upon a specific film format. The scanner that was used is laid out to accommodate shrunken stock and different film dimensions. Hence, even very old films, that are not reproducible on a projector anymore, could be restored. Algorithms could easily be added to convert early experimental sound formats as well.

I see the image based and the audio based restoration methods as complementary rather than rivaling techniques. The more appropriate choice depends upon the defect to be restored.

The fundamentals and an actual implementation are presented in the following chapters, the next section giving an overview.

## 1.1 Overview

First, some existing methods for the restoration of optical movie sound tracks are introduced, along with a description of the new approach.

The different standard formats that were used for optical sound recording since the early days of motion picture sound are explained in the next chapter “Optical Sound Tracks” on page 9. The very early formats are only touched upon — the wealth of these experimental tracks would fill an entire book. This thesis also focused only on 35 mm formats. The same principles apply to the 16 mm and the 8 mm formats, but they were never of great importance due to their insufficient quality.

An overview of the inherent end external errors affecting the optical sound track quality is given in the chapter “Looking at the Error Chain” on page 25. Special attention was turned to the errors introduced by the scanning process itself.

The actual implementation along with the algorithms used can be found in “Restoration and Conversion to Audio in an Object-Oriented Framework” on page 37. Special care was taken to keep the C++ framework as flexible and expandable as possible in order to allow the easy introduction of new restoration modules.

The algorithms could be tested on an actual example, a reel of the first “Ciné-Journal Suisse”. It was kindly placed at our disposal by the Cinémathèque Suisse. Interestingly, it contains examples of different sound track formats and shows some unusual deviations, too, making it perfect for our purpose. This example and the achieved results are presented in chapter “Example and Results” on page 73.

Judging from the wealth of early sound track formats and the various types of errors associated with this type of recording, a lot more could be done. The outlook on page 77 gives some examples of what could be extensions of the current work.

As the main research and development in the area of optical sound tracks was done in the first half of this century, the technical descriptions stem from the same period and are sometimes hard to find. I have included a bibliography in Appendix A on page 79 to start with, if further interest is taken into the field of optical movie sound tracks.

The usage of software developed in the course of this thesis is described in Appendix B, and a description of the accompanying CD-ROM with some sound examples — restored and unrestored — can be found in the last appendix.

CHAPTER 2

# Restoration Methods

The purpose of this chapter is to present some of the methods developed over the years to cope with the impairments of photographic sound tracks. Traditional methods and experimental methods are opposed to the new approach taken in this project.

## 2.1 Existing Methods

### 2.1.1 Traditional "Blooping"

A splice in the positive or the printed image of a splice in the negative will create a loud click during the reproduction of a sound track. A traditional way used in the earlier days of film sound to prevent this undesirable effect, is to cover the area of the splice with a sinusoidal shape long enough so as not to be audible.

On positive prints, bloops are painted using indelible ink and a stencil to create an opaque area covering the splice where the two sound tracks are joined together.

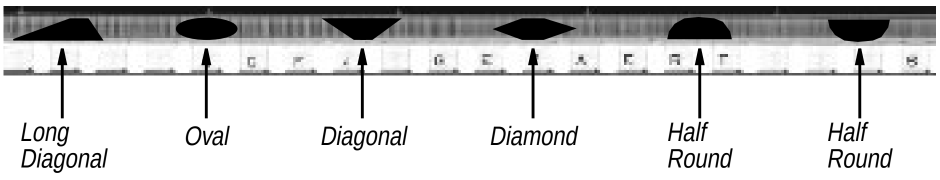


**FIGURE 2-1.** *Bloop configurations used for variable-density and variable-area standard sound tracks (from [2])*

On negatives, bloops are made by punching a hole of a specific shape (see Figure 2-1) into the sound track area, which is going to be opaque on the successive print.

When many negative sections had to be joined together, e.g. for a newsreel, these manually applied bloops are inefficient, so the method of photographic "flash bloops" has been developed. Here, a notch is punched into the perforation area of the negative at the splice location. This notch operates a microswitch turning on a flash light exposing the area of the splice.

Blooping is only suitable for the removal of clicks due to splices. See [1], p.502 and [2], p.647 for more details about blooping.

### 2.1.2 Optical Restoration with Fourier Lenses

In the first half of the eighties, an interesting effort was made by a team of Russian scientists who used coherent (laser) and quasi-coherent (slit source) light for optical scratch suppression. See [26] and [27] for a profound treatment of this approach and its background.

The system is based upon the observation that noise and signal spectra occupy different regions in the spatial domain, making it possible to mask out the noise spectrum without affecting the signal too much.

A rough overview of the quasi-coherent system is given in Figure 2-2. This system also eliminates the noise introduced by the coherent method. The light emerging from slit 1 is transformed into a plane wave by lens 2. The sound track 3 lies in the left focal plane of Fourier lens 4 and the fourier spectrum of the sound track image is formed on the right focal plane 5. The mask 6 covers the region of noise in the spectrum and the second Fourier lens 7 transform the spectrum back into an image representation which could be either projected on a photo cell for direct sound reproduction or on a negative for creating a restored print.

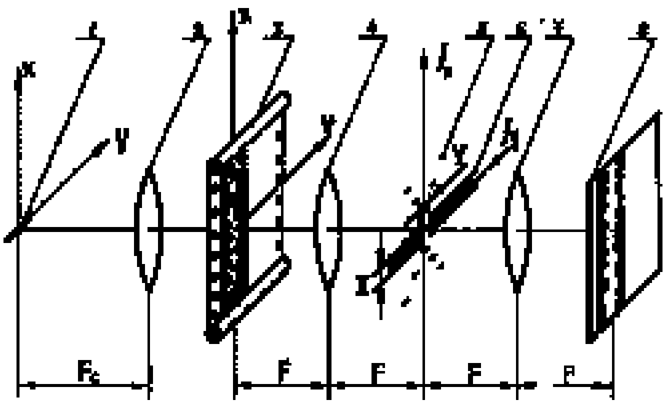


**FIGURE 2-2.** *Coherent spatial filtering of optical sound tracks. (from [27])*

This method can be taken to the digital domain. I tried it on an fragment of a unilateral variable sound track degraded by scratches and film grain using Matlab. The results are displayed in Figure 2-3.

This method is mainly suited to vertical scratches. Other image defects, especially the ones extending horizontally — and therefore audible as clicks — are not covered. The authors describe a theoretical increase in the signal-to-noise ratio (SNR) of 14 dB to 20 dB achieved by this method.

However, the results were only judged visually. An actual conversion to audio was only being prepared at the time of publication. Other possible influences on the sound properties have not been mentioned.

]

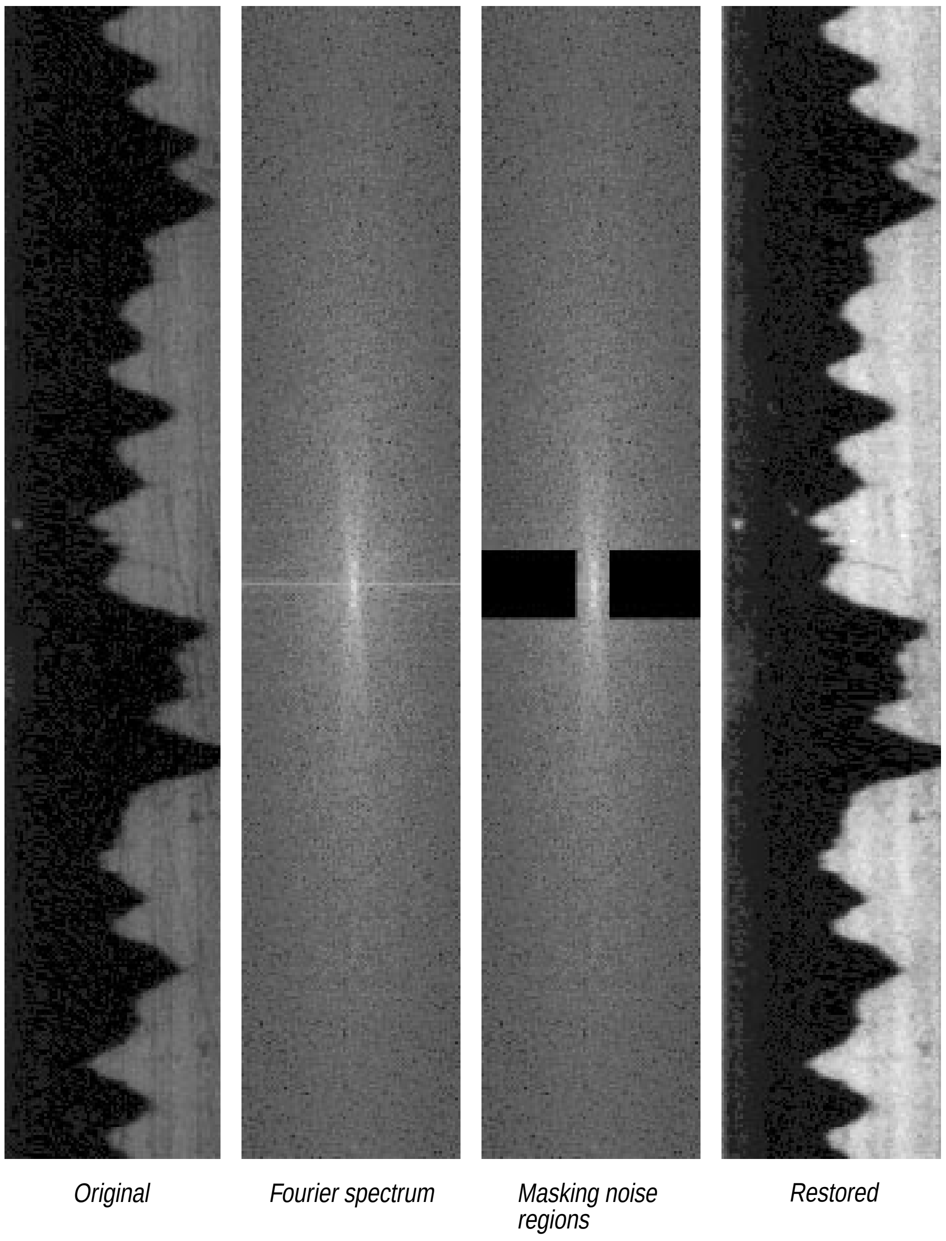


**FIGURE 2-3.** *A digital implementation of the optical method using Fourier lenses in Matlab.*

### 2.1.3 Digital Audio Restoration

A lot of work in the field of audio restoration has been done in various academic and commercial institutions, such as the University of Cambridge in the UK, CEDAR, which emerged from this university as a spin-off company, or Sonic Solutions in the US. The advantage of digital audio processing systems lies in their wide usability for degraded recordings of arbitrary sources. See [5] for in-depth coverage of digital audio restoration methods.

Basic audio restoration procedures are mainly targeted at the removal of clicks and at hiss reduction. More recent and advanced research topics treat the removal of low frequency noise pulses and pitch variation defects. Low frequency noise pulses are — in the case of optical film sound tracks — usually caused by major defects, such as splices. Pitch variation is caused by eccentricities in the film drive system or motor speed fluctuations.

Recent audio-processing methods are based on sophisticated probability theory models and achieve very good results in improving the audio quality.

An advantage of digital audio processing methods is their versatility. Regardless of the medium carrying the audio track, be it old wax cylinders, gramophone disks or magnetic tape, the same methods can be applied on any of them once they are somehow transferred to a digital representation. However, this versatility can also mean sacrificing additional knowledge about the medium, which could possibly endorse the restoration effort.

The parameters for noise reduction or click removal have to be chosen very carefully in order not to affect the actual sound signal.

## *2.2 The New Approach*

Photographic sound tracks are inexpensive and fairly easy to produce and reproduce, but they are subject to various types of errors from the original recording to the final reproduction in a movie theatre. Some of these errors are already made in the optical recording process, others are added later due to wear and tear. See Chapter “Looking at the Error Chain” on page 25 for a description of the different types of degradation.

The approach commonly used today for restoring movie sound tracks is the audio processing method described in the last section.

One has to be aware that “sound restoration” does not necessarily mean improving the sound quality as a whole. The aim is to achieve a result that’s as close to the original master as possible. In the case of an old movie picture, there may be a certain level of noise on the original recordings which belongs to the total characteristic of the movie and should therefore not be removed. Optimally, a restored movie should sound as it was intended by the director and how it has sounded at the time is was run in the cinemas.

The distinction of errors already present on the original master and impairments later introduced in the recording process or due to wear and tear during reproduction is difficult or even impossible to achieve on the pure audio signal.

Our aim was therefore to take the task of audio restoration one dimension higher and to perform the restoration of the optical sound track already on the digital image data, where it is still possible to differentiate between errors and noise belonging to the actual signal on one side, and dust or scratches that were added later on the other side.

### 2.2.1 Data Acquisition

Commercial film scanners like Kodak’s Cineon usually only digitize the content of single picture frames, which is not suitable for the continuous optical sound track.

The scanner realized at the Swiss Federal Institute of Technology together with the University of Basel is much more apt to the needs of film preservation and restoration as it produces an exact digital facsimile of the film, including the optical sound track and the perforation area [11], [12].

The film transport does not rely on the sprocket holes, allowing to scan other film formats than 35 mm as well as shrunken stock. For transport, the film edge is pressed to the drive axle using rubber wheels.

The light source itself is contained in a separate casing, the light being guided by a string of optical fibers to the illumination slit. The section of this fiber bundle is changed from round at the light source to rectangular at the slit.

A Dalsa CL-G1-2098G CCD camera is used for digitizing a line of 2098 pixels across the film width. A PCI board is responsible for acquiring the data and controlling the camera.

The basic operation principle is displayed in Figure 2-4. Refer to [11] and [12] for more details.

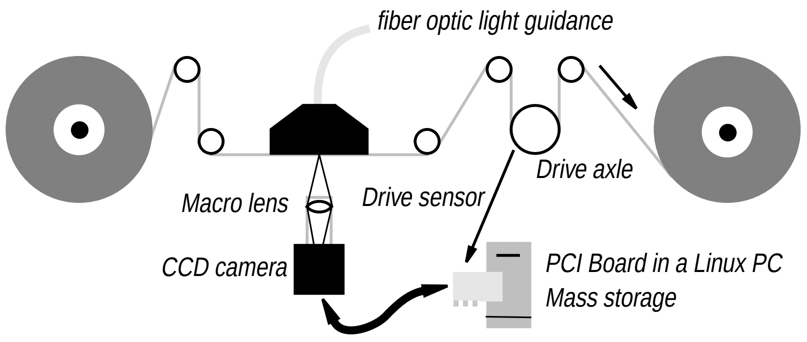


**FIGURE 2-4.** *The scanner setup (from [11])*

For this project, the resolution of the scanner has been increased to about 2700 dpi using a Nikkor 105 mm macro lens and two extensions. Image depth is currently limited to 8 bit in the gray-scale mode.

.

CHAPTER 3

# Optical Sound Tracks

A wealth of optical sound track formats have been developed over the last century. Some of them had a long life, others existed only for a short time, or didn't even make it from paper into reality. Many improvements have been made to the signal encoding, but most steps in photographic recording and reproducing remained the same.

The advent of magnetic recording in the 50's caused a decline in the popularity of optical movie sound tracks, due to the superior quality of multi-channel magnetic tracks. The development of the Dolby A noise reduction system and the stereo optical tracks in the 70's paved the way for a revival of this technique. It was cheaper to produce and was less susceptible to handling damage than magnetic tracks — and now at the same level of quality.

Magnetic sound tracks are still used for 70 mm prints. These prints cost up to 14 times as much as a 35 mm copy with an optical track. Eventually, the magnetic tracks will be replaced by the digital optical sound track formats used for 35 mm prints today.

The following sections cover only 35 mm optical sound tracks formats. Similar formats existed for 16 mm and even 8 mm stock. Magnetic sound track systems are omitted.

## 3.1 Overview

After many early experiments, a set of widely used optical tracks have emerged, and the sound track location for different release print formats has been standardized by the SMPTE. In the following, the nomenclature published in 1938 by the Research Council of the Academy of Motion Picture Arts & Sciences has been used [31].

Beneath the distinction between analogue and digital optical tracks, the analogue optical tracks themselves have to be divided into two incompatible classes. There are the single optical sound tracks on one side — they are often called standard tracks — and the push-pull type of optical sound tracks on the other side. These two classes cannot be reproduced by the same means. The difference is explained in the following sections.

Variable-density and variable-area types, both with and without noise-reduction schemes, are further derivations of the two analogue categories, but they are compatible in reproduction.

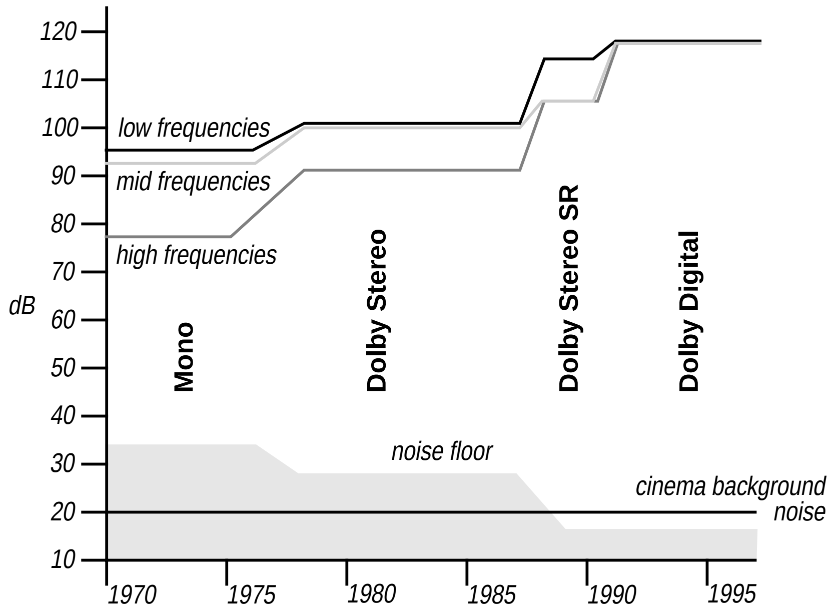
The push-pull sound tracks had better quality, as they were inherently “noiseless”, but required two photo-cells for reproduction (see 3.4 on page 17) and better adjustment of the optical system during recording and playback. Due to the better quality, this type has mainly been used for original recordings, where the better quality justifies the greater effort necessary for adjusting the recording and reproduction equipment.

Variable-area systems can achieve a higher signal-to-noise ratio of up to 6 dB compared to variable-area tracks, due to the lower film grain noise from the track’s opaque area [1], p.357. They allow a higher percentage of modulation, too. On the other side, variable-density tracks are less susceptible to a deviation of the azimuth of the scanning slit (see 4.4.1 on page 31 and 5.14 on page 61) and to non-uniform illumination of the slit.

With the advent of color movies and magnetic sound tracks in the 50’s, the variable-density type disappeared completely. But except for solitary efforts in the area of multichannel sound, the variable-area mono tracks have been a standard until the early seventies. The development of the Dolby noise reduction systems sparked new interest in optical sound tracks, and has led to a steep increase in cinema sound quality over the last twenty years.

An overview of the dynamic range of different sound track types is given in Figure 3-1 below. Typical frequency ranges were 45-8000 Hz for the mono tracks (also due to the standardized Academy characteristic, see 5.21 on page 69), 45-12000 Hz for Dolby Stereo tracks, 20-16000 Hz for Dolby SR tracks and 20-20000 Hz for digital optical sound tracks.

**FIGURE 3-1.** *Relative dynamic range improvement over the last 30 years (from [46])*



The following sections are dedicated to a description of the standardized sound track formats used from the 1920’s until today. One section treats a few of the many early examples of photographic sound recording. Covering all developments in detail would fill an entire book. Some interesting papers about the technical history of motion picture sound are [15], [16], [17], [18], [19], [20] and [21].

## 3.2 The Principle of Optical Sound Track Reproduction

The reproducing unit consists of a light source which illuminates the sound track area (between the perforation and the picture frames) through a lens system and a slit mask. This narrow band of light across the sound track falls onto a photo cell at the back of the film. When the transmission through the film varies, the current induced by the photo cell will vary according to the amount of light falling on it, thereby modulating the sound signal. Thus, the shape of optical sound tracks does not matter as long as it varies the over-all transparency of the sound track area. See Figure 3-2 for a schematic overview.

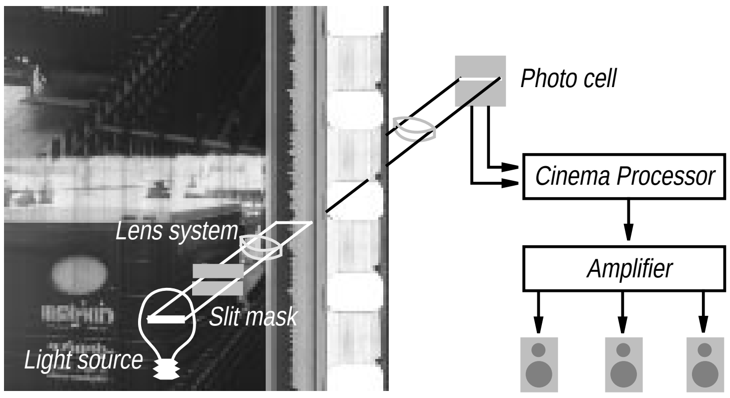


**FIGURE 3-2.** *The principle of reproducing an optical sound track*

## 3.3 Single Optical Sound Tracks

These are the most widespread and most likely to be encountered sound track formats for 35 mm stock. Their position and dimensions have been standardized since at least 1930 and haven't changed notably since then. Because the sound reproducing head can't be positioned at the same spot as the projection unit, the sound track is offset by 21 frames ± 1/2 frame ahead of the corresponding picture's center [49]. This displacement has varied from 20 to 24 frames over the years, the 21 frame offset being the current standard.

The reproducing speed is standardized to 96 perforations or 24 frames (18 inches or 457 mm) per second.

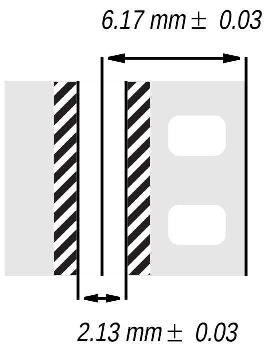


**FIGURE 3-3.** *Distance from the edge and scanned width for mono single optical sound tracks. These were valid over the last 70 years. See the ANSI/SMPTE 40-1997 standard [49] for more details*

### 3.3.1 Mono Variable-Density Tracks

An audio signal on a variable-density track will consist of alternate dark and light lines, extending over the entire track width and merging gradually into each other. An example is depicted in Figure 3-4 (a).

The most important dimensional characteristics can be seen on the right in Figure 3-3. The total width of the sound track was 100 mils, corresponding to 2.54 mm. Not the entire width is scanned by the reproduction unit in order to allow for a certain degree of film weave.

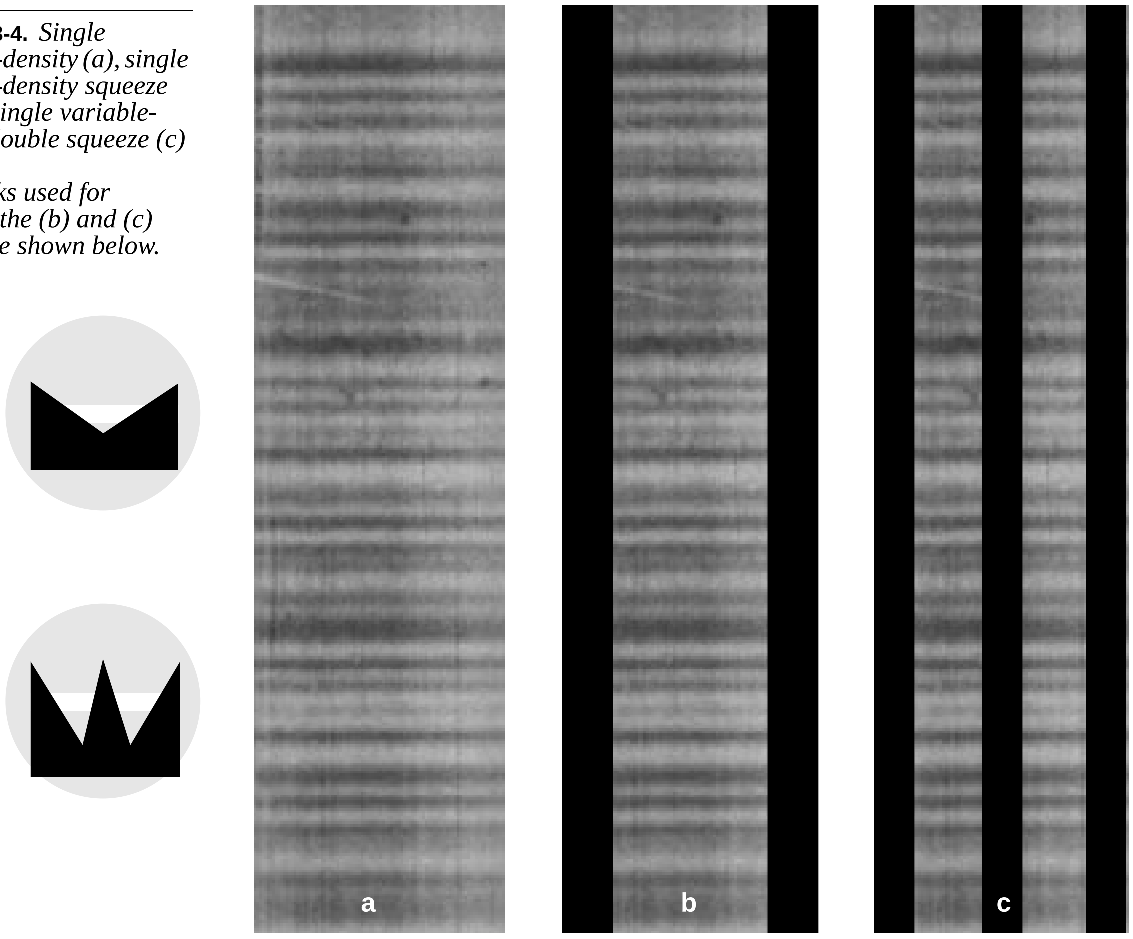


FIGURE 3-4. *Single variable-density (a), single variable-density squeeze (b) and single variable-density double squeeze (c) tracks. The masks used for creating the (b) and (c) tracks are shown below.*

The first optical film sound experiments often used light sources that could be varied quickly enough to follow the audio frequency. These systems usually produced some kind of variable-density track. One of the earliest examples is the one Ruhmer experimented with in Germany in 1900 (see Figure 3-5). He called his invention “photographophon”, but never commercialized it [20].

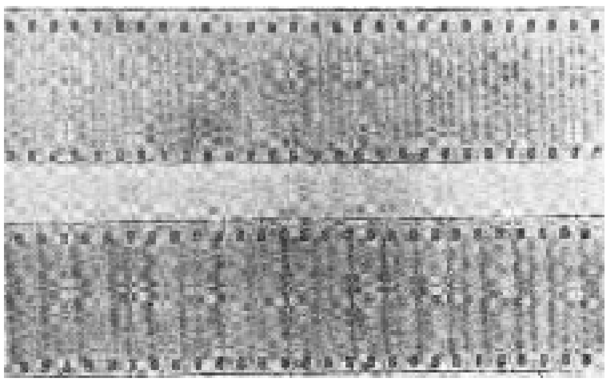


FIGURE 3-5. *Ruhmer’s variable-density sound track of 1900.*

The standard method for recording variable-density sound tracks was the light valve. Most commonly, a two-ribbon light valve was used. The two ribbons are arranged in parallel at a constant distance, usually between 0.2 and 0.5 mils (0.0051 mm and 0.0127 mm), allowing a certain amount of light to pass through. The ribbons move under the magnetic field induced by the signal current flowing through them. Depending on the attracting or repelling force between the ribbons, more or less light can pass through.

The light valve is at right angles to the film negative which is moving at the usual frame rate behind it, which leads to the typical pattern visible in Figure 3-4.

The rather coarse grain of early emulsions introduced considerable noise. Shutters were used to narrow the exposed area, improving the SNR by 3 to 6 dB. Depending on the shape of these shutters, the so altered tracks were termed “single variable-density squeeze” and “single variable-density double squeeze”. A V-shaped mask was used for the squeeze tracks, a W-shaped mask was used for the double squeeze tracks as displayed in Figure 3-4 (b) and (c).

Sometimes, additional control tracks, consisting of a single or of multiple frequencies, were used to control the volume of the sound tracks. This technique was more often used for multi-channel recordings (e.g. the Fantasound system presented in 3.5 on page 18), but existed for mono tracks as well. The control tracks were placed next to the actual signal track, inside the perforation area, or along the film edge. Interesting are the control tracks in the perforation area. The 96 Hz modulation of the sprocket holes was used for controlling the volume. More exactly, the amplitude of this 96 Hz tone determined the amplification of the sound track. The amplitude could be controlled by exposing the interperforation area. Transparency caused only light modulation, while opacity led to high amplitudes. See Figure 3-6 for some examples.

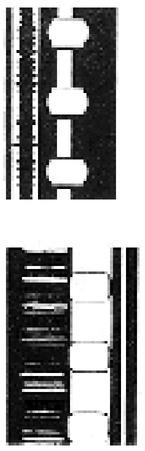


**FIGURE 3-6.** *Examples of control tracks used with mono variable-density and variable-area sound tracks.*

### 3.3.2 Mono Variable-Area Tracks

The other widespread standard tracks are the variable-area tracks. This type of track has the same dimensions as the variable-density tracks presented in the last section (see Figure 3-3) and can be reproduced using exactly the same equipment. However, it is recorded differently.

Usually, a galvanometer is used for the recording, but a light valve could be used, too ([1], p.370). A typical galvanometer setup is shown in Figure 3-8. A mirror is mounted on a knife edge, the mirror being held by two ribbons. The knife bends according to the magnetic field induced by the modulation current from the audio signal, resulting in a rotary motion of the mirror. Depending upon the type of variable-area sound track, a different shape of light is projected onto the mirror which reflects it through a narrow slit onto the negative film. The mask shape determines how much of the recording slit is illuminated and thus, how much of the sound track width is exposed.

Various mask shapes and their corresponding variable-area sound track types are presented in Figure 3-7.

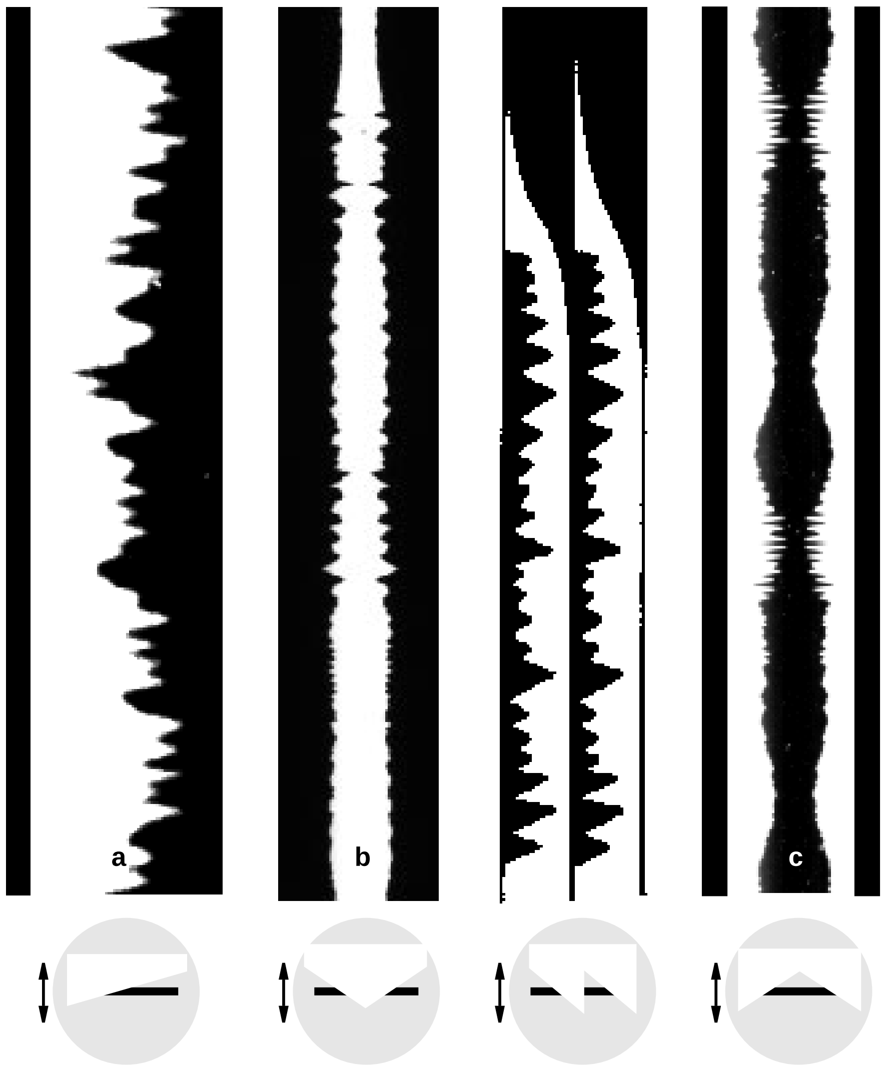


FIGURE 3-7. *Different types of mono single variable-area sound tracks and the corresponding galvanometer light shapes. Unilateral (a), bilateral (b), dulateral (c) and duplex (d).*

As this type of sound track is partly opaque, it is less susceptible to noise caused by film grain. However, the transparent parts are very susceptible to scratches and dust. Therefore the exposed width, which would cover half the track in times of no modulation, is reduced in these times of low or no modulation, in order to keep the transparent area as small as possible.

This was achieved by one of two ways. One way was to use noise-reduction shutters, placed on both sides of the recording slit. They follow the envelope of the signal curve, leaving only a small area transparent when no signal is present. The biggest drawback of this method is the delay caused by the inertia of the shutters. The first few modulation peaks are clipped when a signal is suddenly present again, causing audible harmonic distortions.

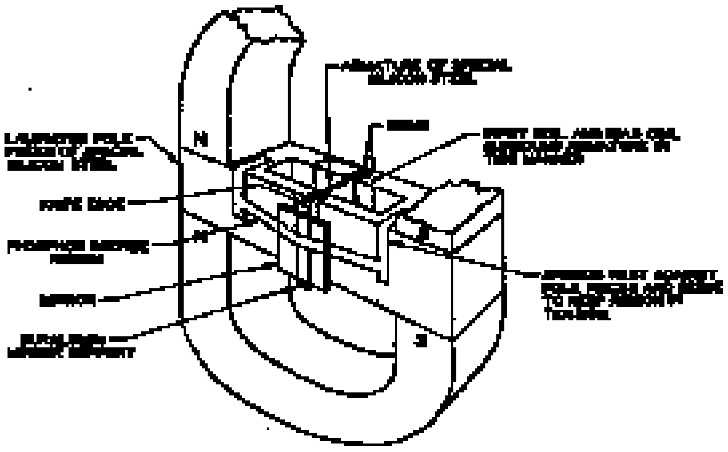


FIGURE 3-8. *A recording galvanometer (from [1])*

The other way is to apply the noise-reduction bias current not to the coils operating the shutters, but to a coil attached to the mirror. In times of low or no modulation, the bias current pulls the galvanometer mirror to one side, narrowing the exposed area ([2], p. 608). The effect of noise reduction on variable-area tracks can be seen in Figure 3-9.

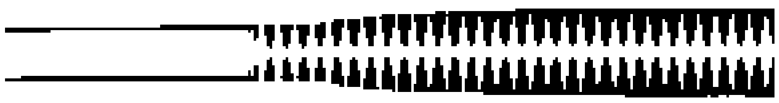


FIGURE 3-9. *Noise reduction on variable-area sound tracks. Clipping of modulation peaks can occur due to the slow attack time of the noise reduction shutters.*

### 3.3.3 Stereo Variable-Area Tracks

The first stereo sound tracks that overcame the experimental phase were magnetic tracks. Soon after, in 1955, an optical stereo track was proposed, but failed adoption. Later, in 1973, the Dolby A noise reduction system could be used to enlarge the signal-to-noise ratio and to compensate for the worse quality due to the narrower track width available for each channel. This led to the breakthrough of the stereo sound tracks.

The recording and reproducing principle is basically the same as described in the previous section, but two photo cells and two galvanometers have to be used for the two separate tracks.

It was soon realized that the two channels could be combined to form a third center channel for better localization of dialogue. This concept eventually led to the four-channel Dolby Stereo and compatible formats (UltraStereo, DTS Stereo). See Figure 3-10 for an example.

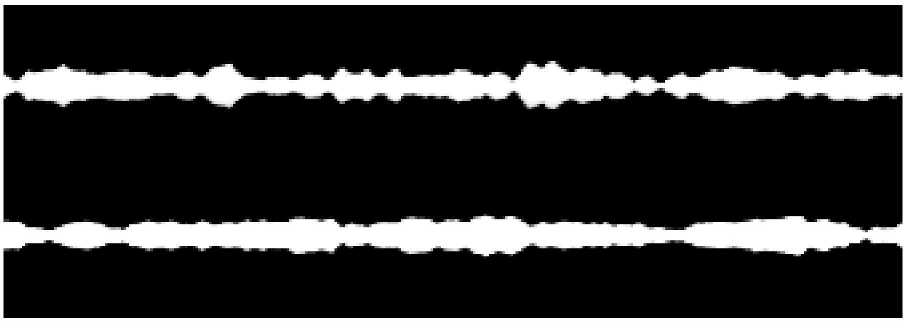


**FIGURE 3-10.** *A modern Dolby Stereo SR sound track. Four channels are matrix-encoded on the two visible Lt and Rt tracks.*

In the Dolby Stereo system, a center channel (C) and a surround channel (S) exist in addition to the usual left (L) and right (R) channels. Such a four-channel track can also be played in a theatre only equipped for mono or stereo sound.

The encoding scheme is depicted in the following conceptual diagram. The L and R inputs go directly to the Lt (Left total) and Rt (Right total) outputs. The center channel is attenuated by 3 dB and divided equally to Lt and Rt. The surround channel is also reduced by 3 dB and band-limited to a frequency range of 100Hz to 7 kHz. A modified form of Dolby type B noise reduction is applied, too.

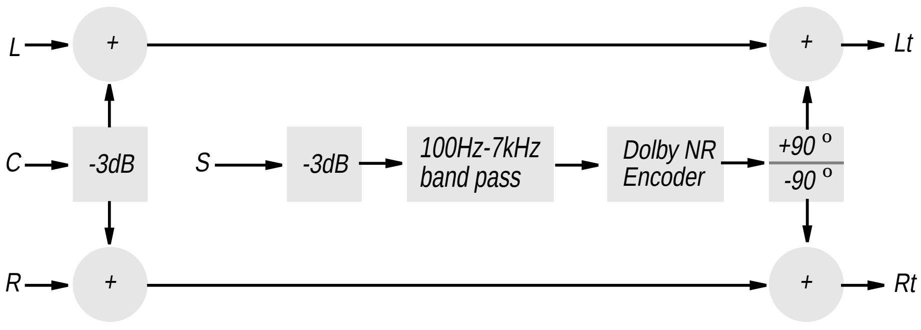


**FIGURE 3-11.** *A scheme of the Dolby MP (Motion Picture) Matrix, known as Dolby Pro Logic in the consumer area (from [37])*

The decoding process is as follows. The Lt signal goes directly to the left speaker, the Rt signal is passed to the right speaker. The center channel is left both on the Lt and Rt signal, so it forms a phantom sound between the left and right speakers. Additionally, the sum of Lt and Rt is fed to the center speaker for better localization of dialogue. Since the surround signal is recorded 180 degrees out of phase, it can be decoded by forming the difference between Lt and Rt. Additional 7kHz low-pass filtering is performed and inverse noise reduction applied, then the signal is fed to the surround speakers. The left and right speakers reproduce the surround signal, too, but as it is out-of-phase, it will be heard diffused [37].

The SR type of noise reduction usually applied to the Dolby Stereo tracks today exists since the late eighties. More information about this noise reduction scheme can be found in [40].

Among others, the papers [34], [35], [36] and [37] treat stereo and multi-channel optical sound tracks and recording techniques is more detail.

## 3.4 Push-Pull Sound Tracks

In the mid-thirties, the quest for better sound quality led to the development and use of slightly different optical sound tracks termed “push-pull” tracks. The idea was to record a second track of the same signal 180 degrees out of phase and then, during reproduction, feed the difference between the two signals to the amplifier.

This made the push-pull tracks less susceptible to in-phase errors, i.e. errors occurring on both tracks at the same time. Typical examples would include horizontal film splices and, very important, the area covered by the noise reduction shutters. This allowed shorter attack times without introducing audible frequencies, which, in turn, reduced the clipping effect and resulted in better sound quality.

Another advantage was lying in the broader tolerance towards sub-optimal exposure in printing. The distortions introduced by image spread (see 4.3.1 on page 28) on standard sound tracks are highly attenuated on push-pull type sound tracks.

The disadvantage of the push-pull system was its high requisition to precise adjustment and the more complex — and therefore more expensive — reproduction unit which had to use two photocells for the two signal parts. The incompatibility to the standard optical sound heads caused reluctance among the cinema owners to buy the new reproduction equipment necessary for playing the push-pull tracks. These new heads were backward compatible to the single tracks, however.

For all these reasons, push-pull tracks were never widely used for release prints, but they were very successful in the studios that used their better quality for original recordings. Mainly for this use, double width (200 mils instead of 100) class A tracks were developed as no space had to be spared for the pictures.

First, the class A type tracks were used. They were succeeded by the inherently “noiseless” type B tracks, which were even more difficult to adjust, thus leading to a slight variation of the B type tracks, the class A-B tracks. All of them are explained in more detail below.

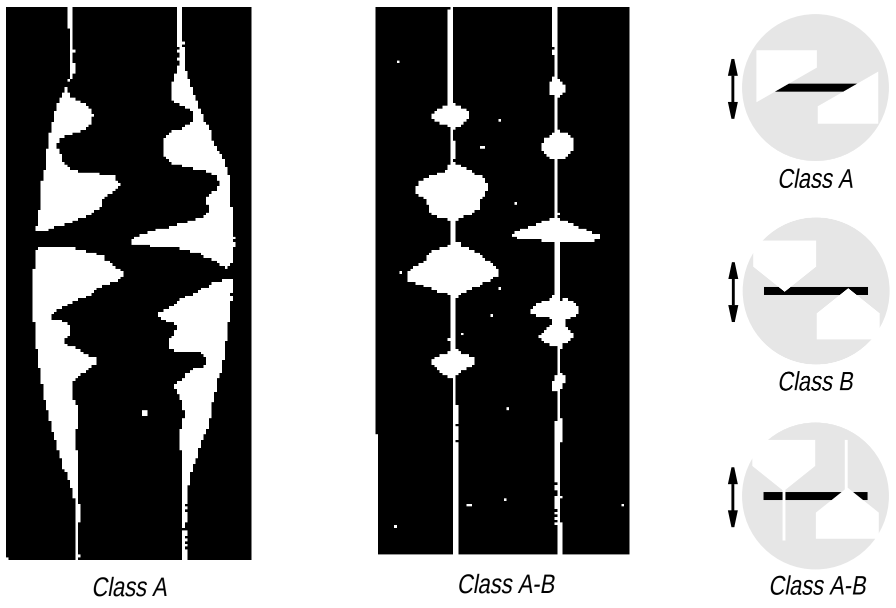


**FIGURE 3-12.** *Class A and class A-B tracks, the A-B track being basically a B track with narrow bias lines permitted to appear. The mask shapes are not independent, they always move together in the same direction.*

The basic recording scheme for push-pull tracks is the same as for the standard tracks. Here, too, the distinction between variable-density and variable-area tracks has to be made. Shown in Figure 2-11 are the masks used in conjunction with a galvanometer to record the variable-area representations of the push-pull types. Light valves (with three or four ribbons) have been used for class A tracks, too. They created variable-density push-pull sound tracks.

More about push-pull recording and reproduction can be found in [1], p.305f and p.348f, in [15] and in [29]. A mathematical analysis is available in [30].

### 3.4.1 Class A Tracks

Class A tracks can be imagined as two adjacent variable-density or unilateral variable-area tracks, one of them being 180 degrees out of phase. In the case of the variable-area type A example in Figure 3-12, this means that the right track is the opposite of the left one. A double-width class A track could be reproduced on a standard projector, because both tracks contain the same signal. Care must be taken, however, if shorter attack times were used for the noise reduction shutters. Reproducing only one side of the push-pull track could lead to audible low frequencies in this case.

### 3.4.2 Class B Tracks

In contrast to the class A, the mask shapes do not overlap in the vertical dimension for type B tracks. As a result, the positive part of a modulation wave is always on the left track, the negative part is always on the right track. In times of no modulation, the entire sound track width is effectively opaque, giving the ultimate in noise reduction. Exactly adjusting the apertures, such that no light can expose the film when no signal was present proved to be hard, so a slight modification was added and led to the class A-B type of track.

### 3.4.3 Class A-B Tracks

The presence of the added bias lines as shown in Figure 3-12 provides most of the signal-to-noise ratio of class B, but at the greater production ease of class A. The bias lines prevent clipping of low-level sounds by improper adjustment of the masks.

## *3.5 Other Analogue Photographic Sound Tracks*

Especially in the first quarter of this century, many more experimental formats have existed. I have picked two of these interesting early formats that had a certain importance at the time they were developed. One of them is particularly interesting from a Swiss perspective, the other is an early but advanced (1937) example of multi-channel sound. Very good papers about the history of motion picture sound tracks are [17], [18], [19] and [20], [21]. A more compact version can be found in [15]. More focused on the developments in Europe is [16].

### 3.5.1 Tri-Ergon

The “work of three”, as the greek name indicates, began in 1918, in Germany. It is interesting not so much because it was technically advanced — there were better systems at the time — but it was the system that was nearly spread in Europe by a Swiss consortium. I will only give a brief outline. See [9], 85A, for the whole story.

Technically, the sound track was some kind of a variable-density system and was placed outside the sprocket holes on a special film format having standard-sized pictures [20]. The system was patented in 1919 along with about seventeen other patents ranging from the photo cell to the fly-wheel taking out film speed variation during playback. These patents were so general that they caused extensive litigation over years, because practically no development in the area of optical sound tracks was possible without infringing one of the patents.

A financial group in Zürich bought the patents in 1923 for one million Swiss Francs. Some short sound pictures of about one hour total running time were produced and shown publicly at the Bellevue cinema in Zürich. The public preferred the silent movies and the cinema owners were reluctant to buy the new reproduction equipment needed to play these movies.

In 1924, Tri-Ergon is bought by another Swiss industrial consortium, and they signed a licence agreement with UFA in Berlin, limited to 15 years. The plan was to produce a major sound picture with support from the UFA. However, the production of Metropolis at the same time consumed enormous sums and nearly led to UFA’s ruin. Consequently, only a minor film of rather poor technical quality could be produced, which failed miserably. UFA was sold and Tri-Ergon, now consisting of merely five persons, was too small to act alone.

The patents were sold to Fox in the U.S. for $50’000.

### 3.5.2 Fantasound

The Fantasound system was developed for Walt Disney’s Fantasia. Work begun in 1937. It was the first to use a surround system.

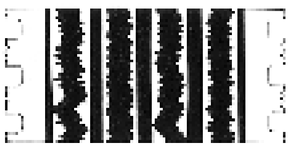


**FIGURE 3-13.** *The Fantasound system. Three signal channels and one control channel*

The original recordings were made on eight push-pull variable-area tracks. The aim was to have a much better reproduction of orchestra music than it was usually available on sound pictures at that time.

In re-recording, these eight tracks were mixed down to three push-pull class A variable-area tracks, that were accompanied by one push-pull control track with three frequencies (250, 630 and 1600 Hz) that controlled the volume of the three audio channels. See page 13 for the principle of control tracks. These four tracks were recorded on a separate 35 mm film and, of course, required a special reproduction unit. Probably because of this, this quite revolutionary format failed to gain acceptance.

It took nearly 40 years until surround sound and multi-channel optical sound tracks became widely used.

## 3.6 Digital Optical Sound Tracks

Starting in the early nineties, the analogue optical tracks covered so far were joined by new digital formats. Three of them are currently used, one is appearing in the very near future, and one has already ceased to exist. A short description of each is given in the following section.

THX is often regarded as a sound format, which it is not. In fact, it is a certificate of quality with respect to sound reproduction in the cinema. Not only sound quality is considered, but some other criteria, too. Size of the screen being one of them. There is, however, some additional hardware in THX certified amplifiers, mainly for increasing the surround channel's quality on analogue tracks.

All of the following formats can be present on a single release print, including a standard analogue type of optical sound, mostly Dolby Stereo SR or a compatible format.

### 3.6.1 Dolby Digital (DD or SR-D)

The Dolby Digital format is treated in greater detail as it was originally part of this thesis to extract and convert this format to audio, too. Some pre-processing steps could be performed, but detailed information about the exact format was not available from Dolby due to licensing restrictions.

Dolby Digital is one of the three widespread digital formats today (it exists since 1992). It provides six channels: left, center, right, rear left and rear right and a subwoofer. Because the subwoofer channel does not take up as much bandwidth as the other five channels, it is often termed a 5.1 channel format. The DD system does not require 5.1 channels however. It is possible to encode mono, stereo or stereo surround sources as Dolby Digital. All channels can have a resolution from 16 bits to 24 bits, 16 bits being the typical choice. The frequency range is 20 Hz to 20 kHz, the dynamic range spans over 90 dB.

Rather concise information can be found in [24], the original patent. It is included for reference on the accompanying CD-ROM. I will give a brief overview over the steps that would need to be performed for extracting the digital audio information from the digital picture obtained by the film scanner.

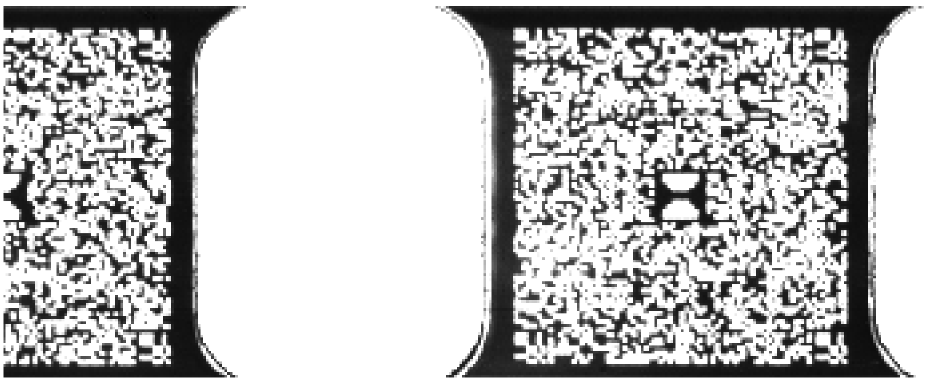


**FIGURE 3-14.** *Dolby Digital cells in the interperforation area.*

A 384 kbit/s AC-3 encoded audio stream is packaged into small 76-by-76 pixel (or fixel as Dolby calls them) cells, each fixel corresponding to a bit in the data stream. These matrices are positioned between the sprocket holes. See Figure 3-14 for an example. The fixel side length is 32 microns, corresponding to little more than three pixels at the current resolution of our scanner (one pixel at a resolution of approximately 2700 dpi has a side length of about 9.4 microns).

In the reproducer, an image of the fixel cells is acquired with a setup nearly identical to our scanner (see “Data Acquisition” on page 6). A Dalsa CL-C3-0512 CCD camera is recommended for this purpose, in conjunction with a lens system capable of imaging the 2.5 mm wide cells onto the 512 CCD elements. This corresponds to over 5000 dpi, nearly twice the resolution achieved by our current setup, but this optical resolution is later reduced to 256 electrical samples for one line

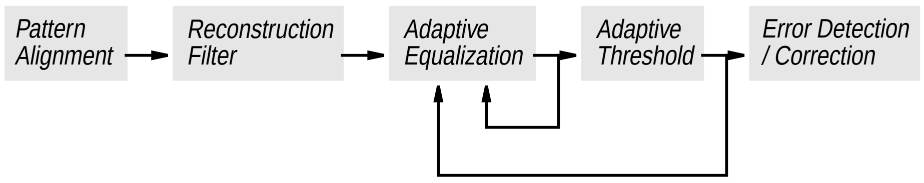


**FIGURE 3-15.** *A block diagram of the image processing part needed for decoding a DD sound track.(from [24])*

The fixels are arranged in a regular grid of known dimensions, so each fixel has a defined position. It is necessary, however, to know a reference point, relative to which the fixel positions can be calculated. This is achieved by four alignment patterns in the corners of a fixel cell. For a robust detection of the alignment patterns, these should be designed to have a low auto-correlation value with the rest of the cell. A cross-multiplied 7-bit Barker code, surrounded by a one-fixel wide black frame is used for this purpose (see Figure 3-16)

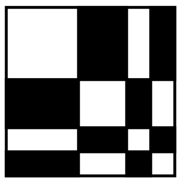


**FIGURE 3-16.** *The cross-multiplied 7-bit Barker code (1110010).*

This alignment scheme has been implemented in this project. For greater accuracy, not only one barker code mask was used, but 16 masks, each shifted by one fourth of a pixel in either direction. If one position of a top-left alignment pattern is known, the approximate location of all successive pattern can be calculated, and thus the area that has to be correlated to the Barker mask can be limited.

Correlation of the 16 alignment pattern inside that estimated area leads to a series of values, the highest of which corresponds to the best fit. As only one alignment pattern can exist inside the search area, we can simply look for this highest correlation value and memorize which pattern was used. This is done for all four corners of a cell. With these coordinates, the location of the single fixels can be computed, assuming that the film speed was linear while scanning the cell. This step corresponds to the first block in Figure 3-15.

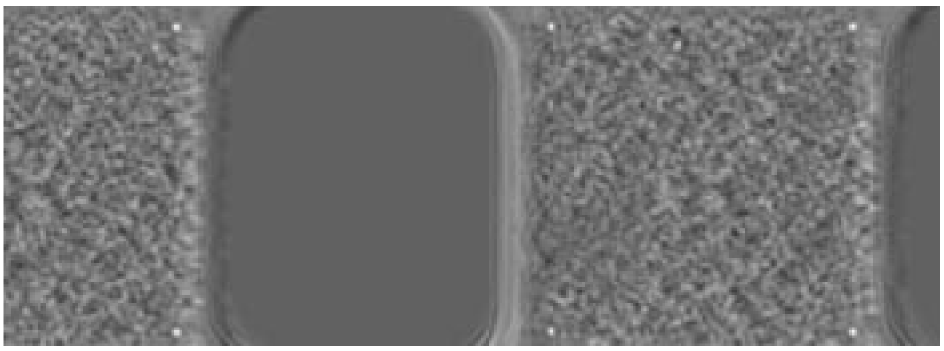


**FIGURE 3-17.** *A correlation map for the image shown in Figure 3-14. The distance between the highest intensities corresponding to the alignment pattern position and other high correlation values is larger than it appears on this printed picture.*

As a next step, the reconstruction filter has to be applied. It is a 3-by-3 two-dimensional FIR filter that has to be constructed from a set of twelve coefficients as explained in [24] in order to take the offset of a pixel to the exact fixel center into account. This offset is also calculated at a resolution of one fourth of a pixel, similar to the Barker masks above.

After this upsampling stage, adaptive equalization is applied to reduce interference between the fixels due to the rather low resolution (around 1.5 times the Nyquist sampling rate). See [24] for more details about this processing stage. The equalization performed adapts to the output of itself and the successive adaptive threshold.

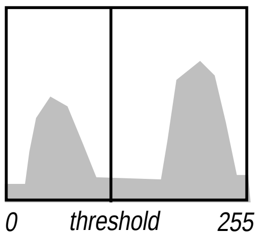


**FIGURE 3-18.** *A typical histogram of a black-and-white image with the empirical location of an adaptive threshold.*

The adaptive threshold works empirically, and I have used nearly the same technique for an adaptive threshold estimation algorithm for analogue sound tracks (see “Removing Film Grain — ContrastEnhancer” on page 53).

Scanned images of black-and-white originals without any gray-scale values (such as the variable-area sound tracks or a fixel cell in this case) result in a histogram as depicted in Figure 3-18. One peak corresponds to black, the other corresponds to white. A value between the two peaks is chosen as the threshold. Values closer to the “black” peak give more pleasing results.

On the binary representation of the cell, the fixel values can be read at the calculated positions and be grouped into bytes. An area of two fixels in width and four fixels in height corresponds to one byte. This dimension was chosen because scratches usually occur in the direction of film travel and hence affect less bytes when they extend vertically.

The byte stream is then passed on to the error detection and correction stage. Reed-Solomon codes are used for this purpose.

Additional error detection is done for entire fields of fixels (blocks) and each block is marked “good” or “bad”. If too many bad data blocks are encountered, the sound output is switched back to the analogue stereo variable-area track usually present on the same release print.

The audio data blocks are fed to an AC-3 decoder. More details of the AC-3 perceptual coding technique (also used for DVDs) are available in [41].

### 3.6.2 Dolby EX

This is the newest digital format, offering 7 channels: left, center, right, rear left, rear center, rear right and a subwoofer channel. “Star Wars — Episode I” will be the first picture to feature this type of sound track. It is not really a new format, however. It is basically the Dolby Digital format covered in the previous section, where the two rear channels have been matrix-encoded as described in “Stereo Variable-Area Tracks” on page 15. These two channels are additionally fed into a Dolby Pro Logic decoder after the AC-3 decoding step which produces the additional rear center channel.

### 3.6.3 Sony Dynamic Digital Sound (SDDS)

This format offers 8 channels for left, half-left, center, half-right, right, rear left, rear right and the subwoofer. It uses also transparent and opaque “fixels” to encode the bits of the ATRAC encoded audio stream, but doesn’t group them into cells. The fixels are positioned in two redundant strips outside the perforation area along the film edge.

Disadvantages of the SDDS format are the restrictions for the copy velocity when producing a release print. A speed faster than 24 frames per second is not possible without sacrificing reliability in reproduction.

All channels have a resolution of 16 bit and a dynamic range of over 90 dB. The seven main channels offer a frequency range of 20 Hz up to 20 kHz.

This format exists since 1993. One of the first motion pictures featuring this type was “Philadelphia”.

### 3.6.4 Digital Theatre Sound (DTS)

DTS is not an optical sound format. The sound track is recorded in the compressed CAC format on multiple CD-ROMs. Each CD-ROM can hold 100 minutes of film sound. In order to synchronize the audio data with the film, the DTS time-code is optically recorded on the film, between the analogue sound track and the picture frames.

All channels have a resolution of 20 bit and a dynamic range of over 96 dB (theoretically up to 122 dB).

### 3.6.5 Kodak Cinema Digital Sound (CDS)

The earliest of the digital formats was developed by Kodak and offered six channels. Unfortunately this digital format was positioned at the location normally used by the analogue track. Therefore, copies featuring CDS sound could only be played in cinemas equipped for this type of sound, which was the very likely reason, why this format failed adoption and disappeared after only one film was produced (Terminator 2).

CHAPTER 4

# Looking at the Error Chain

Most of the research done earlier in this century in the field of optical sound recording was devoted to the study of different errors types. Some of them are inherent to the system, others are due to mechanical limitations that improved over time. For the digital restoration effort, it is necessary to have an overview of the possible error sources.

## 4.1 Overview

The production of an optical sound track from the original recording to the final release print is a multi-stage process, and each stage introduces its own typical errors. A schematic view of the error chain is given in Figure 4-1. Most errors are caused by the equipment but external sources, such as dust, can contribute, too.

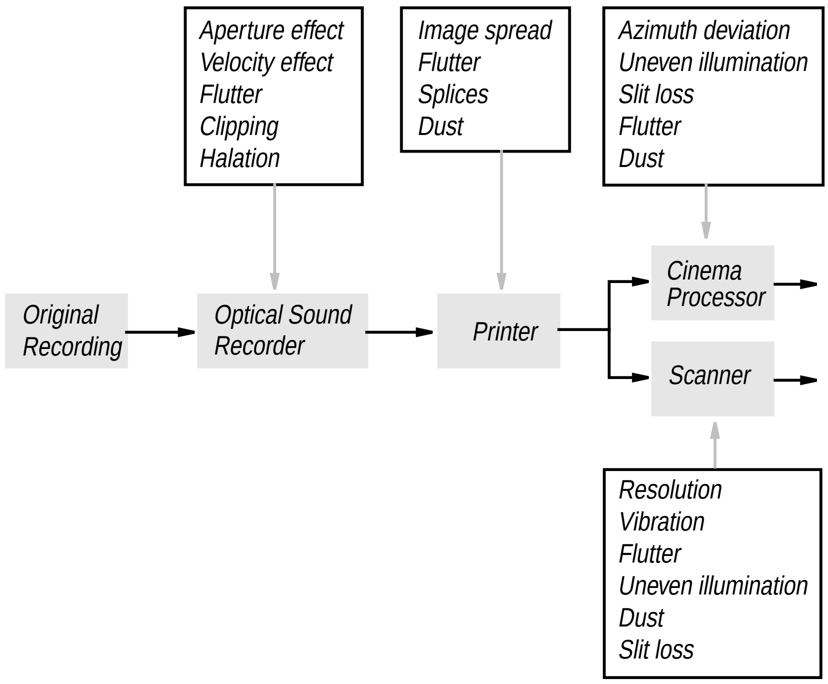


**FIGURE 4-1.** *The error chain. Longer-term defects such as shrinkage and scratches are not displayed*

The following sections each give a brief overview of the different type of errors optical sound tracks are subjected to. References are given for sources providing deeper insight into the topic.

The errors introduced by scanning the film receive special attention. Before trying to restore defects of the film, the additional errors introduced by doing so should be known.

Some of the errors described below can occur in different stages. I have included them in the stage where they are most likely to appear. The distortions occurring while recording the original or the ones introduced by the amplifier and speakers are not treated.

## *4.2 Recording the Negative*

The sound track is usually recorded on a sound negative first. This is a standard 35 mm film base with only the sound track exposed. The input for the galvanometer or the light valve can come directly from a microphone or from any other sound source. At this stage, inherent errors of the recording system are the main source for distortion.

### 4.2.1 Aperture Effect

This type of defect solely applies to variable-area recordings. Only for an infinitely narrow recording slit (aperture) the true amplitude is recorded. For a finite aperture width, the area rather than the amplitude of a sound wave is recorded, thereby introducing distortion, especially in the high frequencies.

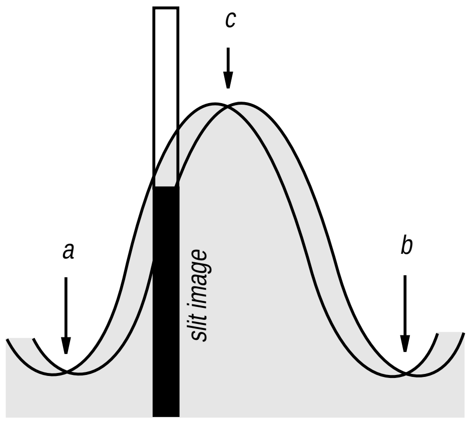


**FIGURE 4-2.** *Actual shape recorded by a finite slit width for a sinusoidal wave. The sharp valleys (a, b) and flat peaks (c) introduce distortion. (from [1])*

The effect is shown in Figure 4-2, the exposed area being shaded while the original signal curve is shown in black. Severe distortion will result from the flat peaks and the sharp valleys (a, b, c) if the slit width is chosen too wide. The slit width should therefore be about ten times smaller than the wavelength of the highest frequency being recorded. See [1], p.355 for more details.

### 4.2.2 Velocity Effect

The velocity effect applies to both galvanometer and light valve recording systems and is caused by the moving parts (galvanometer mirror or light valve ribbons). However, the effect is different for light valves and galvanometers.

When high frequencies are recorded with a galvanometer, the shape exposed on the film does not equal the signal wave form. The reason for this is the finite velocity with which the light beam moves across the slit. The amount of exposure is governed by three factors:

1. The time required by a particular point on the film to travel past the light beam
2. The intensity of the exposure light
3. The time it takes the exposure light to move from its initial position to the desired amplitude and back again

The film speed and the illumination intensity are constant, so the exposure is determined by the speed of the light beam as it travels along the slit. Therefore, the signal peaks of high frequencies are not fully exposed, leading to attenuation of high frequencies and distortion.

For variable-density tracks, the “ribbon velocity effect” describes the distortion and attenuation of high frequencies introduced by the motion of the ribbons in the direction of film travel. See [1], p.298 for a mathematical analysis of the ribbon velocity effect.

### 4.2.3 Clipping

Clipping of signal peaks can occur for two reasons. Either the recorded signal is already overmodulated, which causes the slit width to be fully exposed over a longer time when the galvanometer mirror is fully deflected or the light valve ribbons clash. This is not an inherent problem of the optical sound recording system, however.

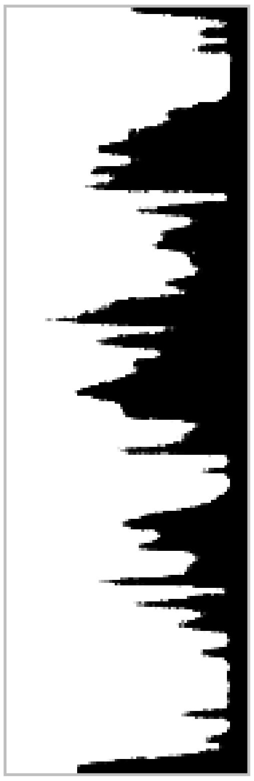


**FIGURE 4-3.** *Clipped peaks due to overmodulation.*

Another type of clipping was caused by earlier types of noise reduction shutters used in conjunction with galvanometers (see “Mono Variable-Area Tracks” on page 13). These shutters were not allowed to move too quickly, in order not to introduce audible frequencies. However, when they were shut and the signal suddenly was present again, the first 20 ms or so were clipped, resulting in distorted reproduction. Later these were replaced by newer systems that recorded the sound with a delay, so the noise-reduction shutters could look ahead and react to the signal in time.

This type of defect would be suited for further image based restoration efforts. The clipped peaks could be approximated by a sine wave, so as to reduce the distortion effect.

## 4.3 Printing the Positive

Release prints are made by copying the sound negative and the picture negative together on a positive. Today, this is usually achieved by contact copying both negatives at the same time onto the positive. The three most serious defects that can occur in this stage are presented below.

### 4.3.1 Image Spread

The perfect variable-area sound track would be completely opaque in the dark portions, completely transparent in the light portion, with a sharp boundary between them. Light scattering in the optical system and in the emulsion prohibits the production of such a high definition sound track negative. The exposed area is a little bigger than the actual wave form and semi-transparent. During the printing process, the shape of the reproduced wave becomes dependent upon the exposure time. In the perfect case, the exposure time wouldn't matter. The effect is visible mainly for high frequencies above 4 kHz (see Figure 4-4).

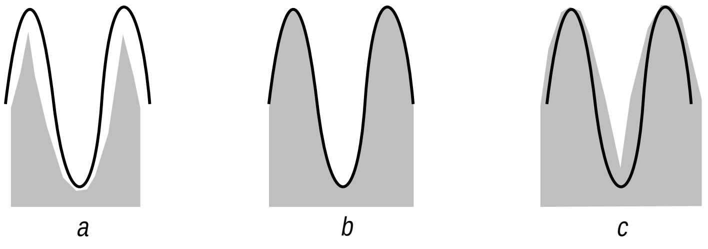


**FIGURE 4-4.** *Image spread on the negative. Under-exposure leads to image contraction (a), over-exposure leads to image growth (c). A correct exposure is shown in (b).*

Case (b) could be achieved on the negative and be preserved for the positive, but due to emulsion characteristics, a very low density of the positive print would result, which in turn leads to a high noise level (film grain) and low output. So the negative must have a higher density, but that leads to image growth as depicted in (c).

When exposing the positive, the same effects happen, so the image growth resulting from a higher density on the negative can be compensated by over-exposing the positive print.

In 1938 a method was presented in [28] to determine the correct exposure time for compensating the image spread of the negative, which would otherwise introduce severe harmonic distortions. This method is still in use today and is often termed "cross-mod".

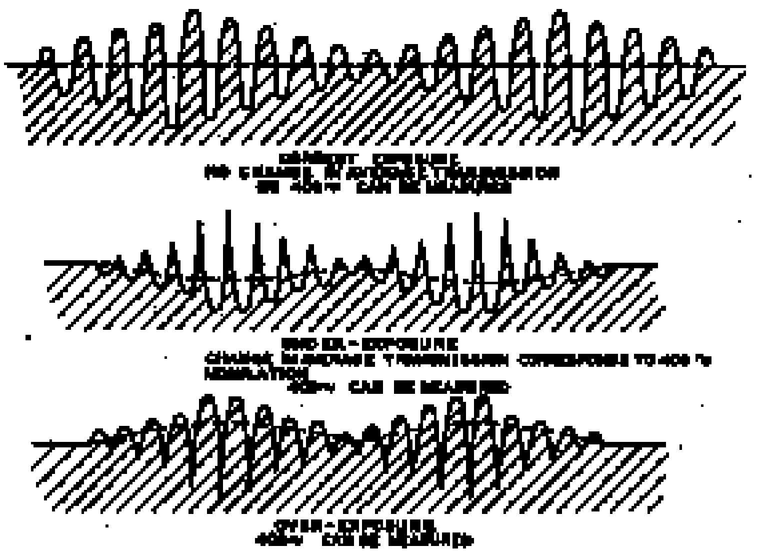


**FIGURE 4-5.** *Average transmission resulting from different exposure times is the key to the cross-modulation method. (from* [28])

The idea was to modulate a high frequency tone (9kHz or 6 kHz) with 400 Hz, resulting in an optical sound track like the ones depicted in Figure 4-6. Of this cross-modulated tone, negatives of different densities can be made. Afterwards, each of the negatives is copied to a positive with different exposure times, resulting in a series like the one in Figure 4-6.

The positives can now be reproduced on a high-quality unit, the output being passed through a band-pass filter which attenuates everything except the 400 Hz frequency. If the printing density was chosen correctly, the average transmission as displayed in Figure 4-5 is constant. If the track was over-exposed or under-exposed, a measurable 400 Hz tone will be produced.

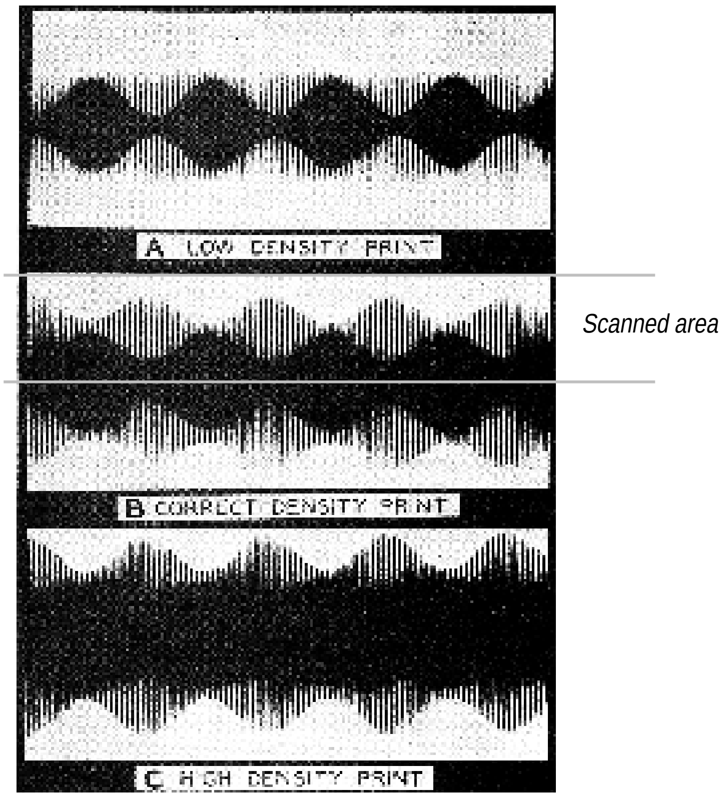


**FIGURE 4-6.** *A typical set of cross-modulation prints. When scanning only half the track width, optimally no 400 Hz output should be measured because the average light transmission is constant in this optimal case (B). (from* [28])

The amplitude of the 400 Hz output can be plotted against the negative density for a specific positive print. This results in curves like the ones shown in Figure 4-7.

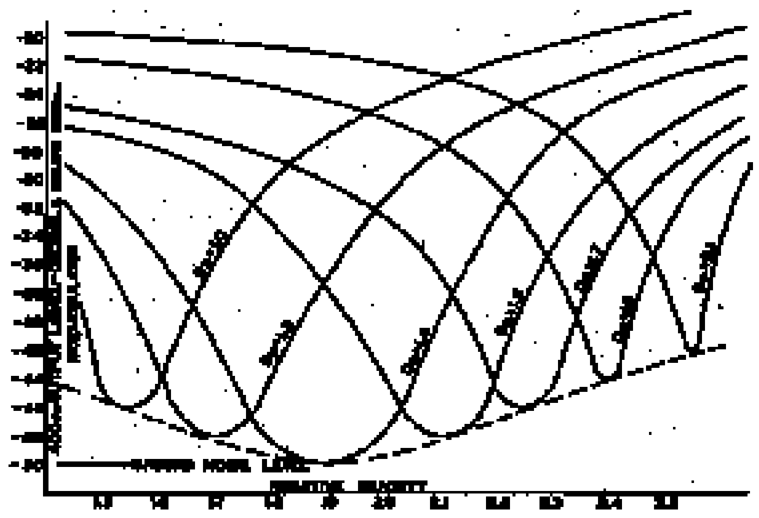


**FIGURE 4-7.** *400 Hz output for various combinations of negative and positive density. In this case, a negative density of 1.9 together with an exposure time leading to a positive density of 1.4 was optimal. (from* [28])

Image spread is an important factor when restoring old negatives. Positives were always used in this project, but the effect of image growth must be simulated digitally when negatives are to be processed. If both positives and negatives of and old motion picture sound tracks are available, better results can probably be obtained from the negative if image spread is taken into account. Otherwise serious distortion will be produced due to the pre-distorted negatives.

### 4.3.2 Splices

Junctions of two film parts are problematic for the standard sound track types. They lead to a loud click, because an intensity jump is usually associated with a splice. Sometimes, blooping, as described in “Traditional “Blooping”” on page 3, was used to avoid the clicks.

It would be possible to remove a sudden transparent line across the track width on the image. However, care has to be taken so as not to employ a traditional audio restoration algorithm without really taking advantage of the additional information present in the image. In this example, checking if there is an all-white line (or multiple lines) and the previous line was much lower in intensity would be an adoption of the slew-rate filter used for removal of clicks in audio data.

Further studying of real splices and their reproduction by the scanner would be necessary to determine if this type of error is better left to subsequent audio processing or should already be done on the image.

### 4.3.3 Flutter

Flutter is defined in [1], p.421 as “The term flutter relates to any deviation of frequency that results from irregular motion in the recording or reproduction of a constant-frequency tone”. This section treats flutter in the stage of recording and printing. Flutter introduced by the scanner is treated separately in 4.6.3 on page 35.

Flutter can have various sources (see [1], p.439).

1. The film is wrapped around a slightly eccentric drum.
2. Irregularities of motor speed
3. Engaging and disengaging of the sprocket teeth introduces a 96 Hz variation.
4. Perforated film doesn’t move as a perfect belt around the recording drum. It travels more like a polygon, being bent the most at the center of the sprocket holes.

Flutter caused during the recording process can’t be removed on the image, because there are no reference points that vary accordingly.

## 4.4 Reproducing in the Cinema

The reproducing equipment is often not as well set up as the production equipment, leading to some additional types of defects, but especially azimuth deviation could occur during the recording stage, too.

### 4.4.1 Azimuth Deviation

Azimuth deviation means that the recording or reproducing slit is not at right angles to the film edge. The angular scanning of the sound tracks leads to high-frequency loss in the case of variable-density prints and introduces harmonic distortion in the case of variable-area prints.

This type of defect was present on one of the examples I was working with and could be corrected on the image. For more details about azimuth deviation and the restoration method, see “Deskewing the Image” on page 62.

### 4.4.2 Slit Loss

The scanning slit used to reproduce the sound has necessarily a finite height, which leads to loss of high frequencies, but no distortion is introduced.

The loss (in dB) for a frequency *f*, caused by a specific slit height can be computed as follows (from [1], p.539):

$$loss = 20 \cdot \log \frac{\sin \pi\left(\frac{f}{f_0}\right)}{\pi\left(\frac{f}{f_0}\right)} \qquad \text{(EQ 1)}$$

where $f_0$ is the cutoff frequency determined by

$$f_0 = \frac{v}{2\beta} \qquad \text{(EQ 2)}$$

Here, $v$ is the film velocity (457 mm/s) and $2\beta$ is the slit height (in mm, too).

One pixel at the resolution currently achieved by the scanner corresponds to a slit height of approximately 10 micron (0.01 mm). The resulting cut-off frequency is therefore 45’700 Hz which is by far sufficient for all types of analogue sound tracks. Attenuation of a 16 kHz tone (the utmost limit for Dolby Stereo SR tracks) would be -4.2 dB.

In the current implementation (see “Image to Audio Conversion — PhotoCell” on page 66) the slight height can be chosen in pixel increments. Slit loss for different values is displayed in Figure 4-8.

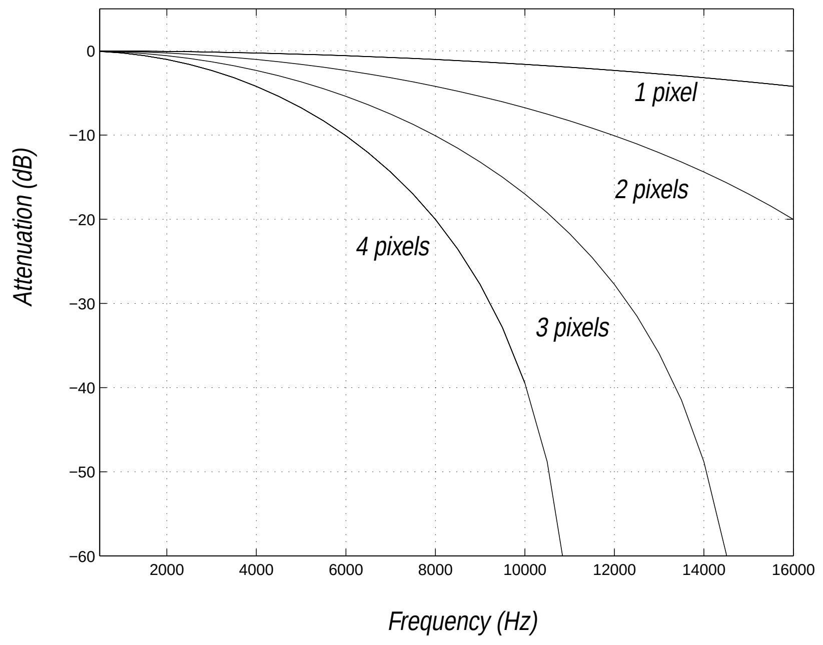


**FIGURE 4-8.** *Slit loss for different choices of slit height (in pixels) at the current scanner resolution.*

## *4.5 Deteriorations Caused by Age*

### 4.5.1 Shrinkage

Shrinkage affects mostly old nitrate-based stock. Judging from the “Ciné-Journal” example, shrinkage occurs mainly transversal to the direction of film travel. The effect on sound quality is a lower volume, and depending upon the degree of shrinkage, other film parts moving into the photo cell’s field of visibility can deteriorate the sound quality if reproduced on a projector. See “Restoration Example” on page 73 for a similar type of defect.

As long as the shrinkage is uniform, it can easily be handled on the image, because no assumptions whatsoever are made regarding the position or dimension of the sound track. Even a shrinkage in the direction of film travel, which would normally alter the frequencies if it could be reproduced at all, is easily accommodated, because the sprocket holes are used to determine the running time and thus the sampling rate.

### 4.5.2 Other

Mechanical defects, such as missing parts and cracks, can possibly be restored in the case of (multiple) symmetric sound tracks, by using the redundancy of the track shape. This method is more closely described in “Dust Removal III — SymmetryEnforcer” on page 59 as a method for dust removal. Defects on a larger scale could only be made less disturbing by avoiding the clicks resulting the sudden transparency of the sound track area.

## 4.6 Scanning the Film

The scanner suffers basically from the same type of problems as the reproduction unit in the cinema, but there are a couple of additional restrictions and problems to be aware of.

### 4.6.1 Resolution

The current resolution of the scanner is nearly 2700 dpi. At this resolution, about 48500 image lines make up one second of sound. Thus, the resulting sampling rate is 48500 Hz, which is by far sufficient, especially considering that the old sound tracks had a limit of 8 kHz and seldom exceeded 5 kHz.

Consider variable-area tracks first. Theoretically, only a transparent and an opaque part exist on one line, and nothing in between. The standardized width of the area scanned by the photo cell in a projector [49] corresponds to 196 pixels at the current resolution. If a single threshold is set on the image, the audio quantization is limited to $log_2(196) = 7.6$ bits. Taking the instability of the least significant bit into account, an audio resolution of not more than 7 bits results. That leads to quantization noise and is therefore not sufficient.

The intensity of a pixel at the border between opacity and transparency can be regarded as a measure of how much of the CCD area was covered by the actual black-and-white edge on the film.

The audio resolution could then be modeled as

$$bits = \log_2(w \cdot (t_{high} - t_{low})) \qquad \text{(EQ 3)}$$

where $w$ denotes the sound track width (modulation area) in pixels. The upper and lower thresholds set on the image are denoted by $t_{high}$ and $t_{low}$. From this equation, a maximum theoretical quantization depth of 15 bits results for a track width of 196 pixels and a range of 255 gray-scale values.

For the actual restoration examples, the difference between the two thresholds was typically 40-60 units, corresponding to an audio quantization depth of roughly 12-13 bits.

More problematic are the variable-density sound tracks. The CCD is limited to 8-bit depth, which is usually sufficient for images as the human eye can’t differentiate more shades of gray. In this case, however, the density range of the film corresponds directly to the audio resolution and can’t be covered by 8 bits. Usually not even the full range of 8 bits is available for the sound track, especially when only one scan of both picture and sound track is to be made.

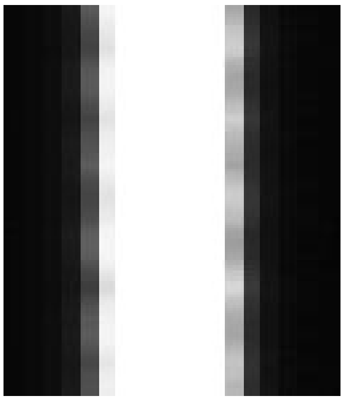
As a conclusion, the current resolution is sufficient, but not optimal, for variable-area tracks and insufficient for variable-density tracks. The higher resolution proposed in [11] should alleviate both problems. Especially a 16-bit CCD would provide a large enough range to image both the picture and the sound part at the same time with the depth they need.

### 4.6.2 Vibrations

In order to reach the higher resolution, the lens system had to be elongated by two extension rings. Due to the increased length, it became more susceptible to vibrations caused by the ventilator of the light source and the motor drive.

As the camera may not move with respect to the film plane, both camera and the film transport are directly joined by a strutting which conveys the vibrations to the camera and the lens system.

Even on a still scan without transporting the film, the vibrations are visible (see Figure 4-9).



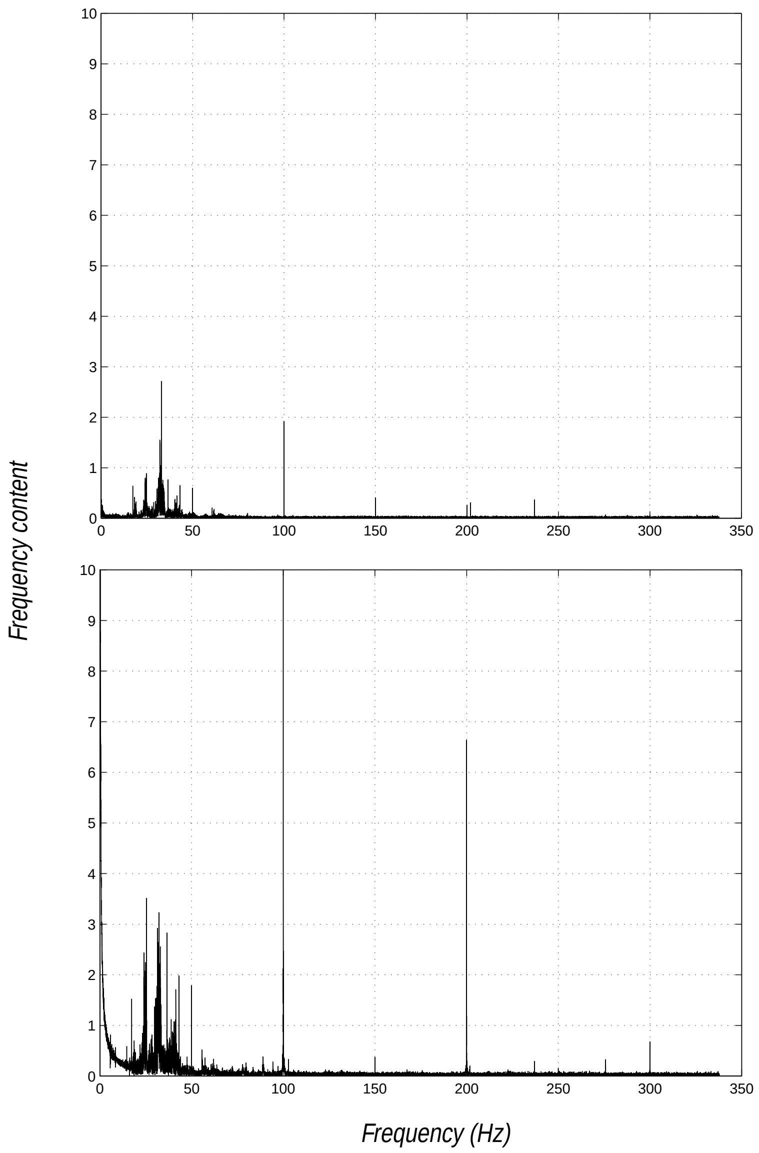


**FIGURE 4-9.** *Effect of vibrations in the picture (as visible above) upon the audio output. The upper diagram shows the frequencies measurable without the motor drive, the lower diagram shows the frequencies with the motor drive running. In both scans, the film was not transported. The frequencies shown are the "real-time" frequencies, as they were present during the scan. In this case, they would have to be multiplied by 72 to get the audible frequencies when playing the WAV file.*

Various tests have been made, and these results were reproducible. They proved to be independent from the scan parameters, such as the clock divisor or the exposure time. The vibrations could partly be damped by applying weights to the struttings. Underlying dampening materials such as styrofoam under the scanner wheels resulted in no improvement.

If the WAV files resulting from this still scan are played, a humming noise is audible. The effect of these vibrations is not very pronounced, however, and is usually masked by the signal.

### 4.6.3 Flutter

Flutter affects the scanning, too. An eccentricity of the drive axle, leading to 3 Hz flutter, was already known.

In order to investigate other sources of flutter, a test film with a recorded 8 kHz signal was used. The first thought was to use a sliding-window FFT to determine the frequency in short intervals and plot the deviation from the 8 kHz reference. However, the window width determines the fundamental frequency (sampling frequency / window width) of which the FFT only can detect multiples. A wider window will miss local frequency variations. Due to this reciprocal relationship, another approach was chosen. Basically, it has the same problems as just described, but the frequency resolution is better.

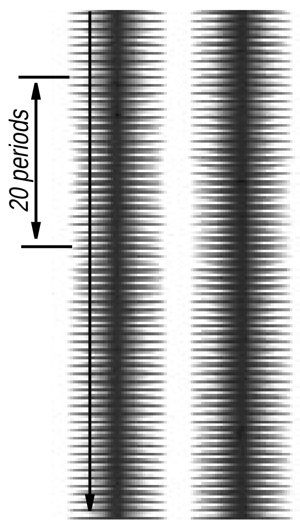


**FIGURE 4-10.** *The traced line on the 8 kHz reference signal. A threshold is applied to the image first.*

The idea was to trace a line through the signal peaks on a scan of the 8 KHz reference signal as shown in Figure 4-10 and determine the length of 20 periods, for example. The frequency f can then be deduced from the sampling rate s, the number of periods p and the length l of the p periods in pixels.

$$f = \frac{s \cdot p}{l} \qquad \text{(EQ 4)}$$

The horizontal position of the line should not matter within certain limits. The distance from peak to peak is the same, regardless where it is measured. Variations in exposure should have no influence, either, because the peaks get narrower or wider if the exposure changes, but their position relative to each other doesn't change.

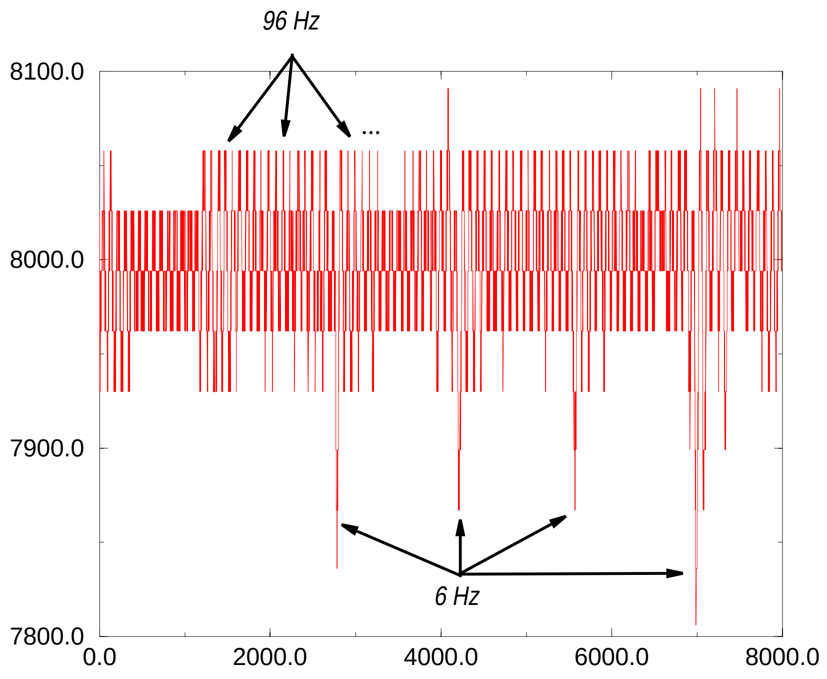


**FIGURE 4-11.** *Flutter for the 8 kHz test film example. Resolution is limited to about 30 Hz by the number of periods counted.
Clearly visible are the peaks corresponding to the 96 Hz and the 6 Hz variation.*

Here, too, the number of counted periods limits the frequency resolution. A variation of one pixel amounts to a difference (in Hz) of

$$\Delta f = \frac{s \cdot p}{l} - \frac{s \cdot p}{l+1} \qquad \text{(EQ 5)}$$

20 periods of the 8 kHz tone at a sampling rate of 48500 Hz amount to a length of about 121 pixels, limiting the accuracy according to (EQ 5) to 65 Hz.

This method can only give a rough impression of the frequency variations occurring over time (displayed in Figure 4-11). A frequency analysis of the data produced by the program doing the period length counting has delivered two main frequencies: 96 Hz and 6 Hz. Unfortunately, I could not determine if the 96 Hz flutter has already been recorded on the test film — it is a usual phenomenon due to the unsteady transportation by the sprocket teeth — or if it is introduced by slippage of the film when the rubber wheels pressing the film to the drive axle pass partly through the sprocket holes.

### 4.6.4 Exposure

Time was not available to investigate the effects of different exposure times on the audio quality. Probably a similar technique some sort of cross-mod as presented in “Image Spread” on page 28 could be used, too. However, a clearly defined reference is necessary for this task. Some experimentation with the 8 kHz has showed increasing second harmonics for very high exposure times.

I chose an exposure time that would be suitable for a scan of the whole film width, including the picture, too. This conforms to the original intention of the scanner.

CHAPTER 5

# Restoration and Conversion to Audio in an Object-Oriented Framework

As we have seen in the last chapter, the optical sound track is subject to numerous defects, the ones actually occurring varying from film to film. To cover all of them in a single program is an impossible task.

Restoration software should therefore not only be easy to configure in order to match the specific needs of a deteriorated movie sound track. It should also be easy to expand, in order to accommodate a different type of defect.

In this chapter, I suggest a modular C++ framework allowing the quick introduction of a new restoration algorithm without knowing too much about the rest of the software. The general overview of the modular structure is followed by a description of the implemented modules.

## 5.1 Overview

The framework is based on the notion of a block traversing many stages, each performing its task on it and passing it on to the next stage. Actually, blocks aren’t passed on to the next stage. Each stage requests a block from the previous one. See Figure 5-1 for a graphical representation of this pipeline structure.

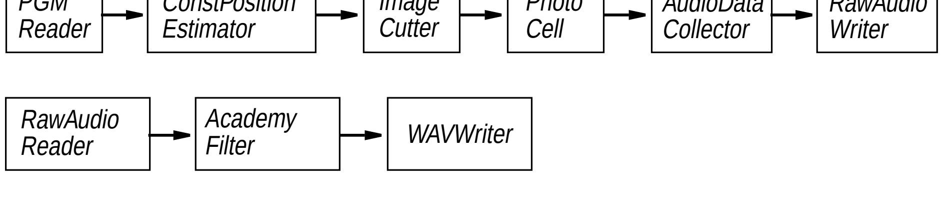


**FIGURE 5-1.** *A typical pipeline extracting the analogue optical sound track from a PGM image file without performing any restoration tasks.*

As you can see, the pipeline is divided into two parts. This is necessary, because the *WAVWriter* and possibly additional modules need global information on the audio data they are about to receive, such as the sampling rate and the maximum and minimum amplitude. These values are only known after the whole image has been processed, which makes it necessary to generate the wave file in two runs.

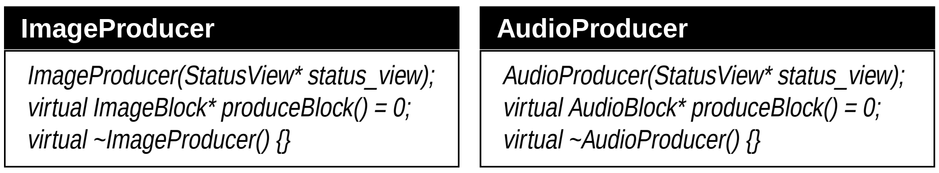
Four main classes build the foundation of this framework. These are *ImageProducer*, *AudioProducer*, *ImageBlock* and *AudioBlock*. Both *ImageProducer* and *AudioProducer* are mainly interfaces, not classes with their own functionality. As their names already indicate, *ImageBlock* and *AudioBlock* are classes containing the image or audio data, along with the block dimensions and methods for reading and setting them. Classes based on *ImageProducer* must define a method *produceBlock* which does the main work and eventually delivers an *ImageBlock*. *AudioProducers* deliver *AudioBlocks* the same way.

See Figure 5-2 for the public class interfaces of *AudioProducer* and *ImageProducer.* A short description of ImageBlock is given in Section 5.2 on page 41. The description of *AudioBlock* follows in Section 5.16 on page 65.

What subclasses of *ImageProducer* or *AudioProducer* normally do in the *produceBlock* method is reading an *ImageBlock* or *AudioBlock* from a previous *Image-* or *AudioProducer*, altering its content, and returning it to the caller — generally some kind of producer, too. This structure allows to plug different modules together, thus forming a pipeline like the one drawn in Figure 5-1. In order to link the different modules, a pointer to the previous producer is usually being passed as an argument to the constructor of the following *Image-* or *AudioProducer.*

*AudioProducers* and *ImageProducers* have a lot in common. It would have been possible to define both of them as derived classes of a *Producer* base class; but then they would have to act on a more general *Block* class, too. This added generality has no functional advantages, however, so I stuck with the separate base classes. Listing 5-1 shows a template for the most common case.

**FIGURE 5-2.** *The foundation of the implemented framework. Base classes and their public interfaces.*



**ImageProducer**

```
ImageProducer(StatusView* status_view);
virtual ImageBlock* produceBlock() = 0;
virtual ~ImageProducer() {}
```

**AudioProducer**

```
AudioProducer(StatusView* status_view);
virtual AudioBlock* produceBlock() = 0;
virtual ~AudioProducer() {}
```

Once the modules are glued together, they need a driving force that pulls the blocks through the pipeline. This is usually achieved by a loop in the main program calling the *produceBlock* method of the last producer in the chain until it returns a null pointer.

The method *produceBlock* of this last block will call its predecessor's *produceBlock* which in turn will call its predecessors *produceBlock* and so on. This means one block is treated entirely before the main calling routine gains control again. Sometimes it is necessary, however, to do something between two stages in the pipeline, e.g. update a status bar, or print out some statistics of the last processing step.

Therefore, and in order to avoid code in the classes writing to *stdout* directly, a class *StatusView* was introduced. An instance of this class or of a class derived from it should be passed to the constructor of ImageProducers and AudioProducers. Thus, there is a defined way of communicating to an instance outside the pipeline. Producers can use it to indicate that they are done with a block or to display an error message.

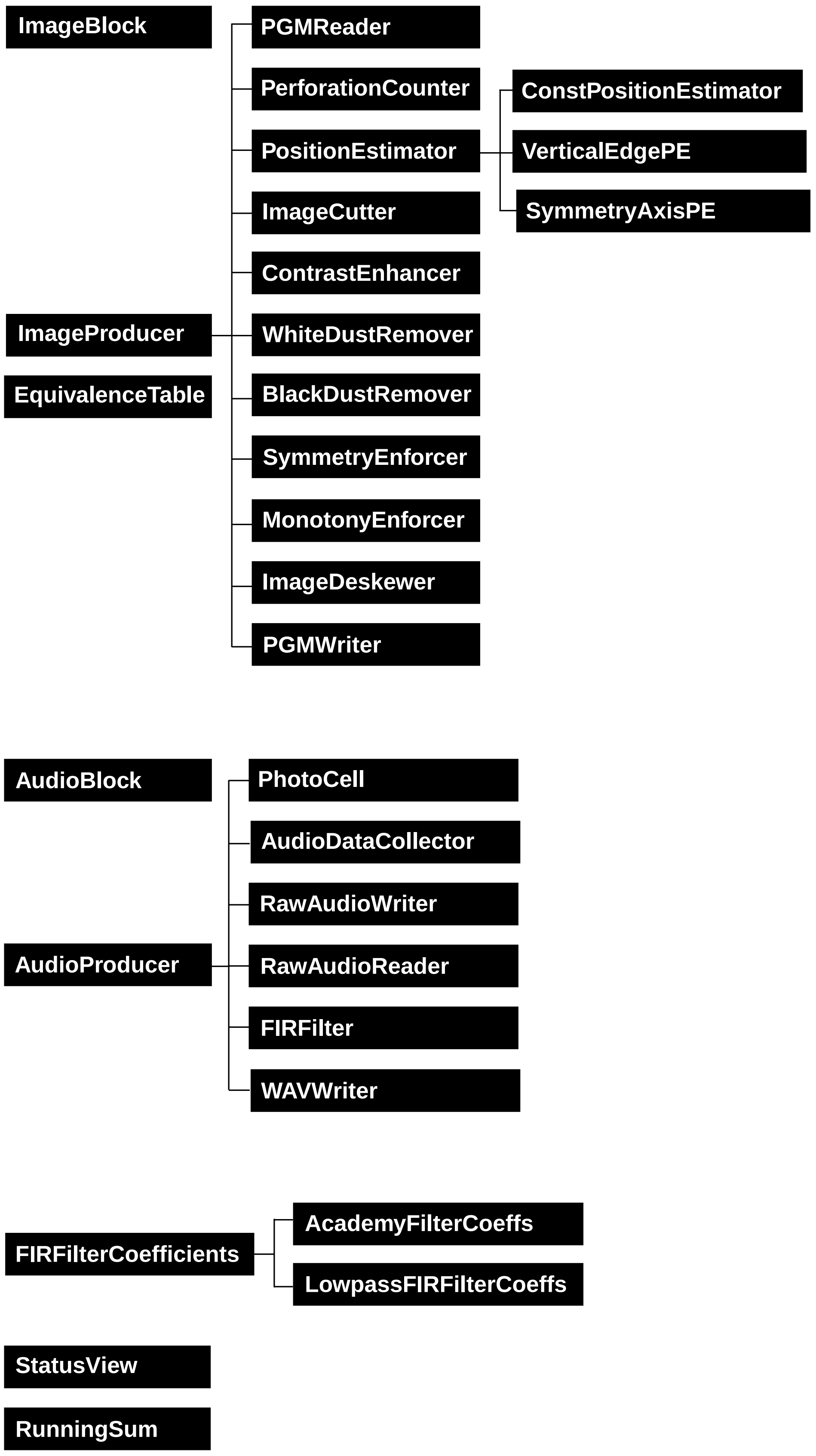


**FIGURE 5-3.** *The class hierarchy of the implemented C++ framework.*

**LISTING 5-1.** *Standard template for an ImageProducer descendant*

```
#include "ImageProducer.hxx"

class NewImageProducer : public ImageProducer {

public:
  NewImageProducer(ImageProducer* producer, StatusView* sv);
  ImageBlock* produceBlock();
  ~NewImageProducer();

private:
  ImageProducer* m_source;
  // task specific data here
};


NewImageProducer::NewImageProducer(ImageProducer* producer,
StatusView* sv) :
  ImageProducer(sv),
  m_source(producer)
{
  // do initialization
}


ImageBlock* NewImageProducer::produceBlock() {
  if (status != 0) status->updateSubtask();

  ImageBlock* block = m_source->produceBlock();
  if (block == 0) return 0;

  // process block here

  return block;
}


NewImageProducer::~NewImageProducer() {
  // clean up here
}
```

The classes in the *ImageProducer* and *AudioProducer* branch of the class tree form the modules of the restoration pipeline. The following sections are dedicated to a detailed description of each module's functionality, including the algorithms that were used and how each module can be configured to match a specific need. The modules are presented in the sequence most likely to be used in a pipeline.

Only snippets of code are reproduced in the following documentation. However, the entire source code can be found on the accompanying CD-ROM.

## 5.2 The Image Data Container — ImageBlock

```
ImageBlock(int width, int height);

unsigned char* getData();
int  getWidth();
void setWidth(int width);
int  getHeight();
void setHeight(int height);
void copy(ImageBlock* block);
```

A call to the constructor, passing the desired *width* and *height* of the image as arguments, allocates a dynamic array of *width* · *height unsigned char*s (later holding the image data) and it sets the *width* and *height* properties accordingly.

A pointer to the first byte in the array can be obtained by calling *getData*. The other *get* and *set* methods allow to learn about the current dimensions of the image or to alter them.

The method *copy* performs a deep copy of an entire image block.

## 5.3 Reading PGM Files — PGMReader

```
PGMReader(StatusView* sv, const char* filename, int blockLines);

ImageBlock* produceBlock();
int getImageWidth();
int getImageHeight();
class FileOpenExcpt {};
class FileFormatExcpt {};
```

This is where the image data comes from. Basically, an instance of *PGMReader* creates an *ImageBlock* and fills it with *blockLines* lines of image data from the PGM file denoted by *filename* on every call to *produceBlock*. The block width is read from the file. If not enough data is available to fill the block entirely, the height property of the *ImageBlock* is adjusted accordingly.

The *ImageBlock* instance is created only once. Subsequent calls to *produceBlock* return always a pointer to the same *ImageBlock*, but filled with new data.

Currently, *PGMReader* supports only PGM type P5 files, i.e. raw binary gray-scale images. It adheres to the following header specification, taken from the `man` page for `pgm`.

1. A “magic number” for identifying the file type. A raw pgm file’s magic number is the two characters “P5”.
2. Whitespace (blanks, TABs, CRs, LFs).
3. A width, formatted as ASCII characters in decimal.
4. Whitespace.

5. A height, again in ASCII decimal.
6. Whitespace.
7. The maximum gray value, again in ASCII decimal.
8. Whitespace.
9. Width · height gray values, in plain bytes, between 0 and the specified maximum value. A value of 0 means black, and the maximum value means white.
10. No whitespace is allowed in the grays section, and only a single character of whitespace (typically a newline) is allowed after the maxval.
11. Characters from a "#" to the next end-of-line are ignored (comments).
12. No line should be longer than 70 characters.

If a different header is encountered, a *FileFormatExcpt* exception is thrown. If the file does not exist or cannot be read, a *FileOpenExcpt* is thrown.

Due to the modular structure of the restoration pipeline, this module could easily be extended or replaced in order to cover other PNM flavors, read TIFF or any other graphics file format, without affecting the rest of the framework. The only prerequisite such an module has got to fulfill, is to make the image data available as an *ImageBlock* with the data itself being a linear array of byte values, each byte representing a gray-scale value between 0 and 255. The data must be ordered row-wise, not column-wise.

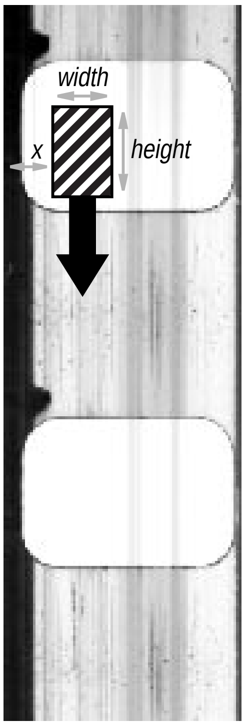


**FIGURE 5-4.** *The running sum of the pixel intensities underlying a sliding window is used to detect a transparent sprocket hole.*

## 5.4 Allow Synchronization — PerforationCounter

It is not possible to tell from the digital image of the optical sound track alone, how many lines of the image correspond to one second of sound. Using the standardized dimensions of the sprocket holes, it would be possible to compute this value. But considering a total sound track length of several minutes to more than one or two hours, that value cannot be accurate enough to keep the sound track synchronized with the picture.

Four sprocket holes per frame or 96 holes per second has been a standard for 35mm stock since 1917 [22]. This property can be used to compute the total running time and the proper sampling rate if we know the total number of sprocket holes. Keeping track of the perforation hole count is what the *PerforationCounter* class does, as each block passes through it.

### 5.4.1 Counting the Sprocket Holes

This algorithm makes a couple of assumptions. It assumes for instance that the image intensity between the sprocket holes is lower than inside the holes. The area inside the holes does not have to be white, but it should be uniform in the direction of film travel. Transparent in this context means "a high intensity of the pixels".

The basic idea of the algorithm is a window sliding vertically over the image and computing the average intensity of the pixels it is covering (see Figure 5-4). This value can be compared to a threshold, and if it is greater, the perforation count can be increased by one and we go into the "transparent" state. Figure 5-5 shows the state diagram used.

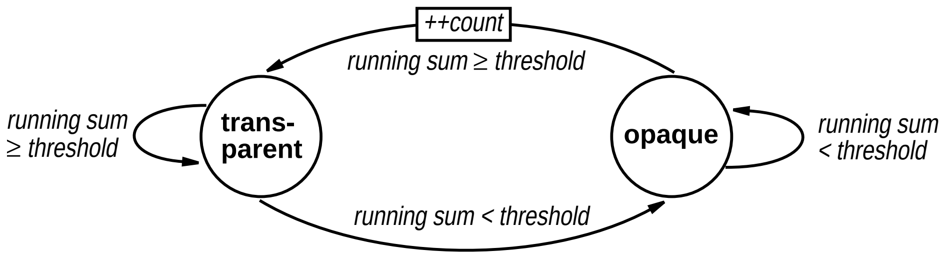


**FIGURE 5-5.** *The state machine used for counting the sprocket holes*

Computing the sum of the pixel intensities under the window is efficiently achieved by using a “running sum” of line intensities. Every time the window moves one line down, the pixel intensities of the new line are added together. This sum can be added to the existing window intensity while subtracting the intensity of the line dropping out of the window. A general template *RunningSum* was implemented for keeping track of the line intensities. If we choose the threshold suitably, we can use the sum directly instead of the average for the comparison.

Because the algorithm does not suppose the sprocket holes are completely white, we need to determine a suitable threshold. I have somewhat arbitrarily decided to do this the following way: In the first block a *PerforationCounter* processes, the minimum and maximum running sums are computed. One assumption was, that the area inside a sprocket hole is more or less homogenous. So the running sum inside every hole will be rather close to the maximum. I chose to set the threshold equal to the maximum minus one tenth of the difference between minimum and maximum. This setup proved to be reliable in practice.

If a sprocket hole is not entirely visible at the top of an image or at the bottom of an image, it is possible to miss it, depending upon how much of the hole is visible and depending upon the window height. It can happen only at the beginning and at the end of the image file, however, not at block borders — they are handled correctly due to the running sum which acts as a buffer. A loss of two sprocket holes equals two 96th of a second. This delay is acceptable, given that the position of the sound track relative to the picture is standardized ([49], ANSI/SMPTE 40-1997) with an accuracy of ±2 sprocket holes, too.

### 5.4.2 Using the PerforationCounter class

```
PerforationCounter(ImageProducer* producer, StatusView* sv,
                   int x_offset, int width, int height);
ImageBlock* produceBlock();
int         getCount();
```

The constructor takes five arguments. Besides the source (an *ImageProducer*) where the *PerforationCounter* gets its data from and the instance of *StatusView* used for the output of status messages, the location and dimensions of the sliding window have to be specified (see Figure 5-4 on page 42).

Use the *getCount* method to get the current count of sprocket holes.

## 5.5 Where is the Sound Track? — PositionEstimator and its Subclasses

```
PositionEstimator(StatusView* sv) : ImageProducer(sv) {}

virtual int* getXPositions(int i = 0) = 0;
virtual int* getYPositions(int i = 0) = 0;
virtual int  getWidth(int i = 0) = 0;
virtual int  getCount(int i = 0) = 0;
```

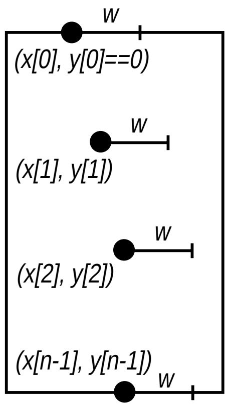


**FIGURE 5-6.** *The x- and y-coordinate arrays returned by a call to getXPositions and getYPositions describe these points in the current ImageBlock. The size n can be obtained by a call to getCount. Method getWidth returns a positive or negative width w.*

We would like to let the restoration pipeline be as independent from the specific position and dimension of the sound track as possible. For instance, a module removing dust from the image should not have to worry first about where the track borders are.

For this reason, the class *ImageCutter* (see 5.9 on page 50) was introduced. It extracts the sound track area from an image which usually would cover the entire film width [11]. In order to tell the *ImageCutter* in an independent way where the track is located, the *PositionEstimator* class was created.

It is an abstract class, providing only a general interface for inquiring a set of [x,y] coordinates. Thus, a class using an instance derived from *PositionEstimator* has a defined way of obtaining the coordinates, while leaving the task of calculating them up to the subclass.

The position of the coordinates is as shown in Figure 5-6. The decision how many coordinate pairs are calculated is up to the subclass. Method *getCount* returns that value.

*PositonEstimators* are part of the processing pipeline and are to be placed somewhere in front of a module using it, in order to have the coordinates ready when the other module needs them. The coordinates always relate to the current ImageBlock.

The current implementation contains three classes derived from PositionEstimator. The following sections describe them in more detail.

## 5.6 Standard Offset — ConstPositionEstimator

```
ConstPositionEstimator(ImageProducer* producer, StatusView* sv,
                       int position, int width);
```

This class can be used to specify a constant offset in x-direction and a width. Only two coordinate pairs are created, one for the top of the *ImageBlock*, the other for the bottom. The x-coordinate of both points as well as the y-coordinate of the upper one (it always equals zero) remain constant, while the y-coordinate of the bottom point is adjusted to the respective *ImageBlock* height.

## 5.7 Compensating Film Weave — VerticalEdgePE

A slow change of the sound track location over time is commonly referred to as film weave. It usually stems from the film drive, as a perfect fit between sprocket holes and sprocket teeth seldom exists. In our case, the deviation came not only from the scanner's film drive, but was already caused by the contact printing process used to save the nitrate original on safety stock. See Figure 5-7 for an example.

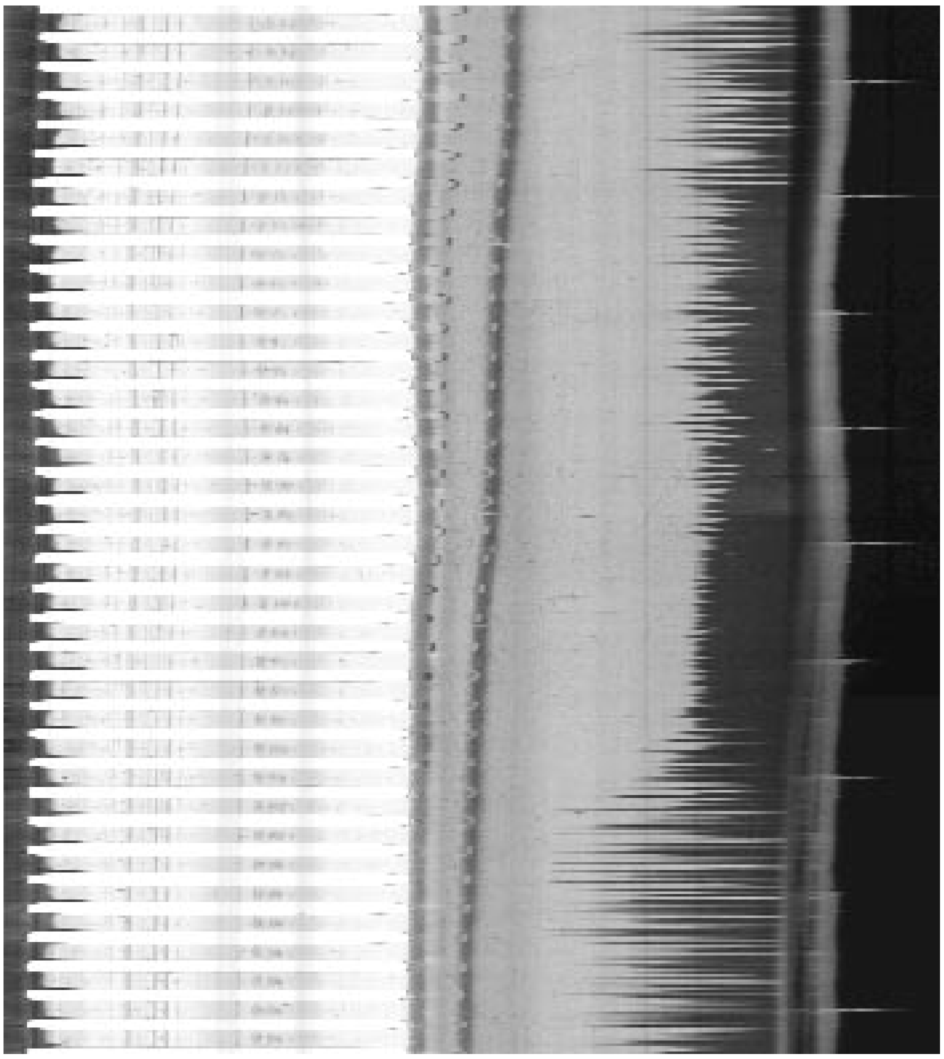


**FIGURE 5-7.** *Film weave on the scanned image data. The clipping covers an area of 740 by 20000 pixels (corresponding to less than half a second of sound) and was squeezed along the vertical axis to make the deviation visible. The light area on the left are the sprocket holes – parts of the picture frames are visible on the right. Note the difference between actual film weave introduced by the scanner (visible along the perforation) and the additional film weave introduced while contact-copying the original to safety stock.*

As long as the entire sound track is within the area covered by the photo cell or its digital equivalent, the wrong track location does not cause any distortion by itself. Only the total brightness within a slit's face (along a pixel row in our case) is decisive, not the position of the transparent area itself.

However, because the sound track now covers a wider area over time than intended, other transparent areas such as the one between perforation and sound track in Figure 5-7 can move into the focus of the photo cell, which will lead to a strong increase in noise.

Another possibility is the clipping of transparent sound track parts, as they move out of the area seen by the photo cell over time. Consider for instance the unilateral track shown above. If the right track border moves out of the photo cell's field of vision, the volume will decrease.

More serious is the clipping of the signal peaks during times of strong modulation. It will lead to severe harmonic distortions due to the rectified wave form.

Still another advantage of correcting the deviation is that other modules residing later in the restoration pipeline can depend on a straight track conforming rather exactly to the model of an ideal sound track. In the case of the unilateral sound track depicted above, they could safely assume that after the track has changed from opaque to transparent, it doesn't change back to opaque again. This property will be important later when we do dust removal on the image (see "Dust Removal II — MonotonyEnforcer" on page 58).

Film weave of up to 8% of the sound track width has been observed in the scans made from the unilateral mono sound track of the "Ciné-Journal".

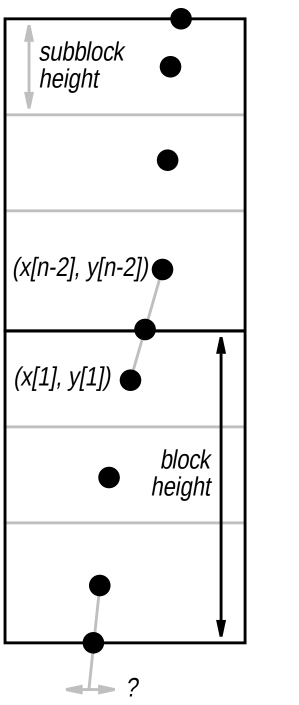


**FIGURE 5-8.** *Problem concerning block edges. In order to compute the last x-coordinate, the first one of the successive block has to be known. Coordinate pairs are computed for the vertical center of a subblock.*

### 5.7.1 Edge Detection

Film weave can only be observed at vertical edges or lines in the image, so taking them as reference points is an obvious choice. Luckily, there are often edges in the close surroundings of the sound track, undergoing the same positional changes. Class *VerticalEdgePE* produces a set of coordinate pairs in dependency of a vertical edge. The algorithm works as follows (as always, the source code is located on the accompanying CD-ROM)

1. The algorithm should be robust against dust or other image defects, so the edge position is not computed for every line, but only for subblocks of a given size. If the block height is not divisible by the subblock height, the last subblock will always be bigger than the others, up to double the subblock height. The height of the subblocks can be chosen by the user of the *VerticalEdgePE* class.
2. For a subblock, the column average is computed in a window around the supposed edge location (the user provides a hint), so we have a reliable guess for the intensities in the environment of the actual edge location.
3. We compute the gradients in x-direction along these intensities. A positive or negative weight $w$ is used for detection of either transitions from dark to light or vice versa.

$$\nabla f = w \cdot (x_{n+1} - x_{n-1}) \qquad \textbf{(EQ 6)}$$

4. The value of $n$ where (EQ 6) reaches its maximum is taken as the position of the edge in the processed subblock.
5. The computed edge location is now used as a hint for calculating the edge position in the next subblock.

The position computed by a *PositionEstimator* should be easy to use by another class, and there should be no jumps at block edges. Therefore, each block contains (redundant) coordinate pairs at the start as well as at the end of a block. Computing the x-coordinate of the last point turns out to be difficult, because we have to know the edge position of the first subblock in the following block (see Figure 5-8).

As a consequence, *VerticalEdgePE* buffers one block and thus introduces a delay of one step into the pipeline.

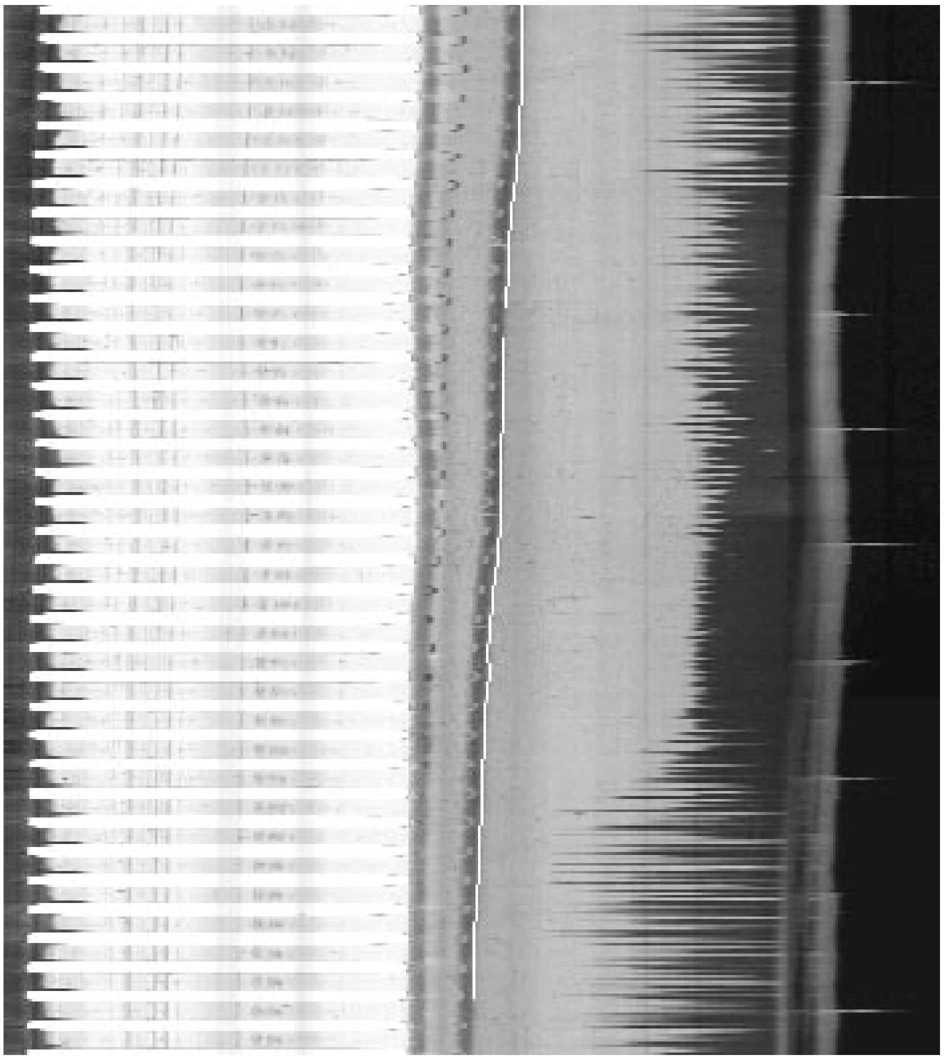


**FIGURE 5-9.** *Edge detection performed by the VerticalEdgePE class. It delivers the edge position as a set of coordinate pairs, shown here as a connected white line. An image block is divided into subblocks first, allowing for a constant granularity of the edge positions, independent of the ImageBlock size.*

### 5.7.2 Using the VerticalEdgePE class

```
static const int PREFER_NEGATIVE_EDGE = -1; // white-->black
static const int PREFER_POSITIVE_EDGE =  1; // black-->white

VerticalEdgePE(ImageProducer* producer, StatusView* sv,
               int hint, int width, int offs, int edgePreference,
               int subblockSize = 300, int window = 40);
```

I have omitted the public methods inherited from PositionEstimator.

In addition to the two standard arguments, an *ImageProducer* acting as the source for the image data and a *StatusView* for communication to the outside world, the *VerticalEdgePE* constructor expects at least five more arguments. A sixth is optional.

1. You have to give an initial guess, where the edge you would like to follow starts. Use the argument *hint* for the edge's x-coordinate on the first line. This value is usually provided by the user on the command line.
2. A *width* is included in the PositionEstimator in order to store data relating to dimensions in one place. The *width* is used to normalize the track position, too. This means the coordinate arrays delivered by the standard *PositionEstimator* methods *getXPositions* and *getYPositions* denote always a left border. So you can have the right border

of a sound track detected, without having to take any precautions in the following modules.

3. The offset *offs* serves a similar purpose. Maybe some edge in the picture would be very convenient for compensating film weave, but is not one of the sound track borders at the same time. An offset specifies where you want the positions to be relative to the detected edge. An offset can be positive or negative, too (negative means “to the left of the detected edge”, positive means “to the right of the detected edge”). A combination of *width* and *offs* allows arbitrary positions relative to an edge.
4. An edge can be a transition from white to black (as the one detected in Figure 5-9) or from black to white. Use the argument *edgePreference* with a value of one of the two predefined constants *PREFER_NEGATIVE_EDGE* for transitions from white to black, and *PREFER_POSITIVE_EDGE* for transitions from black to white.
5. As its name already indicates, *subblockSize* allows to configure how many subblocks (and thus positions) are calculated for each block. I have typically used a value of 300 lines.
6. The last — and optional — argument allows to change the width of window used to limit the area searched for the maximum gradient. The default value is 24, meaning 12 pixels to the left and 11 to the right of the hint are taken into account.

## 5.8 Determining the Symmetry Axis — SymmetryAxisPE

Very similar in intention and in implementation to the *VerticalEdgePE* is this class used for determining the position of the symmetry axis of bilateral variable area sound tracks. The symmetry axis, too, is affected by film weave (see Figure 5-10). In addition it is important to know its position for the restoration task presented in “Dust Removal III — SymmetryEnforcer” on page 59.

Unlike the edge detection module, this class expects only the sound track area to be contained in the *ImageBlock* it receives, i.e. the original image has to be passed through an *ImageCutter* first (see 5.9 on page 50).

### 5.8.1 How the Symmetry Axis is Computed

A first thought could be to compute the point on a line where half the line intensity is reached. This approach has a couple of disadvantages, however. First, it is very dependent on the location of the track in our picture. This holds especially for negatives, where the high intensity values of the white pixels surrounding the opaque area have a strong influence on our result. Even in positives, the opaque (black) pixels surrounding the transparent (white) track never have an intensity of zero, and hence still influence the position of the symmetry axis. Second, this method is susceptible to dust and other image defects, too. So a more reliable solution has to be found.

Eventually, I have decided to use the following method, being more independent of dust and positional changes:

1. Similar to the *VerticalEdgePE* class, the column averages for a subblock are computed across the track width.
2. Next, the minimum and maximum intensities in these column averages are determined, along with their positions.

3. Now follows the main step. See Figure 5-11 for an accompanying drawing. In the case of a positive sound track, we can assume that the position of the maximum intensity observed lies between the track edges somewhere near the symmetry axis. The same is valid for the minimum position in the case of a negative. The basic idea is now to go from there in both directions towards the track borders and record the distance it took to get there. A pixel lying on the track border will have an intensity between the minimum and the maximum, so we can choose such a value and count the pixels we pass on our way to the right or to the left until this value is reached. As it is not guaranteed that such a pixel with the computed value actually exists, we can replace the condition “equals” by “is less than or equal” in the positive case or “is greater or equal” in the negative case.
   In order to add robustness against intensity variations, the distance to the track border is not only computed once, but for every intensity value between the minimum plus a specific offset and the maximum minus a specific offset. This offset is added or subtracted because we want to be sure only intensities belonging to the track edges are taken into account.
4. These computations are made for every subblock in each track and they are provided in arrays like the ones used in *VerticalEdgePE*. If there are multiple tracks, the corresponding array can be requested by a call to *getXPositons(track)* and *getYPositions(track)* respectively. Track numbers start at zero.

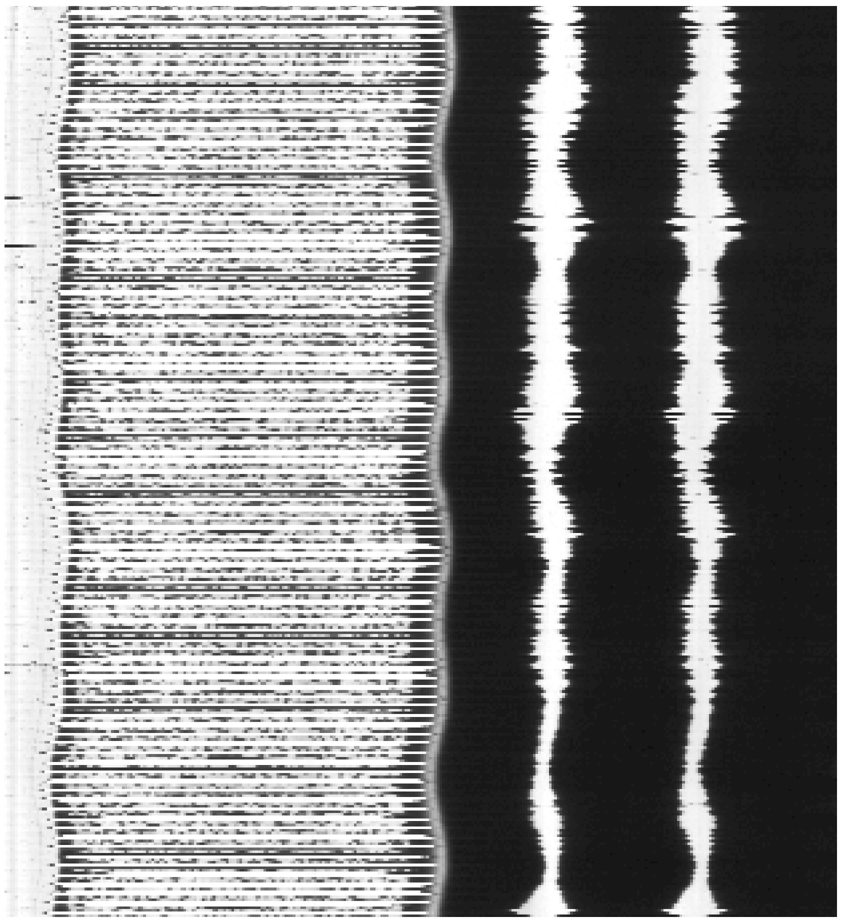


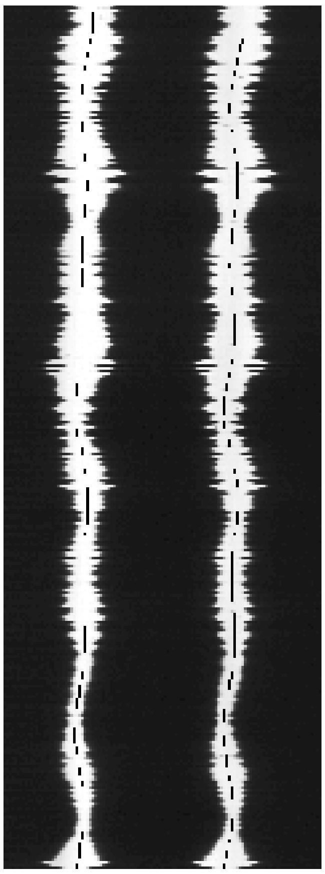
**FIGURE 5-10.** *Film weave visible on a modern symmetric variable-area sound track. Again, the cropping was squeezed in the direction of film travel. The original height was 48000 pixels, corresponding to about one second of running time.*

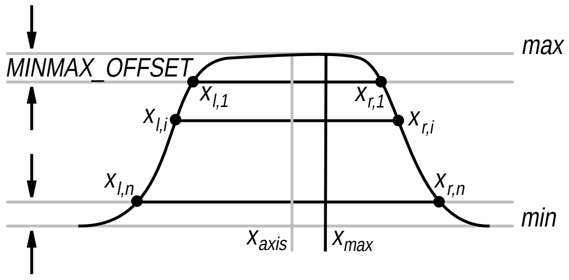
### 5.8.2 Using SymmetryAxisPE

```
SymmetryAxisPE(ImageProducer* producer, StatusView* sv,
               int tracks, bool positive, int subblockSize);
int getTracks();
```

Besides the standard arguments for every *ImageProducer*, *SymmetryAxisPE* needs three more arguments:

1. The number of tracks in the picture. It is assumed that if the image width is divided by the number of tracks, each track lies within the respective image area. E.g. for a stereo optical sound track, the image is divided in half, and one channel is assumed to lie in the left half, the other one in the right half.
2. Specify if you are processing a positive or a negative. The boolean value true means positive, false means negative. A transparent track on an opaque background is considered positive.
3. The subblock size is used as in *VerticalEdgePE* (see 5.7 on page 45).





$$x_{axis} = \frac{\sum_{i=1}^{n} x_{r,i} + \sum_{i=1}^{n} x_{l,i}}{2n}$$

**FIGURE 5-11.** *Symmetry axis of the two variable-area tracks as estimated by SymmetryAxisPE.*

## 5.9 Extract the Sound Track — ImageCutter

This class was introduced to gain more independence from the input images. Such an image will typically cover the entire film width, but could as well only show the optical sound track and the perforation area. The restoration modules should not have to care about these differences and perform the calculations for determining the area to be processed over and over again.

The *ImageCutter* class uses a class derived from *PositionEstimator* to know about the track location and copies just the sound track area into a new (smaller) *ImageBlock*. All successive modules can safely assume that the image data they process only contains the optical sound track.

In the case of a constant offset and width, this task would be simple. However, we have seen the presence of film weave in the last two sections and thus we have to take slanted edges into account. Furthermore, the one-pixel resolution is not sufficient as it leads to sudden jumps in the track position, which can, under circumstances, lead to jumps in the audio stream as well. It is not desirable for other reasons, too. A later restoration task might depend on the continuity of the track edge, which in this case would be sacrificed.

For these reasons, I have chosen to resort to subpixel precision.

Using floating point arithmetic for this task would make life easier, but it comes at the cost of a performance penalty. As this is a basic module, probably used by every application, I decided to spend more time on finding a solution based only on integer arithmetic.

## 5.9.1 Calculate the New Intensities Using Integer Arithmetic

Moving a row by a subpixel increment can be imagined as laying a new grid with exactly the same dimensions as the line, just offset by a small increment *dx/dy*, over the image row and adding the pixel contributions for each cell (see Figure 5-12). This operation just adds the contribution of the leftmost pixel and subtracts the contribution of the rightmost pixel from the line intensity (or vice versa, if the row is being moved to the right), which is the proper behavior, considering our application. More formally expressed:

$$I_{newrow} = (I_{p_0} + I_{q_1}) + (I_{p_1} + I_{q_2}) + \ldots + (I_{p_n} + I_{q_{n+1}}) \tag{EQ 7}$$

Grouping things differently, (EQ 7) can be written as

$$I_{newrow} = I_{p_0} + \sum_{i=1}^{n} (I_{p_i} + I_{q_i}) + I_{q_{n+1}} \tag{EQ 8}$$

Considering that original line intensity was

$$I_{oldrow} = \sum_{i=0}^{n} (I_{p_i} + I_{q_i}) = \sum_{i=0}^{n} I_i \tag{EQ 9}$$

we can finally write (EQ 8) as follows, and we see that the operation has the desired properties.

$$I_{newrow} = I_{oldrow} - I_{q_0} + I_{q_{n+1}} \tag{EQ 10}$$

The new grid divides each pixel in two parts, the ratio of the two parts being the same for every pixel in the row. This ratio is determined by the gradient of the straight line connecting the two points we got from the *PositionEstimator.*

Using only integer arithmetic, we can express the contribution of a pixel as follows (see Figure 5-12 again for the geometric meaning of the variables)

$$I_{p_i} = \left(\frac{I_i \cdot y \cdot \Delta x}{\Delta y}\right) - \left(I_i \cdot y \cdot \frac{\Delta x}{\Delta y}\right) \qquad \textbf{(EQ 11)}$$

In the usual mathematical sense, the right side of this equation is equivalent to zero. Using integer arithmetic and evaluating the terms in the correct sequence, however, we calculate the desired part of the pixel intensity.

Considering performance, it is unfavorable to evaluate (EQ 11) for each pixel. If we set

$$a = y \cdot \Delta x - y \cdot \left(\frac{\Delta x}{\Delta y}\right) \cdot \Delta y \qquad \textbf{(EQ 12)}$$

we can express (EQ 11) as follows:

$$I_{p_i} = \frac{a \cdot I_i}{\Delta y} \qquad \textbf{(EQ 13)}$$

The value of *a* remains constant for an entire row of pixels. Having calculated one part of a pixel’s intensity, we get the other part for free:

$$I_{q_i} = I_i - I_{p_i} \qquad \textbf{(EQ 14)}$$

The code using this calculation scheme, and pointer incrementation for efficient navigation in the image data is printed in Listing 5-2.

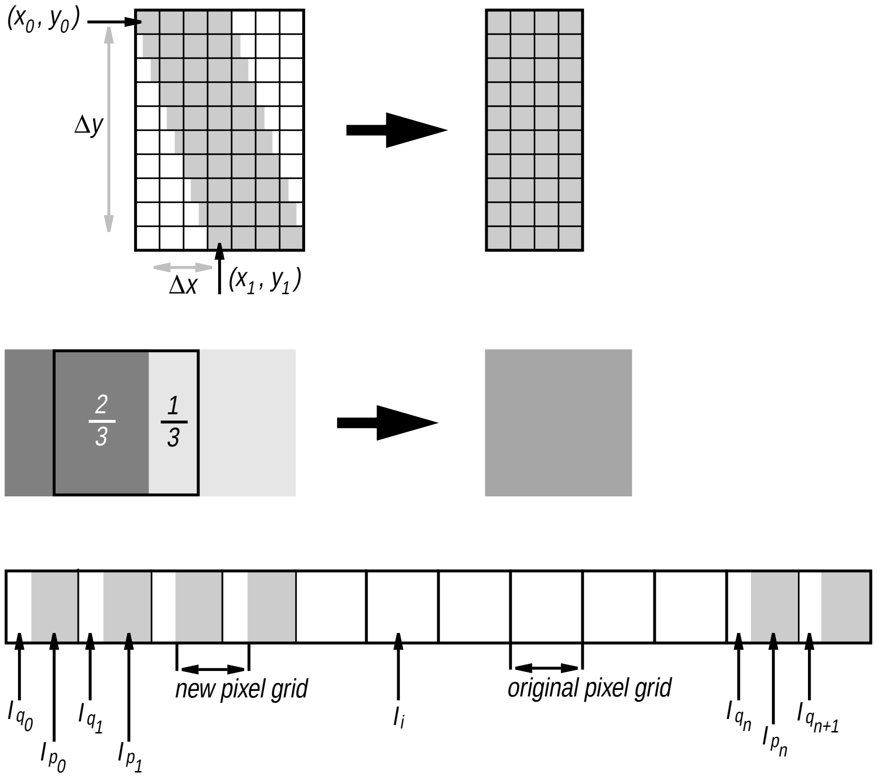


**FIGURE 5-12.** *The task of cutting out a row of specific width, allowing for subpixel increments, is displayed in the upper part of this figure, along with the variable names used in the above equations and in the source code. The middle part shows an example of the simple scheme how a new pixel intensity is being calculated. The bottom part shows a (longer) row and the contributions of two pixels at a time to the intensity of a new pixel.*

### 5.9.2 Using the ImageCutter Class

```
ImageCutter(ImageProducer* producer, StatusView* sv,
            PositionEstimator* estimator,
           int padL = 0, int padR = 0, unsigned char padVal = 0);
```

Using the *ImageCutter* class is straightforward. Basically you just have to pass a pointer to a *PositionEstimator* subclass in addition to the standard arguments.

You can choose to add some columns of a constant intensity on both sides of the resulting image block. This could come handy, if you would choose to restore clipped peaks on variable area sound tracks, where you would need to enlarge the sound track in order to accommodate the newly calculated areas.

```
int x0 = x_pos[i];
int x1 = x_pos[i+1];
int y0 = y_pos[i];
int y1 = y_pos[i+1];

int dx = x1 - x0;
int dy = y1 - y0;

for (int y = y0; y < y1; y++) {
  int Iq, Ip;
  int pos = x0 + (y-y0) * dx / dy;

  unsigned char* in  = image_in + pos + y*in_w;
  unsigned char* out = image_out + m_pad_l + y*out_w;

  int a  = dy*(x0-pos) + dx*(y-y0);

  if (dx < 0) {
    Iq = -a * *(in-1) / dy;
    a += dy;
  }
  else {
    Iq = (dy-a) * *in / dy;
    ++in;
  }

  for (int i = 0; i < w; i++) {
    Ip = a * *in / dy;
    *out = Iq+Ip;
    Iq = *in - Ip;
    ++in;
    ++out;
  }
}
```

**LISTING 5-2.** *The code cutting out an image subblock if the x-coordinates of the starting point and the end point differ. The simple case where they are the same is treated specially and hence more efficiently. The reference points x0, x1, y0, y1 are initialized with the values of a PositionEstimator like VerticalEdgePE.*

## *5.10 Removing Film Grain — ContrastEnhancer*

```
ContrastEnhancer(ImageProducer* producer, StatusView* sv,
                 int low = -1, int high = -1);
```

Besides dust, film grain is a major source of noise, especially on older stock. Using suitable thresholds on the digital image can reduce this type of noise. Care must be taken in choosing the thresholds so as not to increase quantization noise due to the reduced number of gray-scale values available. See 4.6.1 on page 33 for more details about the connection between quantization noise and image depth.

Image intensities below the *low* threshold are set to zero, intensities above the *high* threshold are set to white. The remaining values are linearly scaled between black (0) and white (255).

The constructor creates a lookup table (LUT) for the new intensities. It is later used by *produceBlock* to convert each pixel to its new value.

If the *low* and *high* thresholds are not specified, an attempt is made at guessing sensible values for them. The method is heuristic and based upon the histogram of the first *ImageBlock* processed.

Usually you'll determine good values using `xv` or a similar tool, however, as a good choice of the thresholds is critical to the performance of the following *DustRemovers*.

## *5.11 Basic Dust Removal — WhiteDustRemover and BlackDustRemover*

Dust and scratches are the main source of noise on optical sound tracks. They come in two flavors. What could be called black dust is dust physically present on the film, covering parts of the transparent track area. White dust on the other side was already present on the negative and was then printed on the positive, resulting in transparent spots on the print. Depending on depth and the resulting diffraction of light, scratches can belong to both categories. Examples of both categories can be seen in Figure 5-13.

A connected-component labeling algorithm as outlined in [8], p. 44ff, was applied to remove isolated areas of opacity or transparency. A detailed explanation of the algorithm and the adaptation to the problem of dust removal follows in the next section.

The two classes of *WhiteDustRemover* and *BlackDustRemover* differ only slightly. Nonetheless, I have decided to implement them as separate classes, so as not to clutter the code with too many distinctions of cases.

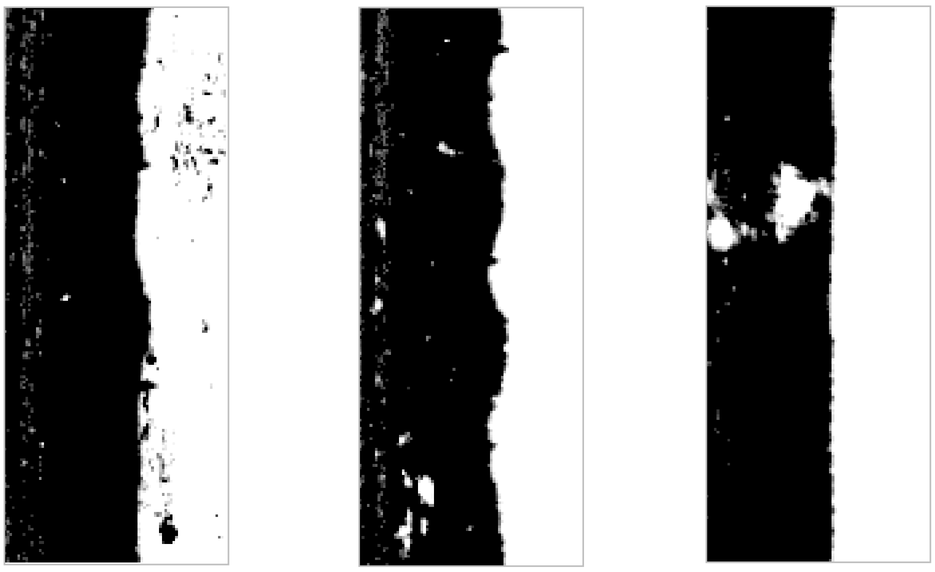


**FIGURE 5-13.** *Examples of white and black dust, taken from the "Ciné-Journal" unilateral variable area sound track.The track has already been processed by the ContrastEnhancer.*

### 5.11.1 The Connected-Component Labeling Algorithm

The basic algorithm presented in [8] was originally developed by Rosenfeld and Pfaltz [48].

It usually operates on a binary image, classifying adjacent groups of pixels of the same value as "connected components". Applied to our situation, and considering only white dust for the moment, we can regard the transparent (i.e. white) area as one connected component on an opaque background. Each additional white component can be regarded as dust if it is independent from our main component.

Connectedness is a global topological property. It is possible to detect global adjacency properties by local means, however. How this is done is presented in the next paragraph. See the above-mentioned sources for more background information.

The first run of two doesn't affect the image itself yet. Governed by the image information, it first builds a component map.

The operator examines every image row from left to right. As it advances through the image, it inspects two of its neighbors in each step. Consider Figure 5-14.

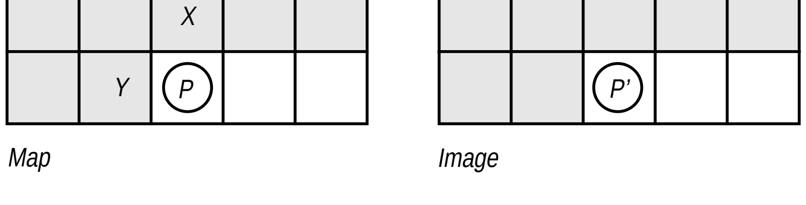


**FIGURE 5-14.** *The cases a CC operator can encounter.*

The shaded pixels have already been processed. If Pixel P' does not have the background color, its top and left neighbors X and Y in the component map are inspected. One of four cases can be encountered:

1. X and Y have the background label (0). P gets the next available label number.

2. If X belongs to the background and Y has a label greater than 0, i.e. it belongs to a component, P gets the same label as Y. If Y belongs to the background and X to another component, P gets the same label as X.
3. If X = Y, and both don’t belong to the background, P is set to the label of X.
4. The final and most interesting possibility remains: X and Y don’t belong to the background component, but are not equal. In this case, P is the connection between two regions that were originally taken for separate components. P is set to the smaller label of the two, and the equivalence of the components represented by X and Y must be saved.

The actual dust removal is performed in the second run. Basically, we want to keep the background and the signal component and throw away the rest. For each pixel, we can ask the equivalence table for the corresponding label. Assuming for the moment that the background label is zero and the label of the modulated track is one, we can keep the pixel intensity if its corresponding label is less than or equal to one. Otherwise, we set the pixel intensity to black.

The same ideas apply to black dust removal.

Using the connected component labeling algorithm for our task of dust removal requires some additional thoughts. We must, for instance, be able to tell the signal component from the others. Furthermore, we can’t safely assume that the signal component is always connected. Film splices, dust and overmodulation can interrupt the track. If no special precautions are taken, this would result in the loss of large parts of the signal component.

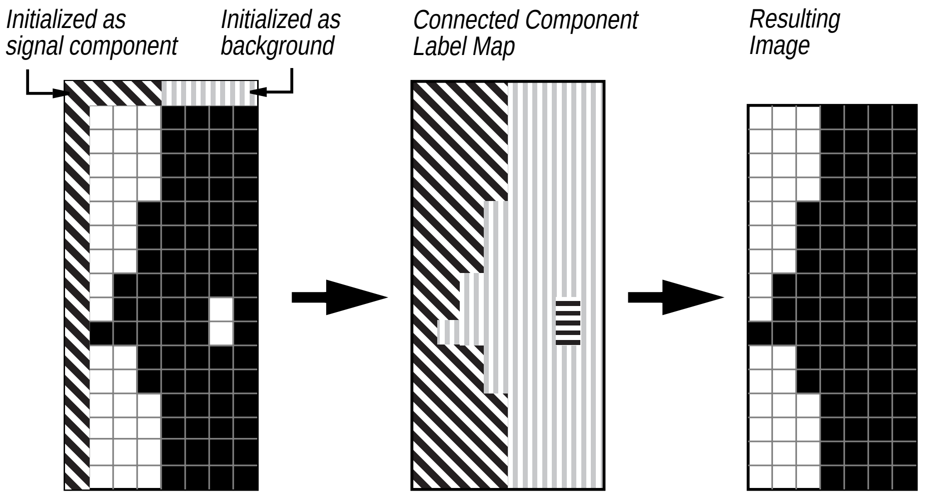

Initialized as signal component
Initialized as background
Connected Component Label Map
Resulting Image

**FIGURE 5-15.** *Adoption of a connected component labeling algorithm to the task of dust removal. Precautions are necessary in order to avoid loss of signal component parts.*

Because we inspect the adjacent pixels on the left and on top, the label map has to be one pixel larger in width and height than the image itself. We can elegantly use this property to initialize the label map.

The initialization depends upon the type of sound track being processed. The example in Figure 5-15 shows a positive mono unilateral variable area sound track. In this case, the signal component is located in the left half of the image, and we can explicitly set the labels of the border surrounding the left half of the image to “signal component”. The rest of the border labels are set to “background”. Executing the algorithm outlined above on the shown image would result in the loss of the lower part of the white component if we had not initialized the left border to be part of the signal component.

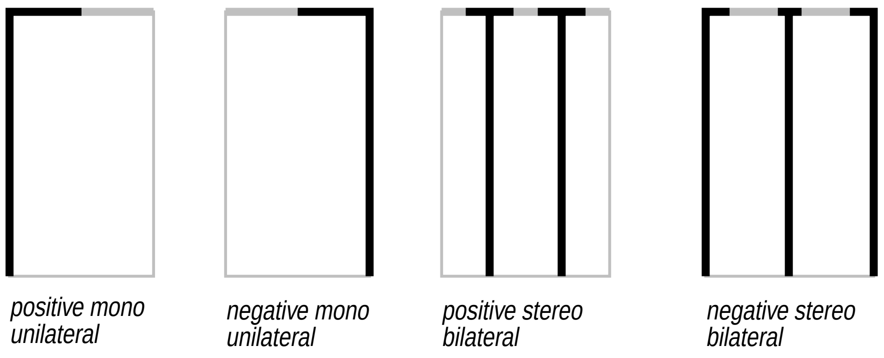


**FIGURE 5-16.** *Various initialization schemes for different variable area sound track formats (for WhiteDustRemover). Signal component labels are shown in black, background labels are shown in gray.*

The classes *WhiteDustRemover* and *BlackDustRemover* are held generally enough to cope with different combinations of channels and symmetry. Figure 5-16 shows various examples of initialization schemes for two of the most common sound track types.

If you look at Figure 5-17 and compare the images to Figure 5-13, you will notice that — due to its nature — the dust removal algorithm was only successful in removing dust particles not touching the border between transparency and opacity. The next section shows how the remaining dust can be removed under certain circumstances.

The presence of deep scratches on the sound track area could possibly lead to the loss of signal peaks when they are separated from the signal "body" by the scratch. This phenomenon has not been observed in the examples used.

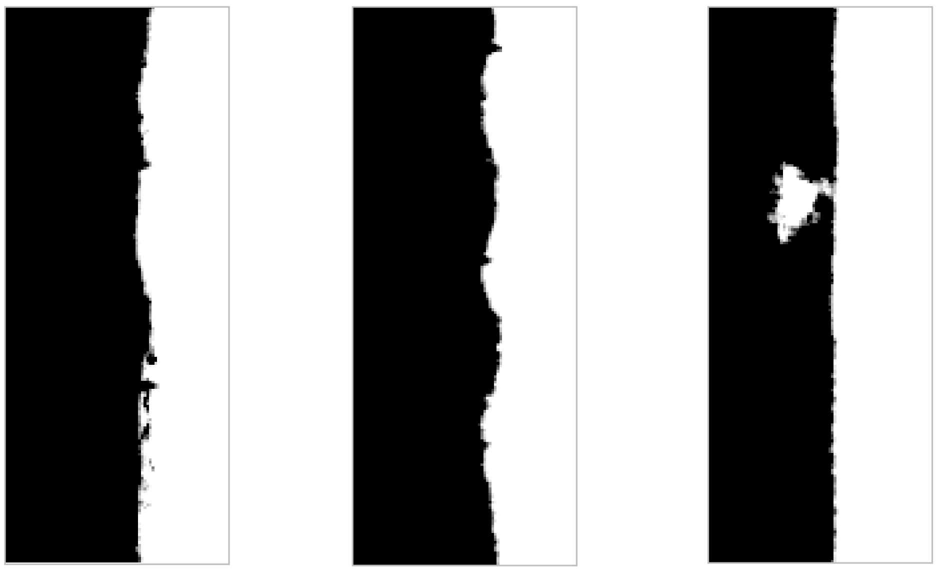


**FIGURE 5-17.** *Results of the "connected component labeling" dust removal algorithm.*

## 5.11.2 Using the WhiteDustRemover and BlackDustRemover Classes

```
WhiteDustRemover(ImageProducer* producer, StatusView* sv,
                 int tracks, bool symmetric, bool positive);
BlackDustRemover(ImageProducer* producer, StatusView* sv,
                 int tracks, bool symmetric, bool positive);
```

Both classes are configurable for an arbitrary number of *tracks*. Usually only one or two tracks are used. There has been a number of photographic sound track systems using more than two tracks, however. The additional tracks were either used as control tracks or they were actually carrying additional audio channels. Examples are the three signal, one control Fantasound system used for Walt Disney's Fantasia. The six-channel Maurer photographic sound track was only a mono signal distributed to six separate variable area tracks for purposes of more robustness against azimuth error distortion and lack of uniform lighting of the sound track (it was used for 16mm film).

The two remaining boolean parameters indicate if the track is *symmetric* and if it is a *positive* or a negative. As usual, positive in the context of a unilateral track means a white to black transition from left to right. The tracks in Figure 5-17 are negative for instance (they aren't real negatives, however; most likely, positives have been used when copying the nitrate films to safety stock).

## *5.12 Dust Removal II — MonotonyEnforcer*

If you cut a photographic sound track across its width and look at the intensities, you can notice that the intensities are monotonously increasing or decreasing between the track borders or between the track borders and the symmetry axis. Every deviation from this model indicates an error on the track such as a dust particle or a scratch.

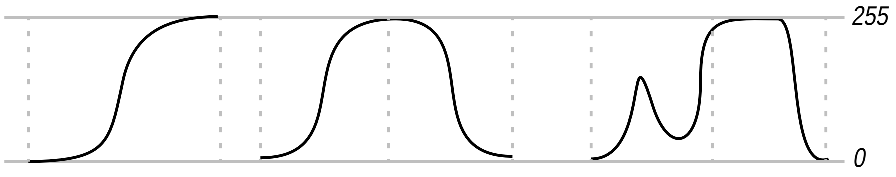


**FIGURE 5-18.** *The monotony property of photographic variable area sound tracks. The situation on the right indicates an error.*

In the most general case, it is not possible to decide locally if the monotony disturbance is caused by a white spot on a black background or vice versa. In certain cases we can assume that only real (black) dust exists. This is true for direct positives or for negatives.

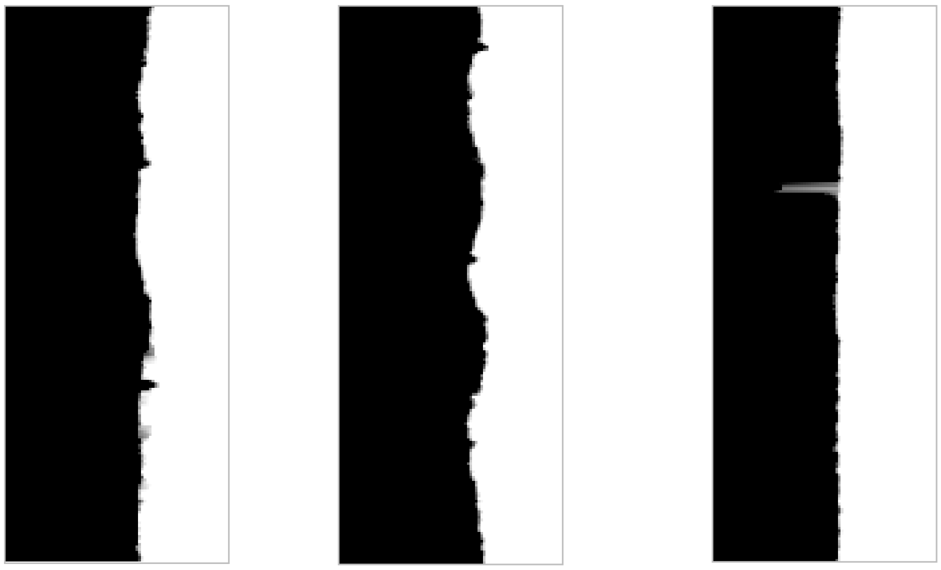


**FIGURE 5-19.** *Monotony enforcement applied to the previously shown dust examples.*

In this case, we can employ a simple monotony enforcement scheme: For every line, the pixels are traversed from the opaque to the transparent side. As long as the intensities increase, no action has to be taken. If we encounter a decrease in intensity before the other border or the symmetry axis is reached, we remember the last correct position along with its intensity and advance until a pixel with a higher intensity than the current maximum is encountered. Then we can interpolate the pixel intensities between the remembered position and the current one.

This scheme works only under the rather strict assumptions above. If scratches or white spots exist nonetheless, this simple algorithm can introduce major errors to the image, resulting in audible clicks in the audio output. Using the dust removal algorithms of the previous section “Basic Dust Removal — WhiteDustRemover and BlackDustRemover” on page 54 first and then applying this monotony enforcement scheme can remove remaining dust as shown in Figure 5-19. A better suited application is shown in Figure 5-20 on page 60 in conjunction with an another approach to dust removal for symmetric variable area photographic sound tracks.

Taking more than one line into account, especially the location of the edge between opaque and transparent areas on correct lines, could make this approach more versatile. Time was missing for further investigating this approach.

### 5.12.1 Using the MonotonyEnforcer Class

```
MonotonyEnforcer(ImageProducer* producer, StatusView* sv,
                 int tracks, bool positive, bool left2right,
                 SymmetryAxisPE* symm_pe = 0)
```

Again, this class can be configured for an arbitrary number of *tracks* as well as for negative or *positive* tracks.

If the intensities are monotonously increasing or decreasing from left to right is determined by the *positive* flag. However you can influence the direction used for enforcing the monotony. This is a crucial choice. The setting *left2right* should be chosen such that the track region not containing any dust is traversed first.

## *5.13 Dust Removal III — SymmetryEnforcer*

Due to the redundancy of the track shape, symmetric sound tracks are suited to an image based restoration attempt. In the ideal case, pixels of the same distance to the symmetry axis should have the same intensity.

For stereo and mono bilateral or dulateral tracks we can exploit the symmetry of one track. On older mono multiple bilateral variable area tracks, even more redundant track edge data is at our disposal. In this case, four rather than two pixel intensities have to be equal.

Again, the detection of an error on the track is easy, but correcting it requires additional knowledge or restricting assumptions about the dust and scratches on the sound track. For the example in Figure 5-20, the assumption that only black dust exists is valid, thus the symmetry property can be enforced by preferring the lighter pixel over the darker one.

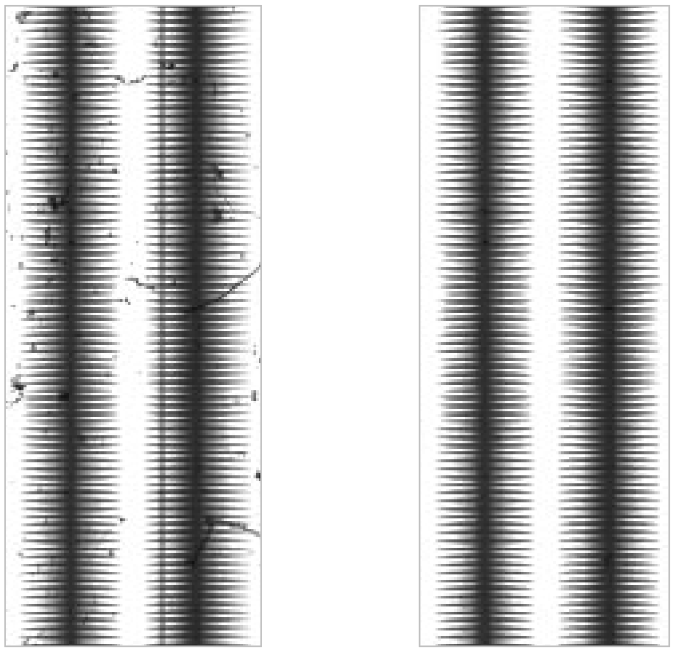


**FIGURE 5-20.** *A bilateral negative symmetric variable area sound track normally used for adjusting a cinema processor's sound head. Dust removal using the symmetry and monotony enforcing algorithms.*

The occurrence of dust at equal positions relative to the symmetry axis is not very likely due to the random nature of dust, even more so if two identical symmetric tracks can be taken into account. Along with the monotony property, very good results can be achieved as shown above.

If both black and white dust can occur, this method is not sufficient. More sophisticated decision techniques must be applied for distinguishing the different cases. Clearly, a more global view is necessary.

### 5.13.1 Using the SymmetryEnforcer Class

```
SymmetryEnforcer(ImageProducer* producer, StatusView* sv,
                 SymmetryAxisPE* symm_pe, bool positive,
                 int diff_allowed);
```

A good estimate of the symmetry axis position is crucial to the correctness of the symmetry enforcing scheme. The *SymmetryEnforcer* uses an instance of a *SymmetryAxisPE* position estimator for this task. See "Determining the Symmetry Axis — SymmetryAxisPE" on page 48 for more details.

You have to indicate whether you are processing a *positive* or a negative. The example in Figure 5-20 is considered a negative.

Some room for aberrations stemming from a slight discrepancy in the location of the symmetry axis, for instance, should be conceded. The argument *diff_allowed* specifies the tolerable difference between the two pixel intensities being compared.

## 5.14 Correcting Azimuth Deviation — ImageDeskewer

The restoration modules presented so far have been specific to variable area photographic sound tracks. The restoration task covered by this class is independent of the sound track type.

The term “azimuth deviation” describes the misalignment of the film recorder’s optical slit ([2], 18.43), which should be exactly at right angles to the direction of film travel. See Figure 5-21 for an example of a variable density sound track recorded with a misaligned optical slit.

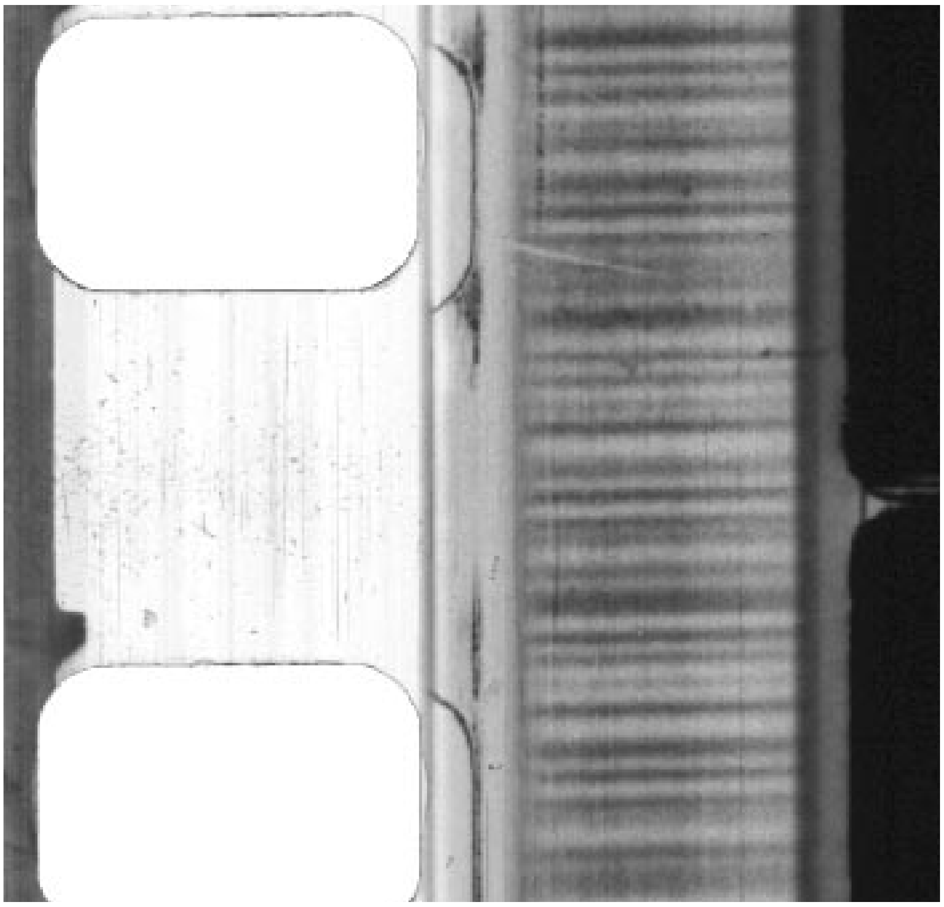


**FIGURE 5-21.** *Azimuth deviation of the recording slit visible on the “Ciné-Journal” variable density sound track part.The deviation is about 4 pixels across the track width of 196 pixels, corresponding to an angle of about 1.17 degrees.*

Even in the late 50’s and early 60’s a deviation of more than 0.00015 inch for a slit length of 0.096 inch was considered unacceptable ([2], 18.47). This corresponds to an angle of less than 0.1 degrees. Considering a typical scanned sound track width of 200 pixels, the deviation must therefore not exceed about one third of a pixel. In the image above, the deviation amounts to about four pixels in the direction of film travel.

Variable density tracks are less sensitive to azimuthal deviation than variable area tracks (see [1], p. 350ff for an in-depth mathematical analysis of the effect of azimuth deviation on variable area sound tracks). On variable density tracks, it results only in loss of high-frequency amplitude, while it introduces serious distortion on variable area tracks. The distortion effects vary among the different types of area tracks. Bilateral and unilateral tracks are less sensitive than unilateral tracks, especially when inspecting the second harmonic. The added robustness against azimuthal deviation was one of the main reasons for the introduction of multiple symmetric tracks.

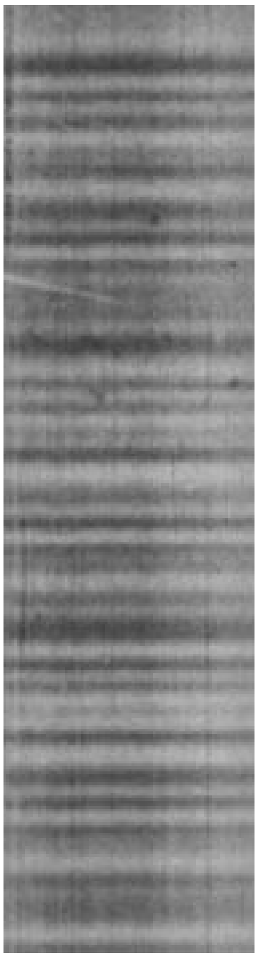


FIGURE 5-22. *The restored image processed by the algorithm outlined on the right.*

## 5.14.1 Deskewing the Image

As we have seen, the deviation angles are usually quite small, thus subpixel accuracy is a premise. The image is assumed to be skewed, not rotated. The following thoughts apply only to angles up to 45 degrees, which is sufficient considering the usually small deviations.

The algorithm is similar to the one described in “Extract the Sound Track — ImageCutter” on page 50, only this time, intensities are shifted in the direction of film travel, not horizontally. Therefore we have to cope with block borders. This problem is solved as a side-effect of the chosen implementation.

Figure 5-23 shows the assumptions made for the deskewing process. The following explanations are for positive angles. Analogous ideas apply for the negative case. The *ImageDeskewer* class handles both.

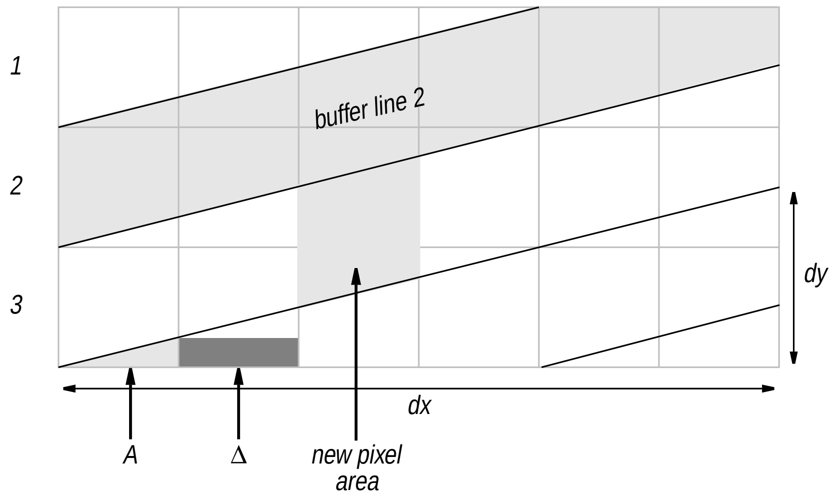
The azimuth deviation *dy/dx* determines how many lines are affected, let’s say *n*. In the Figure below, three lines are affected. This constrains us to process *n* lines until we can output a restored line. Therefore a circular buffer comprising *n* lines is allocated. This buffer solves the problem of the block limits, too.

Vice versa, we can say that each original image line contributes to *n* new lines. Another property is that every new pixel is composed of the intensities of exactly two original pixels.

Traversing each line i from left to right, the upper part of the first pixel is assumed to belong to line *i* and the lower part to line *i+1*.

On the first pixel in every row *i*, an area *A* is covered by the slanted line *i+1*. This area increases by Δ for the adjacent pixel on the right. Using *A*, we can calculate the intensity contribution of the current pixel to line *i+1*. The remaining intensity belongs to line *i*. We add both contributions to the corresponding positions in the buffer — the lower part in line *i+1*, the upper part in line *i*.

FIGURE 5-23. *The notations and assumptions of the deskewing algorithm.*

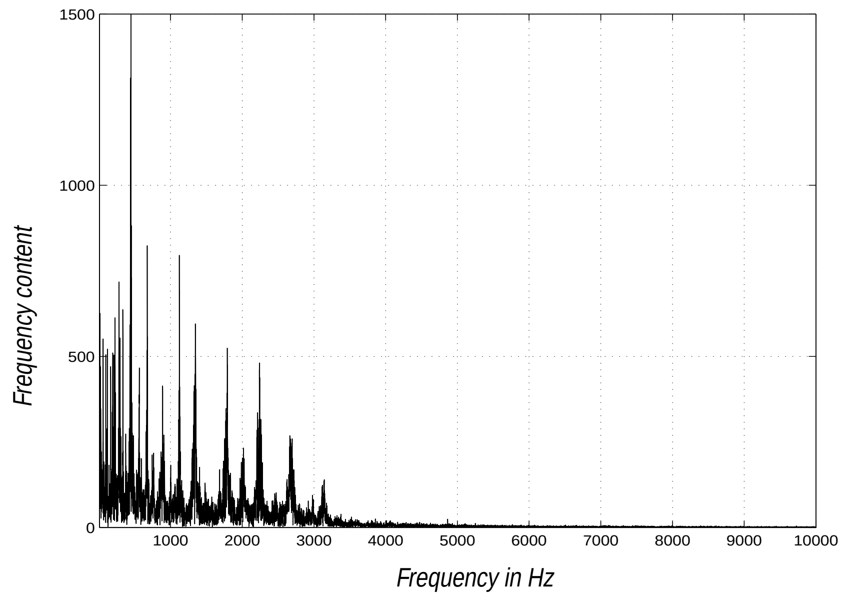


We do this for every pixel in the line, adding Δ to *A* in each step. If *A* exceeds 1, the value 1 is subtracted, and the line numbers to which the pixels contribute have to be adjusted.

After every image line processed, one buffer line is ready, i.e. the following rows won't contribute to its intensity, and we can write it to the image.

The values *A* and Δ have are simply related to *dx* and *dy*:

$$\Delta = 2A = \frac{dy}{dx} \qquad \textbf{(EQ 15)}$$



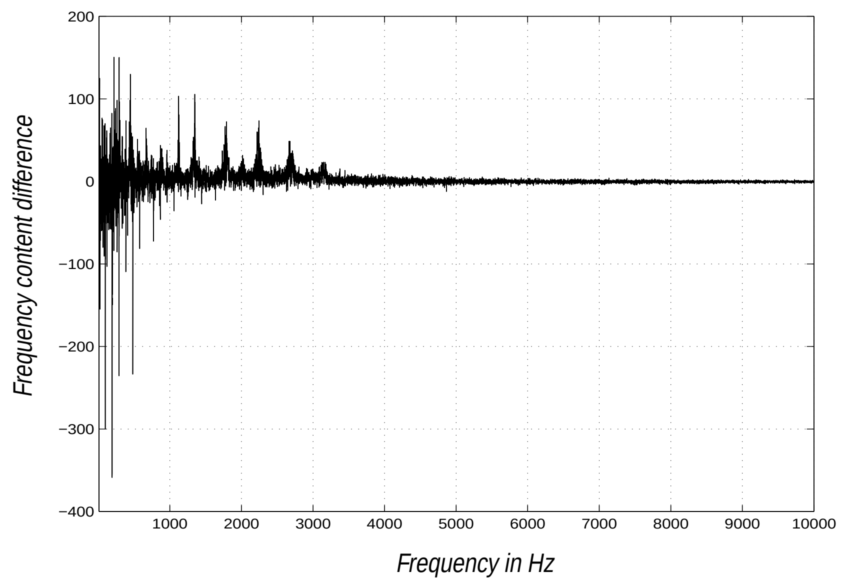


**FIGURE 5-24.** *The upper graph shows the spectrum of the audio data generated from the unrestored image. The lower graph displays the spectrum difference of the audio data produced from the restored and from the unrestored image.The gain in higher-frequency amplitude is clearly visible. The sound track is the variable density track taken from the "Ciné-Journal" containing intermezzo music.*

### 5.14.2 Another Approach

The algorithm described in the last section serves its purpose well, because only the entire line intensity is important for the conversion to audio. However, it is not entirely correct, as it is not being calculated what the original intensities were that led to the current value of a pixel.

That could be done in an ideal case. Consider the situation in Figure 5-25.

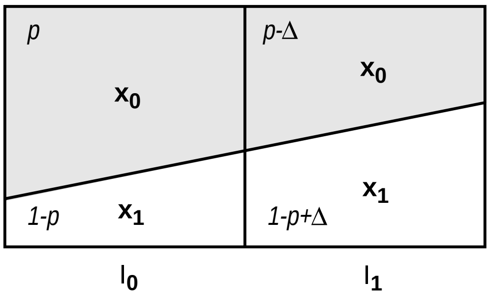


**FIGURE 5-25.** *Calculating the original values $x_0$ and $x_1$ leading to the pixel intensities $I_0$ and $I_1$*

The intensities $I_0$ and $I_1$ can be expressed with the following two equations.

$$I_0 = p \cdot x_0 + (1 - p) \cdot x_1 \qquad \textbf{(EQ 16)}$$

$$I_1 = (p - \Delta) \cdot x_0 + (1 - p + \Delta) \cdot x_1 \qquad \textbf{(EQ 17)}$$

Solving this equation for $x_1$ results in

$$x_1 = \frac{p \cdot (I_1 - I_0)}{\Delta} \qquad \textbf{(EQ 18)}$$

$$x_0 = I_0 - x_1 \qquad \textbf{(EQ 19)}$$

The presence of image noise and film grain makes it difficult to apply this model directly to the problem of azimuth deviation. Hence I have not further investigated this approach.

### 5.14.3 Using the ImageDeskewer Class

```
ImageDeskewer(ImageProducer* producer, StatusView* sv,
              double dy, double dx);
```

As this restoration task is independent of a specific sound track type, the only arguments beside the standard ones are *dy* and *dx*, expressing the azimuth deviation as depicted in Figure 5-23.

## 5.15 Writing Image Files Back to Disk — PGMWriter

```
PGMWriter(ImageProducer* producer, StatusView* sv,
          const char* filename,int max_gray_val = 255);
```

This class does not fulfill a restoration task and is not a necessary part of the pipeline, but it can be latched between two other modules in order to check the result of their algorithms.

The constructor expects a file name and optionally the maximum gray value assumed for the image. It is usually 255 for binary gray scale images.

The implementation is straight-forward. On every call to *produceBlock*, the *produce-Block* method of its predecessor *producer* in the pipeline is called, the data contained in the received *ImageBlock* written to disk, and the *ImageBlock* passed on to the calling instance, unaltered.

The only problem is the header, which doesn't have a fixed size in the PGM format (see "Reading PGM Files — PGMReader" on page 41). The image width can be taken from the *ImageBlock* class, but the number of lines is unknown until no more blocks are available.

Luckily, any program following the PGM file specification should be insensitive to white-space in the header. So we can reserve enough space at the beginning of the file and write the header after the last block has been processed. The maximum number of characters possible using a *long* integer type is assumed for header space reservation.

## 5.16 The Audio Data Container — AudioBlock

```
AudioBlock(int samples, int channels);

float* getData();
int getSamples();
void setSamples(int samples);
int getChannels();
void setChannels(int channels);
```

| *Left* | *Right* |
|---|---|
| **0** | **1** |
| **2** | **3** |
| **4** | **5** |
| **6** | **7** |
| **...** | **...** |

**FIGURE 5-26.** *Order of the samples in the float array for two channels.*

This is the equivalent of an *ImageBlock*, but for audio data. Instead of pixel intensities, audio samples are stored.

A call to the constructor creates a dynamic linear array of *samples · channels floats*. The order of the samples is given in Figure 5-26.

The dimensions can be inquired or changed using the *get* and *set* methods

## 5.17 Image to Audio Conversion — PhotoCell

Like its real counterpart, this class performs the conversion of the image representation to audio.

Photocells have a linear characteristic. Even the phototubes used in the early years of photographic sound tracks had a directly proportional relationship between the quantity of incident light and the resulting current ([1], p.77ff).

The CCD has a linear characteristic, too ([10], [11]). Therefore, the basic conversion becomes very simple: We just have to sum the pixel intensities across the sound track width and we get one sample value.

However, we have to take the slit height into consideration. The slit height. has to be chosen in accordance with the width of frequency spectrum desired to transmit. Slit heights range from 0.5 mils to 1.8 mils (one mil is one 1000$^{th}$ of an inch), corresponding to 0.013 to 0.046 mm or 1.4 to 4.7 pixels at a resolution of nearly 2700 dpi. It was standardized to 1.2 mils in 1946 ([1], p.530), but the dimension is missing in current standards [49]. A slit height of 0.5 mils is an often mentioned choice for today's tracks.

The *PhotoCell* class allows different slit heights, but only in increments of one pixel (about 0.37 mils at a resolution of 2700 dpi as achieved by our scanner).

For a closer look at the frequency loss introduced by the finite slit height, see "Slit Loss" on page 31 and [1], p. 539ff.

If a slit height of more than one pixel is chosen, the samples are computed using a running sum of the line intensities. The *RunningSum* template already mentioned in 5.4 on page 42 is reused for this purpose.

### 5.17.1 Using the PhotoCell Class

```
PhotoCell(ImageProducer* producer, StatusView* sv,
          int slit_height, int channels);
```

The *PhotoCell* class forms the interface between *ImageProducers* and *AudioProducers*. It is an *AudioProducer* itself, but expects an *ImageProducer* to be its predecessor. A further assumption made by the *PhotoCell* class is that an *ImageBlock* it receives contains only the sound track area, i.e it has been passed through an instance of *ImageCutter.*

The *slit height* has already been mentioned above. Another argument allows the configuration of the *PhotoCell* instance for an arbitrary number of *channels*. Usually, this is only one or two. See "Optical Sound Tracks" on page 9 for an overview of different multi-track formats used in the history of motion picture sound.

The sample values are stored in a a float array as it is described in "The Audio Data Container — AudioBlock" on page 65. This *AudioBlock* is returned to the caller of the *produceBlock* method.

## 5.18 Storing Audio Information in One Place — AudioDataCollector

```
AudioDataCollector(AudioProducer* producer, StatusView* sv);

float  getMinAmplitude();
float  getMaxAmplitude();
long   getSamples();
int    getSamplingrate();
int    getChannels();
int    getBytesPerSample();

void   setSamplingrate(int sr);
void   setBytesPerSample(int bps);
void   reset();
```

Most of the modules in the audio processing part of the pipeline need or alter information describing the audio data. As it is an advantage to keep information such as the sampling rate, the minimum and maximum amplitude encountered, the number of channels etc. in one place, this class was designed for that purpose.

An instance of this class is passed to the constructor of other classes using or modifying audio properties.

## 5.19 Saving Audio Data for Further Processing — RawAudioWriter and RawAudioReader

```
RawAudioWriter(AudioProducer* producer, StatusView* sv,
               const char* filename);
RawAudioReader(StatusView* sv, const char* filename,
               int samples, int channels);
```

These classes are only necessary because some audio processing modules such as the Academy filter or the WAVWriter need global information about the audio data that is only available after the entire sound track has been processed. This is primarily the sampling rate and the minimum and maximum amplitude of the sample values.

The current implementation of *RawAudioWriter* simply writes the raw *float* data to a temporary file. The dimensions are kept in memory — in the *AudioDataCollector* class. *RawAudioReader* does the inverse, it reads the raw data from disk into an *AudioBlock* which is suitably initialized with the values of the *AudioDataCollector.* On powerful machines, these modules could keep the data in memory, instead of writing it to disk. In this implementation, I opted for the more general case, even though a powerful machine was available. A typical amount of space used by the audio data in the raw format is about 200 KBytes per second or about 12 MBytes per minute of sound (48500 lines per second, mono, 32bit float).

## 5.20 Audio Filtering — FIRFilter

The image based restoration approach is complementary rather than alternative to audio processing methods. Some tasks cannot be performed on the image. There is for instance the standard electrical reproducing characteristic of the Research Council of the Academy of Motion Picture Arts and Sciences, widely known as the “Academy characteristic”. It is presented in more detail in the next section.

Mainly for filtering the audio data according to the Academy characteristic, the *FIRFilter* class was introduced. It is general enough to be used for other filtering tasks, though. The filter coefficients are computed in separate classes and provided in a common format (see the next section).

FIR (Finite Impulse Response) filters are also known as feedforward filters. Filtering in this case means convoluting the audio data with the filter coefficients of a specific length (see [3], p.61ff and [4], p.85ff for instance) — or simply put, “combining delayed versions of signals”, as it is shown in Figure 5-27.

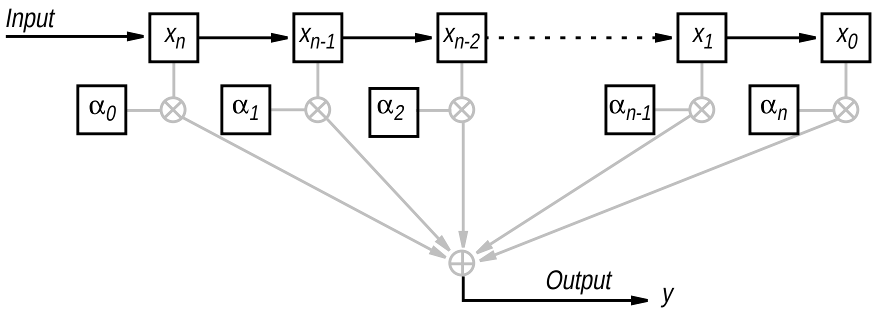


**FIGURE 5-27.** *The operation principle of a FIR filter.*

Consider one step in the computation of a filtered signal. The filter length is *n*. A new sample value is inserted at the first position $x_n$ of the sample chain. The preceding samples are shifted one position. In the actual implementation, a circular buffer is used for this task, so no actual shifting takes place. The values in the sample chain are then multiplied with their respective filter coefficient and summed up to produce an output sample.

When *x* denotes the sample chain which is basically a window laid over the samples, and *y* is the filtered value, the filtering operation can be mathematically expressed as follows.

$$y = \alpha_0 \cdot x_n + \alpha_1 \cdot x_{n-1} + \ldots + \alpha_{n-1} \cdot x_1 + \alpha_n \cdot x_0 = \sum_{i=0}^{n} \alpha_i \cdot x_{n-i} \qquad \text{(EQ 20)}$$

### 5.20.1 Using the FIRFilter Class

```
FIRFilter(AudioProducer* producer, StatusView* sv,
          FIRFilterCoefficients* fir_coeff);
```

Besides the standard arguments, the *FIRFilter* constructor expects only an instance of a class implementing the *FIRFilterCoefficients* interface. See the next section for a description of this abstract class. As always, a call to produceBlock will filter a block of audio data and return the filtered block to the caller.

## 5.21 Providing the Filter Coefficients — FIRFilterCoefficients

```
virtual double* getCoefficients() = 0;
virtual int     getNumberOfCoeffs() = 0;
```

In the previous section, the basic operation of a FIR filter was presented, but without mentioning where the filter coefficients come from. *FIRFilterCoefficients* is an abstract class providing an interface to the *FIRFilter* class for inquiring the filter length and the actual coefficients. A class derived from *FIRFilterCoefficients* can either calculate the coefficients itself or make a precomputed set of coefficients available. The coefficients have to be stored in an array of double precision floating point values.

### 5.21.1 An Example — The Academy Filter

In the first half of this century, the Research Council of the Academy of Motion Picture Arts and Sciences has made an effort of standardizing the electrical characteristic used for sound reproduction in motion picture theatres. It was a result of prolonged listening tests under the reproducing conditions common at that time, and the aim was mainly to reduce the audibility of noise.

It is basically a lowpass characteristic, linear until 2kHz, down about 7 dB at 6kHz and highly attenuated at 8kHz. This SMPTE standard was widely adopted by the industry. See [1], p.546f and [2], p.722 for more details.

However, there were quick improvements in the quality of the film medium, such as new types of fine-grain emulsions, which made higher frequencies possible without overly increasing the level of noise. Eventually, this led to the “pre-emphasis” technique, where higher frequencies were boosted in order to compensate for the attenuation caused by the Academy reproducing characteristic.

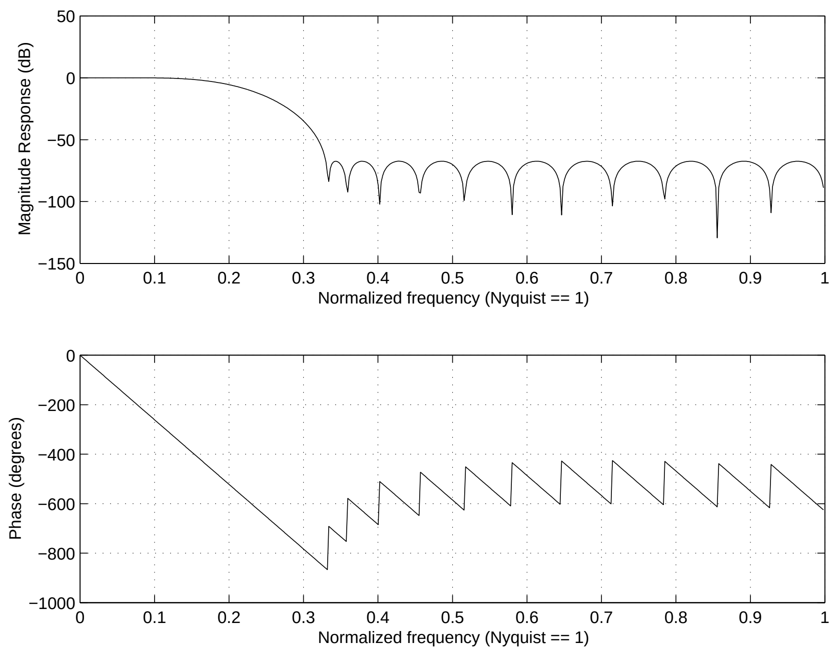


**FIGURE 5-28.** *The Academy filter frequency response as it is typically calculated by the AcademyFilterCoeffs class. The sampling rate was assumed to be 48500 Hz, so the range displayed in the graph begins at 0 Hz and ends at 24250 Hz. The filter length is 30.*

Modern playback equipment has a nearly linear characteristic, at least in the frequency range used by optical sound tracks (see page 10 for an overview of the typical frequency ranges of photographic sound tracks from the early days until today). In order to reproduce these old tracks as they were heard at the time they were recorded, and to compensate for the high-frequency distortion introduced by the pre-emphasis, the Academy characteristic has to be achieved by suitably filtering the audio data.

The *AcademyFilterCoeffs* class computes the filter coefficients for a low-pass filter with about the same characteristic as the Academy standard. No pre-computed set of coefficients can be used, as they are dependent upon the sampling rate.

The calculation of FIR filter coefficients is an intricate task, therefore the GNU library by Jake Janovetz was used. It employs the Parks-McClellan algorithm and provides the same functionality as the `remez` function in Matlab.

See Figure 5-28 for the typical frequency response of the filter calculated by this class.

### 5.21.2 Using the AcademyFilterCoeffs class

```
AcademyFilterCoeffs(double samplingrate);
```

The class is only dependent upon the sampling rate. After creation it has to be passed to the constructor of the *FIRFilter* class.

## *5.22 Creating a Playable Audio File — WAVWriter*

The only task remaining is getting the raw audio data into a playable form. I chose to use the WAV format as it is widely used and players exist for every platform. Surprisingly, this format is not so well documented as expected. A short description can be found in [4], p. 111, a more detailed description is available in [50].

The WAV format is part of the more comprehensive RIFF format. It is composed of different chunks and allows for a variety of formats regarding the audio data, such as PCM (Pulse Code Modulation), mu-law, a-law and ADPCM. This class creates files in the standard PCM format, i.e. the sample values are stored directly, either as unsigned bytes for 8-bit resolution, or as signed 16-bit integers for higher resolutions up to 16 bits per sample.

The header and data of a WAV file must be little endian, i.e. the least significant byte must come first. Thus special care must be taken on big endian machines, such as the Sun SPARCs. The *WAVWriter* class is kept portable and should produce valid results both on little and big endian machines.

WAV files can’t be bigger than 2 GBytes, the corresponding header field is limited to 32 bits. But even a 2 hour stereo 16-bit sound track at a sampling rate of 44.1 kHz would take up only 606 MBytes, so this format could be used for extracting and restoring the entire sound track of a typical motion picture. The file size is determined on-the-fly and written to the header after the last audio block has been processed.

The sampling rate is adopted unaltered. From a structural point of view, it would be no problem to introduce a *SamplingrateConverter* class into the pipeline. Due to the intricacies of sampling rate conversion, I have renounced to an implementation. Many public-domain programs are available for this task, such as the AFsp tools of the Telecommunications & Signal Processing Lab of the McGill University [51].

### 5.22.1 Using the WAVWriter Class

```
WAVWriter(AudioProducer* producer, StatusView* sv,
          AudioDataCollector* collector, const char* filename,
          int type = WAVE_FORMAT_PCM);
```

1. The *WAVWriter* class uses an instance of *AudioDataCollector* for information about the audio data. Using the maximum and minimum amplitude, the sound track is scaled to 90% modulation.
2. A file name, including the extension ".wav".
3. Optionally a type describing the format of the wave data in the AudioBlocks. Currently only PCM is supported, other formats could possibly work but have not been tested.

# CHAPTER 6 Example and Results

## 6.1 Restoration Example

A reel of the first “Ciné-Journal Suisse”, dubbed by Cinégram in Geneva, was the target of restoration. It contains various scenes from the disarmament conference of the League of Nations in 1932. Different types of optical sound track were used for the various parts, one being a variable-density track. Mostly, a unilateral variable-area system was used. The title of this thesis shows a part of the film, along with a picture of the truck carrying the sound recording equipment (a Visatone system). At that time, Cinégram was the only company in Switzerland with an optical sound recording equipment, and as it was in constant use, Swiss films had to be dubbed in Germany or in the UK ([9], p.124).

The reel was a copy and showed a unique type of defect, that makes clear another advantage of the image based approach: The film stock used until about the late 40s was nitrate based. This material proved to be problematic for several reasons. It is unstable and decomposes over the years, thereby becoming more and more inflammable. At the final stage, decomposed to dust, it ignites at a temperature below 40 degrees Celsius and the fire cannot be stopped, because the material produces its own oxygen as it burns. A main task of film archives consists in copying the nitrate stock to newer safety stock (usually acetate- or polyester-based). When the nitrate original of the “Ciné-Journal” reel was copied to safety stock, no attention was paid to compensate for the shrinkage. Thus the sound track is not at the standardized position anymore. If the copy is reproduced on a standard projector, parts of the copied perforation move into the scanned area and parts of the sound track move out, resulting in a high level of noise and partly to unintelligible speech. Usually, the photo cell of a reproduction unit cannot be adjusted in a wide range, so this would pose a problem if the sound track had to be digitized for further audio processing by traditional means.

The following two sections recapitulate what restoration efforts were made in this project and what the problems and results were.

In a third section, some data for the performance of the restoration algorithms is given.

Audio examples of several restored and unrestored scenes from the “Ciné-Journal” are on the accompanying CD-ROM which can also be played as an Audio-CD.

## *6.2 Summary*

In the following, an overview is given of what was achieved in the course of this project and where the shortcomings are.

An overview of the optical sound track formats was created with the focus on the various standard formats most likely to be encountered on a film that needs restoration.

A catalogue of different error types occurring at the different stages of optical sound recording was compiled. This could serve as a base for further restoration efforts.

The implemented C++ framework comprises already several restoration and auxiliary modules.

4. Due to the counting of the sprocket holes, synchronization with the film is possible. See “Allow Synchronization — PerforationCounter” on page 42.
5. Film weave can be compensated. See “Compensating Film Weave — VerticalEdgePE” on page 45.
6. Film grain contributing to the overall noise can be removed. See “Removing Film Grain — ContrastEnhancer” on page 53.
7. Dust and scratches can be removed to a great extent, leading to less noise. See “Basic Dust Removal — WhiteDustRemover and BlackDustRemover” on page 54. The sections 5.12 on page 58 and 5.13 on page 59 treat this subject as well.
8. Azimuth deviation can be compensated, leading to an increase of 16% for the higher frequencies that were attenuated by this type of recording error. See “Correcting Azimuth Deviation — ImageDeskewer” on page 61.
9. The Academy characteristic of old projecting equipment can be applied to the audio data for reproduction on new playback equipment with a linear frequency response. See “An Example — The Academy Filter” on page 69.
10. The modules for reading and writing image and WAV files can be reused for other purposes.
11. The framework is easily expandable to accommodate new restoration modules, both on the image and on the audio processing side.

The restoration effort leads to an audibly reduction of noise and to more clarity in speech as compared to the unrestored tracks (both are included on the accompanying CD-ROM). However, other defects (such as clipped peaks) that have not been treated, appear clearer than before.

As a bottom line, I’m sure that the image based approach has emerged to fulfill its promises, and the restoration algorithms implemented so far deliver good results. If the resolution and especially the image depth could be increased, the audio quality could be improved, too. This is particularly true for the variable-density tracks that suffer from the 8-bit resolution of the scanner, leading to audible quantization noise.

A quantitative measure of the results is difficult or even impossible to obtain, considering that the aim is to restore the — unknown — original signal including its original defects.

What could not be done in this thesis, either, is a quantitative comparison between the results obtained by audio restoration methods and the methods presented here. This is partly due to the difficulty mentioned above of judging if the original quality has been reached, partly because not exactly the same type of errors have been treated. Professional and expensive audio restoration modules as they are available from CEDAR or Sonic Solutions offer very good quality but cannot be compared squarely to the current implementation done in this project.

## 6.3 Problems

A very general problem is the indirect relationship between image quality and audio quality. A better-looking picture does not necessarily correspond to better audio quality. Over-exposing the sound-track, for instance, leads to better pictures but supposedly introduces harmonic distortion. This effect would have to be studied further but is difficult to gather without some sort of reference.

The other problem are the huge amounts of data generated by digitizing the film. Altogether, about 20 GBytes of sound track samples were scanned. This got worse because the “Ciné-Journal” example contained different optical sound track types that had to be extracted from the raw scans in order to process them with the proper algorithms. This task is like finding a needle in a haystack, but this tedious work can be lessened by watching the film from time to time while it is scanned and record the scan time that has passed until a splice is reached. The approximate location of the splice in the image can be computed from that time.

A similar problem was the current 2 GByte limit for a single scan due to the 32-bit Linux file system. A coherent audio part is unlikely to be contained in only one file, so multiple scans have to overlap and must be joined at the right place to produce a continuous audio file. While finding a splice in a 2 GByte file is tedious, finding the end of one file in another by eye is nearly impossible. I have therefore written a little program that takes the last part of one file and looks for a highly correlating part in another file, printing out the line where it was found. These output values can then be checked manually. After cropping the second file at the right offset, the two can be joined together, now forming a continuous audio track.

To my knowledge, no similar work has previously been done on the topic of digital image based restoration of optical movie sound tracks. Thus no sources were available for an initial orientation. At the beginning of the project, I didn’t have any sources on optical sound tracks, and it took some time to get an overview. Old volumes of the Journal of the SMPTE (which are luckily available in the ETH library) proved to be an invaluable source.

Furthermore, it would have been beneficial to have more knowledge about signal processing in general and audio processing in particular, than what is usually taught in the computer science courses.

## 6.4 Performance

### 6.4.1 Scanning

A scanning speed of 1000 lines per second was possible. At the current resolution of nearly 2700 dpi, this amounts to a factor of about 49 versus real-time. Thus, one minute of sound takes 49 minutes to scan. Compared to the speed of commercial frame-by-frame film scanners, this is quite fast.

### 6.4.2 Restoration

Restoration was performed on a DEC Personal Workstation 500a featuring a 500 MHz Alpha 21164 processor, 670 MBytes of RAM, and a RAID consisting of six 9.1 GByte hard disks and one 4.5 GByte hard disk. The system is running Digital UNIX V4.0D.

The C++ compiler was Digital's `cxx`, used with the switches `-arch host -inline speed -O5.`

Execution speed was timed with `/usr/bin/time`, delivering the real, system and user time consumed. The following table lists average times for different restoration tasks. The time factor indicates how much processing time is used compared to the length of the resulting audio file. The system time is a measure for the time used by I/O.

**TABLE 1.** *Average times for three typical restoration tasks*

| | Time factor vs. real-time | Percent system time |
|---|---|---|
| *Conversion only, fixed position* | *4 [±0.5]* | *21% [ 2]* |
| *Threshold, dust removal, variable position* | *7 [ 0.5]* | *13% [ 0.5]* |
| *Azimuth correction, variable position* | *10 [ 0.5]* | *9% [ 0.5]* |

The left column lists the restoration task performed. For all measurements, sprocket hole counting was done, the slit height was 2 pixels and the block size equaled 1000 lines. If the system time is a measure for I/O time consumed, it can be seen that disk throughput is not the governing factor, even in the simple conversion process from image to audio. However, it has to be noticed, that the system time is only slightly below real-time, but the ratio can be increased by choosing a larger block size.

For azimuth correction, floating-point arithmetic was used, which clearly causes a performance penalty.

The scanned images comprised the perforation area in addition to the sound track area and some space between the sound track area and the picture frames to accommodate film weave. For this setup, and at the current resolution of 2700 dpi, one second of real-time audio corresponds to approximately 34 MBytes of image data that have to be processed. As a rule of thumb, a scan of 2 GBytes results in one minute of audio.

.

# CHAPTER 7 Outlook

## 7.1 Software

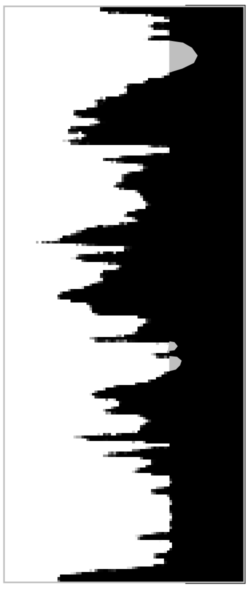


**FIGURE 7-1.** *Possible completion of clipped peaks by sine waves.*

Due to the numerous defects optical sound tracks can be subject to, there is ample space for the development of additional modules. The following two would be particularly interesting.

Clipping of peaks caused by overmodulation and the rather slow attack times of the noise reduction shutters seem to be a frequent problem of optical sound tracks. The clipped peaks could be completed by a sine wave in order to reduce the distortion introduced by the rectified wave form. See Figure 7-1 for an example.

Usually the best resources available are used for restoration. In the case of film and film sound tracks, that would be the original negatives. Currently, these can't be handled by this software, because image spread must be taken into account so as not to reproduce a distorted sound track. Achieving it digitally on the image would certainly be a big challenge. Possibly, the effect could be achieved by choosing an appropriate exposure time for the scanning process.

The software could also be completed with a graphical user interface, preferably written device-independently in Java. This graphical user interface could provide an easy way to navigate a large image and to determine the numerous parameters necessary for an optimal restoration.

The audio processing part of the pipeline should be complemented by a module which does high-quality sampling rate conversion on the fly. Currently, the software relies on one of the numerous tools available for this task.

## 7.2 Resolution

In order to cover variable-density tracks as well, and to produce higher-quality audio output, the resolution of the scanner, especially the image depth, has to be increased. A 6000 element (covering the entire film width), 16 bit CCD would, of course, be optimal.

## *7.3 Performance*

The increase in resolution would cause even more data that need still higher-performance machines. Need for more performance increases with every added restoration module, too. The different modules in the restoration pipeline act independently upon an image block and thus offer themselves to parallelization.

Currently, the time needed for the actual processing is quite small compared to the preparation time. Extraction of relevant parts, possibly joining overlapping scans and parameter estimation take quite a while. If the image based restoration should be practicable for a high volume of film data, all of the components would have to act with as little human interaction as possible.

APPENDIX A

# Bibliography

[1] J. G. Frayne, H. Wolfe: “Elements of Sound Recording”, John Wiley & Sons, New York, 1949, (ETH-Bib 410 305)

[2] H. M. Tremaine: “The Audio Cyclopedia”, Howard W. Sams & Co., Indianapolis, First Printing, 1959

[3] K. Steiglitz: “A Digital Signal Processing Primer”, Addison-Wesley, Menlo Park, 1996

[4] D. Stotz: “Computergestützte Audio- und Videotechnik”, Springer-Verlag, Berlin, 1995

[5] S. J. Godsill, P. J. W. Rayner: “Digital Audio Restoration”, Springer-Verlag, London, 1998

[6] R. C. Gonzalez, R. E. Woods: “Digital Image Processing”, Addison-Wesley, Reading, 1993

[7] M. Sonka, V. Hlavac, R. Boyle: “Image Processing, Analysis and Machine Vision”, International Thomson Computer Press, London, 1993

[8] O. Kübler, G. Gerig: “Bildverarbeitung und Computer Vision II”, Skript zur Vorlesung, ETH Zürich, IKT, Fachgruppe Bildwissenschaft, 1996

[9] H. Dumont: “Geschichte des Schweizer Films”, Schweizer Filmarchiv/ Cinémathèque Suisse, Lausanne, 1987

[10] Dr. L. Rosenthaler, PD Dr. R. Gschwind, A. Günzl, A. Wittmann: “Restoration of Old Movie Films by Digital Image Processing”, Image’Com 96, Arcachon-Bordeaux, Mai 1996

[11] A. Wittmann, Dr. W. Graff, Th. Gössi, Dr. L. Rosenthaler, Prof. Dr. A. Gunzinger, PD Dr. R. Gschwind: “New Scanning Approach for Motion-Picture Restoration and Archiving”, FkTG, August 1997

[12] Dr. W. Graff, A. Wittmann, Dr. L. Rosenthaler, Prof. Dr. A. Gunzinger, PD Dr. R. Gschwind: “New Scanning Approach for Motion Picture Digitizing”, Europto’98, Regensdorf-Zürich, Mai 1998

[13] A. Wittmann: “BIPP Basel Image Processing Package”, Department for Applied Image Sciences, Institute of Physical Chemistry, University of Basel, 1996

[14] E. W. Kellogg: “The ABC of Photographic Sound Recording”, SMPE Journal, 44, 3 (March 1945), p.151-194

[15] J. G. Frayne, A. C. Blaney, G. R. Groves, H. F. Olson: “A Short History of Motion-Picture Sound Recording in the United States”, SMPTE Journal, 85, 7 (July 1976), p. 515-528

[16] H, C. Wohlrab: “Highlights of the History of Sound Recording on Film in Europe”, SMPTE Journal, 85, 7 (July 1976), p. 531-533

[17] E. W. Kellogg: “History of Sound Motion Pictures”, First Installment, SMPTE Journal, 64, 6 (June 1955), p. 291-302

[18] E. W. Kellogg: “History of Sound Motion Pictures”, Second Installment, SMPTE Journal, 64, 7 (July 1955), p. 356-374

[19] E. W. Kellogg: “History of Sound Motion Pictures”, Final Installment, SMPTE Journal, 64, 8 (August 1955), p. 422-437

[20] E. I. Sponable: “Historical Development of Sound Films”, Part I & II, SMPE Journal, 48, 4 (April 1947), p. 275-303

[21] E. I. Sponable: “Historical Development of Sound Films”, Part III-VII, SMPE Journal, 48, 5 (May 1947), p. 407-422

[22] A. J. Miller, A.C. Robertson: “Motion-Picture Film — Its Size and Dimensional Characteristics”, SMPTE Journal, 74, 1 (January 1965), p. 3-11

[23] E. Schwandt: “VII. Der Tonfilm”, Die Rundfunk- und Tonfilmtechnik: ein Hand- und Lehrbuch für das Funkwesen, die Tonfilmtechnik undverwandte Gebiete / Hg: W Lehmann, Mitarb.: Wilhelm Lange

[24] C. G. Seagrave, M. J. Richards, D. E. Mandell, M. L. Atherton: “Apparatus Using One Optical Sensor to Recover Audio Information from Analog and Digital Soundtrack Carried on Motion Picture Film”, United States Patent 5’710’752, January 20, 1998

[25] P. Scheiber: “Stereophonic Sound System”, United States Patent 3’959’590, July 9, 1973

[26] M. E. Zhabotinski, A. A. Lapides, A. I. Shpuntov, “Coherent-Optic Suppression of Scratch Noise on Optical Sound Records”, Applied Optics, 22, 24 (December 15, 1983), p. 4020-4027

[27] A. A. Lapides: “Quasi-Coherent Spatial Filtration of Optical Sound Tracks”, Optics Letters, 10, 3 (March 1985), p. 101-103

[28] J. O. Baker, D. H. Robinson: “Modulated High-Frequency Recording as a Means of Determining Conditions for Optimal Processing”, SMPE Journal, ?, 1 (January 1938), p. 3-17

[29] J. K. Hilliard: “Push-Pull Recording”, SMPE Journal, ?, 2 (February 1938), p. 156-161

[30] C. C. Ceccarini: “Theoretical Notes on the Push-Pull Method of Recording Sound”, SMPE Journal, ?, 2 (February 1938), p. 162-168

[31] J. K. Hilliard: “Research Council Nomenclature for Release Print Sound-Tracks”, SMPE Journal, ?, 6 (June 1938), p. 656-665

[32] J. G. Frayne: “A Compatible Photographic Stereophonic Sound System”, SMPTE Journal, 64, 6 (June 1955), p. 303-307

[33] J. G. Frayne, R. R. Scoville: “Analysis and Measurement of Distortion in Variable-Density Recording”, SMPE Journal, ?, 6 (June 1939), p. 648-673

[34] R. E. Uhlig: “Two- and Three-Channel Stereophonic Photographic Soundtracks for Theaters and Television”, SMPTE Journal, 83, 9 (September 1974), p. 729-733

[35] R. E. Uhlig: “Stereophonic Photographic Soundtracks”, SMPTE Journal, 82, 4 (April 1973), p. 292-295

[36] I. Allen: “The Production of Wide-Range, Low-Distortion Optical Soundtracks Utilizing the Dolby Noise Reduction System”, SMPTE Journal, 84, 9 (September 1975), p. 720-729

[37] R. Dressler: “Dolby Pro Logic Surround Decoder Principles of Operation”, Dolby Laboratories, 1996

[38] J. D. Clifford, J. J. Charles: “The Relative Output of Optical Soundtracks”, 84, 9 (September 1975), p. 730-731

[39] E. Stetter: “Stereophonischer Lichtton”, Vortrag gehalten auf der FKTG Jahrestagung, Trier, Oktober 1978

[40] R. Dolby: “The Spectral Recording Process”, Journal of the Audio Engineering Society, 35, 3 (March 1987), p. 99-117

[41] M. F. Davis: “The AC-3 Multichannel Coder”, Dolby Technical Papers, Publication No. S93/9951

[42] Dolby Laboratories: “The Evolution of Dolby Film Sound”, Dolby Backgrounder, 1996

[43] C. Lerouge, R. Billeaud: “Steps, Techniques and Ethics of Sound Restoration”, Gamma Meeting, Bologna Seminar of July 1996

[44] J. Webers, J. Jondral: “Hat der Lichtton eine Chance?”, Funkschau, 1975, 6, p. 46-49

[45] “The Focal Encyclopedia of Film & Television Techniques”, Focal Press, London & New York

[46] G. Hartstone, T. Spath: “28. Film”, Sound Recording Practice, Oxford University Press, Oxford & New York, 1994, p.545-565

[47] K. Staes, W. Markie: “A Simplified Distortion Balance Test Procedure for Optical Sound Records and Prints”, SMPTE Journal, 90, 2 (February 1981), p. 97-102

[48] A. Rosenfeld and P. Pfalz, “Sequential Operations in Digital Picture Processing”. J. Assoc. Comput. Mach., 12, 1966, p. 471-494

[49] SMPTE Society for Motion Picture and Television Engineers: CD-ROM with SMPTE Standards for Motion Pictures

[50] “Multimedia Programming Interface and Data Specification v1.0”, Issued as a joint design by IBM Corporation and Microsoft Corporation, August 1991

[51] AFsp Tools, Telecommunications & Signal Processing Lab, McGill University.

APPENDIX B

# Software

## B.1 Installation

Copy the files from the `/sources` directory of the accompanying CD-ROM into the directory you wish to use. You must have the BIPP classes [13] installed. Edit the second line from top in `makefile`, `BIPPLOC`, to reflect the path where the BIPP headers can be found. Run the following command

```
make process
```

if you want to remove all object and backup files, type

```
make clean
```

The single classes forming the restoration pipeline do not depend on BIPP. They can be used as stand-alone objects.

## B.2 Usage

After compilation, you can invoke `process` with the following options. An explanation is also available on the command line by typing

```
process -help
```

Required arguments are `-in` and `-out`, one of `-fix`, `-wbe` or `-bwe` and one of `-perf` and `-lps`. The rest is optional and has to be chosen according to the restoration needs.

| | | |
|---|---|---|
| -in | <pgmfile> | PGM (type P5) file to be processed |
| -out | <wavfile> | Path of the output WAV file |
| -fix | <x> <w> | Fixed sound track location, <w> pixels wide, located <x> pixels from the left image border. |

| | | |
|---|---|---|
| -bwe<br>-wbe | <x> <w> <off> [<win>] [<bw>] | Use -wbe for white-black transitions and -bwe for black-white transitions. <x> is an initial guess for the edge location. The sound track does not have to start at <x>: you can specify an offset <off> which is added to <x>, forming the left or the right track limit. Use a negative <w> if <x>+<off> denotes the right track limit.<br>The optional <win> determines the window width searched for the maximum gradient. [default=24]<br>By specifying the subblock size (in lines) <bs>, you can optionally influence how quickly the sound track position may change [default = 300]. |
| -perf | <x> <w> <h> | Location of the perforation. <w> and <h> determine the window dimension used for the detection of a perforation hole. <h> should approximately equal 3/4 of a perforation hole's height. |
| -lps | <lines> | If you already know how many image lines correspond to one second running time, you can specify it instead of having the perforation holes counted. |
| -ch | <channels> | Number of channels/tracks [default = 1 (mono)]. |
| -negative | | Use this option for negatives. A positive is assumed by default. For symmetric variable area sound tracks positive means a transparent track on a black base. Positive unilateral sound tracks have a white to black transition from left to right. |
| -thr | <lo> <hi> | Thresholds for image noise suppression. You can override the automatic threshold estimate by specifying <lo> and <hi>. Pixel values below <lo> will be set to black, values above <hi> will be set to white. The automatic threshold estimator works only for variable area sound tracks. |
| -wdust | | Remove white dust |
| -bdust | | Remove black dust |
| -symm | [‘enforce’] | Use this option for symmetric variable area tracks A unilateral track is assumed by default The optional ‘enforce’ leads to better dust removal. |
| -monotony | ‘l2r’ \| ‘r2l’ | This option leads to better dust removal by enforcing the monotony property of variable area sound tracks. |
| -azi | <dy> <dx> | Corrects an azimuth deviation, specified as dy/dx (tangens of the angle). |
| -slit | <height> | Scanning slit height in pixels [default = 1]. |
| -bs | <size> | Block size in lines [default = 1000]. |
| -academy | | Use the Academy filter (8kHz low-pass). |
| -test | <pgmfile> | Write the processed image into <pgmfile>. |
| -skip | <lines> | Ignore the first <lines> lines of the image. |
| -quiet | | Suppress status messages |

APPENDIX C

# CD-ROM

The CD-ROM is dual-mode. The unrestored and restored audio parts can be played on a normal Audio-CD player. The software sources, and this report in PDF, FrameMaker and PostScript format can be read in a CD-ROM player.

The following tables contain an overview of the audio tracks, directories and files

## C.1 Audio Tracks

| | |
|---|---|
| Track 1 | Intermezzo music, unrestored, scanned at the standard sound track location. This was a mono unilateral mono variable-area track. |
| Track 2 | Same as 1, but the scanning location was slightly changed in order to better cover the sound track, but is still at a fixed offset |
| Track 3 | Same as 1, but with all the applicable restoration task applied. Corrected film weave, thresholds to remove film grain, both black and white dust removal. No Academy filter. A slit size of 2 pixels was used |
| Track 4 | Same as 1, but longer exposure time while scanning |
| Track 5 | Same as 3, but longer exposure time while scanning |
| Track 6 | Other intermezzo music part, unrestored, scanned at the standard sound track position. This was a mono unilateral mono variable-area track. |
| Track 7 | Same as 6, but the restoration methods described in 3 were applied |
| Track 8 | A third musical intermezzo, unrestored, scanned at the standard sound track position. This was a mono unilateral mono variable-area track. |
| Track 9 | Same as 8, but the restoration methods described in 3 were applied |
| Track 10 | The fourth intermezzo music part, unrestored, scanned at the standard sound track position. This was a mono variable-density track. |
| Track 11 | Same as 10, but with azimuth correction applied |

| | |
|---|---|
| Track 12 | Speech of Germany at the League of Nations' disarmament conference, unrestored, scanned at the standard sound track position. This was a mono unilateral mono variable-area track. |
| Track 13 | Same as 12, but the restoration methods described in 3 were applied |
| Track 14 | Speech of France at the League of Nations' disarmament conference, unrestored, scanned at the standard sound track position. This was a mono unilateral mono variable-area track. |
| Track 15 | Same as 14, but the restoration methods described in 3 were applied |
| Track 16 | Speech of China at the League of Nations' disarmament conference, unrestored, scanned at the standard sound track position. This was a mono unilateral mono variable-area track. |
| Track 17 | Same as 16, but the restoration methods described in 3 were applied |
| Track 18 | Raw scan 1, unrestored, standard position |
| Track 19 | Raw scan 2, unrestored, standard position |
| Track 20 | Raw scan 3, unrestored, standard position |
| Track 21 | Raw scan 4, unrestored, standard position |
| Track 22 | A part of the movie “Irren ist männlich”, reproduced by a CP-500 cinema processor and recorded to DAT. Without Dolby SR noise reduction. |
| Track 23 | A subpart of 22, digitized by the scanner, unrestored. This is a Dolby Stereo SR bilateral variable area sound track. No Dolby SR noise reduction. |

## *C.2 Directories*

| | |
|---|---|
| sources | The C++ sources of the restoration software. About 30 classes and 3500 lines of code. |
| report | This report in the formats PDF, FrameMaker and PostScript. |
| AFsp | The AFsp tools of the McGill University's Department of Signal Processing. Various tools to convert the sampling rate and play WAV files on Sun Sparc machines |
| docs | Additional documentation, like the Dolby Digital patent, the WAV file structure etc. |