SCANLINE SYNTHESIS AS A SONIFICATION METHOD FOR DIGITAL IMAGES: TECHNIQUES AND AESTHETICS – A CRITICAL REFLECTION

Marko Ciciliani
IEM – Institute for Electronic Music and Acoustics
University of Music and Performing Arts Graz, Austria
ciciliani@iem.at

Abstract
This paper discusses the method of translating pixel data from a digital image or film to a wavetable for sound synthesis. This method is not new, but has recently reached popularity in many audiovisual projects.[1] It is commonly referred to as *scanline synthesis.*[2]

It is an attractive method insofar as the sound is directly derived from the visual information and it therefore immediately renders audible changes in the visuals. In its direct form, scanline synthesis has in most applications quite a characteristic sound, although in principle any pitched timbre can be produced by scanline synthesis.

The basic principle is that pixel values from a digital image are copied into an audio buffer which is then used as the waveform for a wavetable oscillator.[3]

Keywords
Visual Music, Synthesis, Digital Image, Digital Film, Digital Signal Processing

Introduction

The basic principle of scanline synthesis is based on copying pixel values from a digital image into an audio buffer which is then used as the waveform for a wavetable oscillator. The pixels are usually read along a straight line in the image, but generally speaking an arbitrary grouping of pixels could be translated to a waveform.

The first part of the paper explains the basic principles of scanline synthesis. After laying out the essential aspects some methods are introduced that lead to a higher degree of control regarding the sonic spectrum of the sound. All technical issues of the paper are presented for readers who are not experts of electronic music. A very basic understanding of programming is helpful in order to understand the examples in pseudo code, but not absolutely necessary.

In the last part of the paper practical applications of scanline synthesis will be discussed. An important aspect is its usability in an artistic contest and the degree to which congruence between image and sound can be sensed. Some examples from existing works will be illustrated in

---

1 See for examples: Ryoho Kobayashi *Scanline Computer Music*, Christopher Jette *Soundlines*, Dave Poulter *Scanline Granular Synthesis* or the author's works *Formula minus One* and *Via*

2 not to be confused with *scanned synthesis* which is also a synthesis technique based on wavetables, however unrelated to audiovisuality.

3 All standard oscillators in digital synthesis are reading usually predefined wavetables. Wavetable oscillators give access to the wavetable so that the user can shape the waveform at will.

order to show different artistic solutions.

A quick overview of scanline synthesis

Wavetables

With analog sound generators, oscillations of electric currents are produced by specific circuit designs that generate periodic voltage changes. With digital oscillators these analog oscillations are emulated by so-called wavetables, that can be thought of as lists – usually referred to as arrays – of numbers that describe the shape of a single period of an oscillation with discrete values.

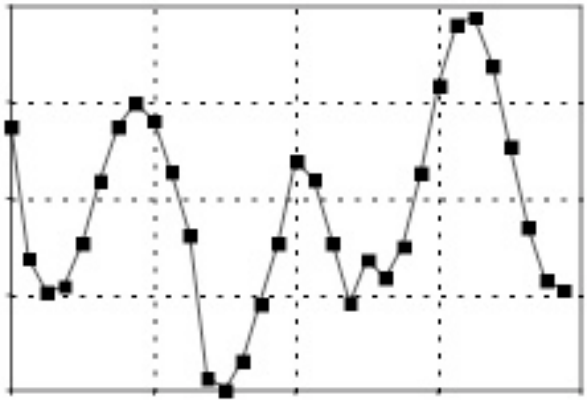

Figure 1: analog signal: continuous line; digital signal: black squares (discrete values)
Source: Smith, Steven W.:*Digital Signal Processing*, San Diego: California Technical Publishing, 1997

By repeatedly reading through these tables many times per second, and eventually converting the digital numbers to an analog signal, a similar oscillating current can be produced. Wavetables in early digital synthesizers contained 256 values or less. Today they are usually multiple times bigger than that.

In simple digital synthesizers these wavetables are usually pre-programmed according to standard waveforms and are not changeable. However, in computer software it is ease to fill wavetables with new values, even while a sound is being produced with the very same wavetable.

Scanline synthesis does just that by taking the numeric values from pixel values of digital images. Hence, an aspect of an image is rendered audible. There is a direct relationship between the pixel values on a screen and the wave-table values, and therefore between the visuals and the sound. Therefore this method has often been used in the context of visual music and audiovisual works in general.

Pixel Values

The values of pixels are stored as groups of bytes, which corresponds to a value range from

0-255. Depending on whether an image is colored or grayscale and whether or not the pixels have a value for opacity (alpha channel), a pixel contains between one and four bytes.

In a grayscale image the first channel is the darkness of the pixel, which is mapped between black and white on the range from 0 to 255. Color images have three bytes for the colors red, green and blue respectively. If an alpha channel is present it is added as a second channel with grayscale pixels or as a fourth channel with color pixels.

Converting pixel values for wave-tables

Digital audio signals consist of floating point numbers in the range between -1 and +1. When translating pixel values for wavetables, the larger numbers have to be mapped to the smaller range by dividing the pixel value by 127.5 and subtracting 1 from the result.

Pseudo code: audioValue = pixelValue / 127.5 – 1;

However, first it has to be decided, which channel of the pixels is used for the conversion, if several channels are available. With color pixels it is common to translate them to gray values – also called the luminance values. The NTSC standard for the conversion of color images to black & white states that the red, green and blue values have to be scaled with different weights.

When translating RGB values to grayscale, the following weights should be used:

red value * 0.299 + green value * 0.587 + blue value * 0.114.[4]

However, it should be noticed that the conversions from pixel values to wavetable values are perceptually largely arbitrary, hence the accuracy of e.g. converting color to grayscale may not be so significant. Issues concerning the perceptual correlation between the two media are going to be addressed further below.

Offset removal

Let us assume we are using a wavetable with 512 values and we are scanning a dark image. A large number of pixel values might be smaller than half of the 8bit range (a number lower than 128), which will result in values in the wavetable that are below zero. As such, this is not a problem but in digital signal processing it is considered not to be optimal if the average of all values in a wavetable is far from zero. Such deviations of the average value are called digital offsets.

In extreme cases such offsets can endanger the loudspeakers. Hence it is a good precaution to check all values that have been extracted from an image for their average value and to shift them in such a way that they average to zero.

If in our example the average (after converting from the 8 bit pixel range to the digital

[4] Wright, Jr., Richard S. et al: *OpenGL SuperBible*, fourth edition, Ann Arbor: Pearson, p.569

audio range) would be e.g. -0.3, every value in the table should be increased by this value, so the average ends up being zero.

Pseudo code:

```
waveTable = [0, 0.2, -0.9, -0.5];
average = 0;
for(i = 0; i < waveTable.size; i++){
        average = average + waveTable[i];
}

average = average/ waveTable.size;

for(i = 0; i < waveTable.size; i++){
        waveTable[i] = waveTable[i] + average;
}
```

For clarity the example codes use only short arrays with merely four values. In scanline synthesis a common size would contain 512 or 1024 values.

Palindromic reading of the wavetable

Let's assume that we are applying scanline synthesis horizontally to an image which is dark on the left side and gradually gets brighter towards the right. This would lead to numbers in the wavetable that are low in the beginning and higher towards the end. Since during synthesis a wavetable is read many times a second, there would always be a rather large jump from a high value to a low value, between the last value of the table to the first. This is not problematic, but every bigger sudden change of values in a waveform – in other words: every sharp edge in the signal – adds high partials to the sounds. In some cases this can lead to an undesired sharpness in the sound.

In order to prevent this, the size of the wavetable could be doubled and the numbers that were generated from the image could be placed in the wavetable twice, but the second time in a mirrored fashion.

Pseudo code:

```
pixelValues = [-0.5, -0.2, 0, 0.9];
waveTable = [0, 0, 0, 0, 0, 0, 0, 0];// twice the size of the array pixelValues
for(i = 0; i < pixelValues.size; i++){
        waveTable[i] = pixelValues[i];
        waveTable[waveTable.size - (i + 1)] = pixelValues[i];
}
result: waveTable = [-0.5, -0.2, 0, 0.9. 0.9, 0, -0.2, -0.5];
```

Effectively this is the same as a so-called palindromic reading of a wavetable. A palindromic reading reads from left to right but instead of jumping from the last value to the first of the wavetable, it reads the numbers from right to left, once it reached the last value of the table.

By using the mirroring above, the same result can be achieved while keeping the reading direction, the disadvantage being that the memory required for the buffer doubles. If the size of the array is supposed to remain the same, the values could be down-sampled (using only every second value) before copying them to the wavetable.

Adding timbral variability to scanline synthesis

Interpolating between wavetables

When working with digital color images, practically any color channel can be used in order to extract a value for a waveform, or – as mentioned above – the values can be recalculated in order to generate luminance values (grayscale). An interesting method is to work with multiple wavetables and to seamlessly interpolate between their values. For example, the R, G and B values of an image could be used to generate three wavetables.[5] By interpolating between the three wavetables, modulations in the timbre can be achieved which add liveliness to the sound that can otherwise easily sound too static.

Interpolation between two tables take place by morphing between the individual sample values of the two tables. Usually a blend value is used for the morphing which is a floating point number between 0 and 1. If the blend value is 0 the values of the first table are used; if the blend value is 1 the samples of the second table are used. If the blend value is 0.5 a value in the middle between the corresponding individual samples is used. In pseudo code interpolations can be expressed as follows:

```
tableA = [0, -0.2, 0.9, -0.5];
tableB = [0.5, 0.3, -0.2, -0.4];

blendTable = [0, 0, 0, 0];// the result of the interpolation
blendValue = 0.2;

for(i = 0; i <  blendTable.size; i++){
        blendTable[i] = tableA[i] + ((tableB[i] - tableA[i]) * blendValue);
};
```

If blendValue would be 0.2, the result would be: [0.1, -0.1, 0.68, -0.48].

[5] The alpha channel is usually not interesting for scanline synthesis, as – if it is present – it is commonly set to the same value for all pixels in a digital image.

Smoothening or harshening a waveform

Any sound can be encoded in an image, smooth sounds or sharp sounds. However, the principle of scanline synthesis will always generate pitched material, since a wavetable synthesizer keeps reading through a table over and over again. Therefore every waveform – disregarding how erratic it might be – will be played periodically, which results in a clearly recognizable pitch.

When applying scanline synthesis to concrete images, the timbre often tends to be rather harsh. As mentioned above any sudden change in the signal of the waveform emphasizes high frequency components. The data generated from digital images often contain quite a few edges, due to different colors or contrasts in brightness in the image.

If it is desired to tame those high frequency components or to even emphasize them, the data from the image could be transferred to the wavetable in a non-linear fashion via a so-called transfer function. A linear conversion would mean that the proportions between the individual values in the source is identical with the proportions of the corresponding values in the scaled target. Non-linear conversions, however apply functions during the conversion, e.g. exponantial functions.

Let us first take a look at two different transfer functions:

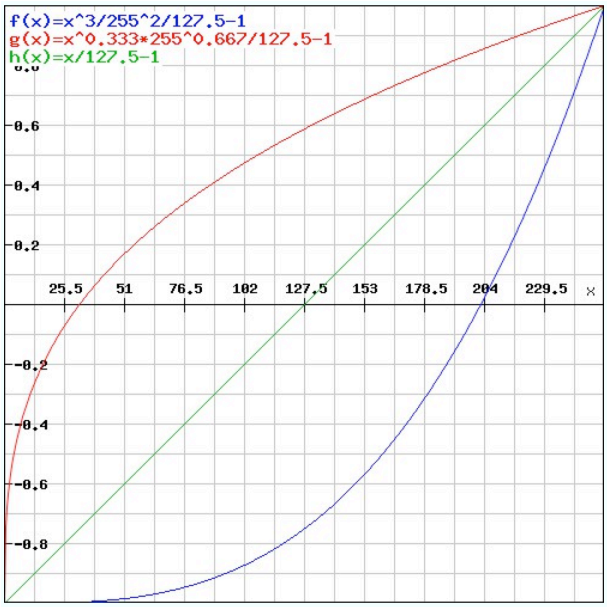

Figure 2: green line: linear scaling, red and blue line: different transfer functions

The green line shows the linear values how they are mapped from the 0 to 255 range (x-axis) to the audio range -1 to +1 (y-axis). The red and blue line are showing two different non linear transfer functions that behave like mirror versions of each other.

The blue line is a cubic exponential function. Here, the cubic values have been taken of each number from the pixel range (0-255). The results have then been divided by the square of 255,

in order to bring the numbers back to the range 0-255. Then they have been converted to the audio range. It is characteristic of this function that values in the low range are closer to each other when compared to the linear function, while the intervals start to get larger than in the linear function, when they are above a value of ~147.

Lets assume we have data from an image with very large contrasts in the low numeric range and we would like to make the differences between the numbers smaller. Here are our original values:

pixelValues = [90, 12, 67, 3]; // differences between the values: -78, +55, -64

If we apply the cubic transfer function, the intervals between the numbers get smaller:

cubicValues = [90^3, 12^3, 67^3, 3^3] / 255^2 = [11.21, 0.03, 4.63, 0]; //
approx. differences between values: -11, +4, -4

Smoother waveforms will lead to stronger bass components and less sharpness in the sound. If the opposite is desired, the same method can be used while using numbers smaller than 1 as the exponent. The red line shows such a transfer function in which the exponent is 1/3:

pow(1/3)Values = [$90^{1/3}$, $12^{1/3}$, $67^{1/3}$, $3^{1/3}$] / $255^{-2/3}$ = [180.21, 92.06, 163.32, 58]; // approx. differences between values: -88, +71, -105

In pseudo code, a transfer function for pixel values can be expressed as:

$pixelValue^{exponent}/255^{(exponent - 1)}$

Transfer functions can easily increase digital offsets, therefore the above mentioned method for offset removal should be applied. Also the amplitude of the signal can be strongly affected and might have to be adjusted.

Experience of congruence between the sonic result and the visual information

As mentioned above, any sound can be stored as an image, therefore it is not really correct to speak about a general character of scanline synthesis. However, with the vast majority of visual material, the resulting sound often has the aforementioned harshness, due to the unevenness of the resulting waveform.

In many contexts this quality might be exactly what is appropriate. However, in some contexts it might not be sufficiently flexible for more subtle musical expressions. In such cases it is highly recommended to use interpolations or transfer functions in order to introduce malleability into the resulting sound.

With scanline synthesis it is usually not possible to find a direct correspondence between a static image an the sound, except for rare cases, where the image leads to a very characteristic standard waveform sound as in the following example:

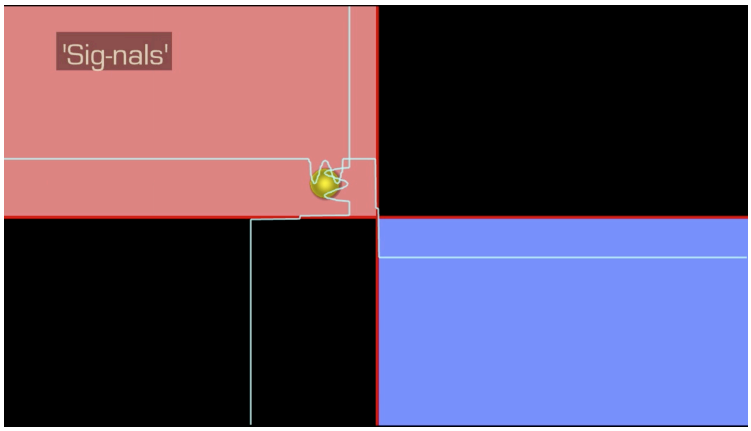

Figure 3: Ciciliani, Marko: *Formula minus One* (here, a vertical and a horizontal scanline are used simultaneously. They are displayed in light blue color)

In this case the symmetric composition of the image generates almost correct square waves which are characterized by the absence of even partials and therefore give it a somewhat hollow and clarinet-like quality. In this example the resulting waveforms are displayed as lines in light blue color.

With scanline synthesis the correlation between image and sound is more commonly experienced by the change of the image (when using films), or the change of the position of the scanline in time. In such a case it becomes obvious that changes in the reading position of the visual image are immediately reflected in changes of timbre. This is also the great strength of this method, as any changes in the image lead to fluctuations in the sound, even though the exact effect can not be predicted. This is why some artists decided to display the scanline in the visual part of the work, thereby making the process more transparent.

Ryoho Kobayashi, for example, used static images in his work *Scanline Computer Music*, but the scanlines are slowly moving across the image and thereby rendering audible constantly changing details. The scanlines are displayed as red lines with either horizontal or vertical orientation. In addition, he displays the extracted data as stripes of grayscale values that are placed on the right side of the image for the vertical scanlines, and underneath the image for the horizontal scanlines. Up to 8 scanlines are used simultaneously with every orientation, which leads to an impressive sonic richness, which is projected through a multispeaker setup.

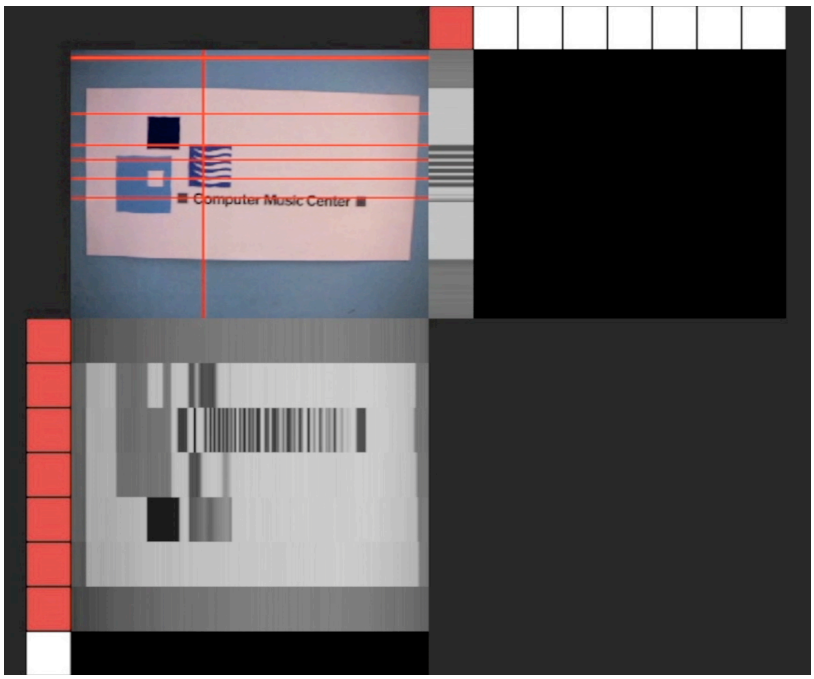

Figure 4: Kobayashi, Ryoho: *Scanline Computer Music*, here multiple scanlines are used on a static digital image. The values that are extracted are displayed on the right side for the vertical lines and underneath the image for the horizontal lines. Each line generates a separate wavetable.

In the next example, *Soundlines* by Christopher Jette, scanline synthesis is applied in a situation where a dancer directly interacts with the synthesis. This implementation is a different one than what was described above. Here, sound is only heard when the dancer moves. Apparently the algorithm is ignoring any pixels that have not changed recently – a technique also referred to as constant background removal. This points in an interesting direction for further investigation, as in this work, scanline synthesis is not merely used for the generation of a particular characteristic timbre, but it also generates rhythmic structure from the visual input. In contrast to the previous examples, Jette does not use any projection in order to display the resulting waveform.

Figure 5: dance performance in Christopher Jette's *Soundlines*, performance at CCRMA, Stanford

This example is also interesting because it presents a strong contrast between the organic movements of the dancer and the resulting sound, which has a erratic and stuttering quality with the typical harsh timbre. Here, two different media collide, allowing for attribute transfers to swap back and forth between the two.

* * *

Numeric translations between media should always be used with great care, if perceptual congruence is desired. Formal translations can neither guarantee that they are perceived as such, let alone that they are perceived as interesting.

Scanline synthesis is suitable for audiovisual contexts because it guarantees temporal congruence. In artistic contexts, however, semantic and mood-based congruences are at least equally important. Therefore the manner how scanline synthesis is applied has to be reevaluated and creatively adjusted depending on the context.

Conclusion and future investigations

The application of scanline synthesis in audiovisual projects is an interesting method to create a direct correspondence between a digital image and the resulting sound. When it is applied in its pure form, the musical variability is somewhat limited, but methods have been presented that are very suitable for adding sonic variability.

Aesthetically, formal translations should always be treated with caution, of perceptual congruence is desired. Scanline synthesis entails temporal congruence but semantic and mood-based congruence still have to be designed according to the artistic context.

How the sound relates to the chosen visual material still leaves a lot of space for experimentation. Future investigations could include the possibility of reversing the direction of data flow. In such a case the shape of a waveform would determine the image and not the other way round. Yet a further step could consist of combining both directions of data-flow and to investigate scanline based feedback constellations.

References

Smith, Steven W.: *Digital Signal Processing*, San Diego: California Technical Publishing, 1997

Wright, Jr., Richard S. et al: *OpenGL SuperBible*, fourth edition, Ann Arbor: Pearson, 2007