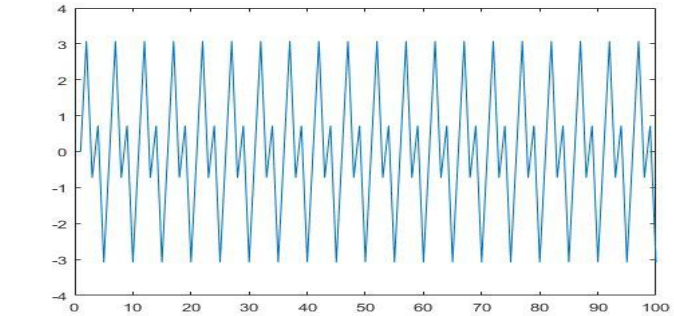
Reverse-Spectrogram Synthesis

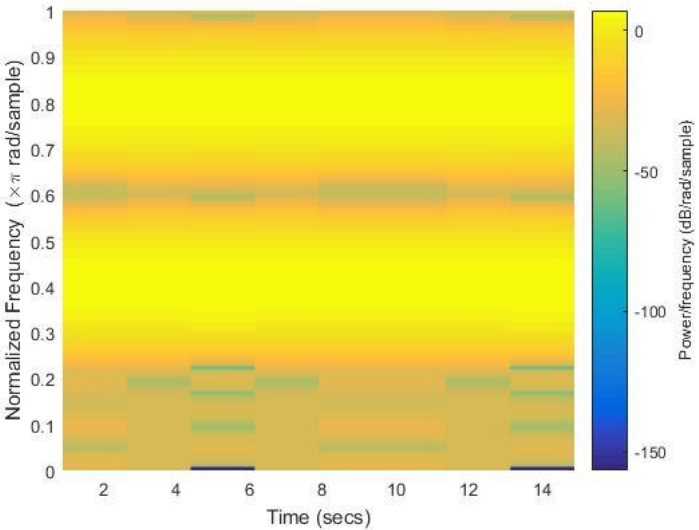
David Porter & Dan Waldman

V.T.N.H.R.

Goal: To generate audio based on an image, using the generic conventions of a spectrogram, but in reverse order. The output audio should be able to be non-harmonic frequencies or organized into musically related intervals. The user should be able to choose the input image, and control many of the parameters which dictate the details of the conversion, such as the length of the audio clip and the output frequency range.

What is a Spectrogram?



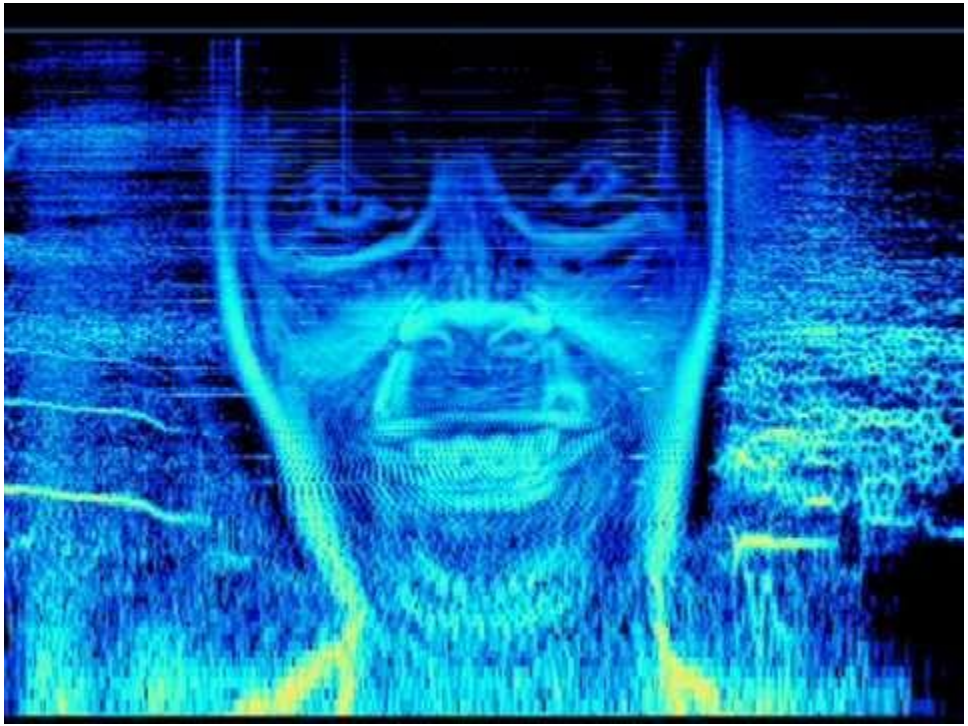


A spectrogram is an intensity plot of the Short Time Fourier Transform (STFT). Spectrograms have 3 dimensions, displaying the frequency (y-axis) versus time (x-axis), and showing with a range of colors the intensity of the signal at each frequency bin and time slice. Spectrograms are a popular form of visualizing audio data, as it is easy to see how the signal changes in power over frequency and time simultaneously.

Why Go Backwards?

People seek visualizations of sound that facilitate understanding, are novel, or are aesthetically pleasing. Conversely, artists and scientists occasionally turn visual art or representations into auditory representations. We would like to take an image and turn it into music. We are motivated to take the spectrogram, something we see as equally an important representation of data and an artistic object, and auralize it. That is to allow a user to make music out of something that previously had no auditory component, but has a significant visual in it, whether it be the user's face or another image of importance. There is a specific source of influence for this project coming from a song by experimental electronic music artist Aphex Twin's song "ΔMi−1 = −αΣn=1NDi[n][Σj∈C[i]Fji[n − 1] +Fexti[n−1]]", commonly known as "[Equation]". Videos showing the spectrogram of this song reveal several identifiable images in the spectrogram including a face. The sounds were generated using a program called MetaSynth, but we chose to design our own.

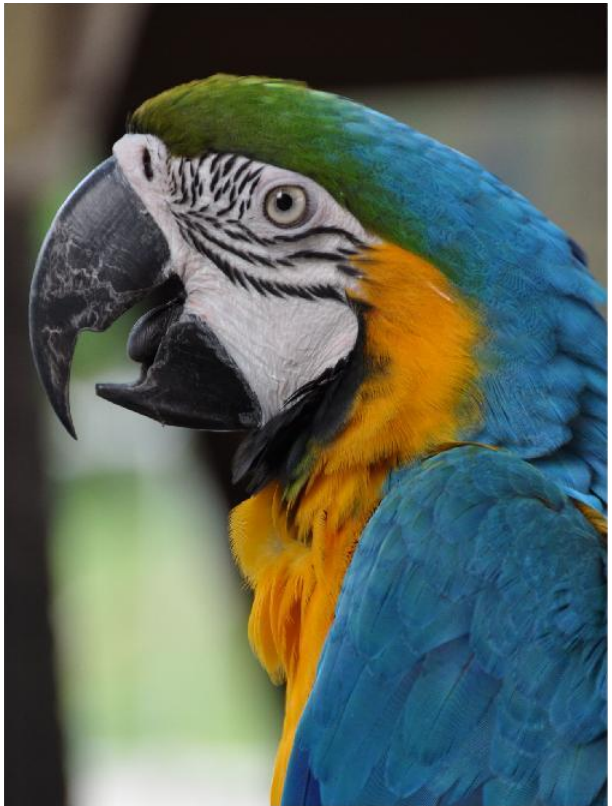
Screenshot from the spectrogram of the Aphex Twin song “[Equation]”



This is one of the images we used to test our code, in an effort to return this part of the song.

Step 1: Get Image

The user can choose an image to import.



Step 2: Convert to Grayscale

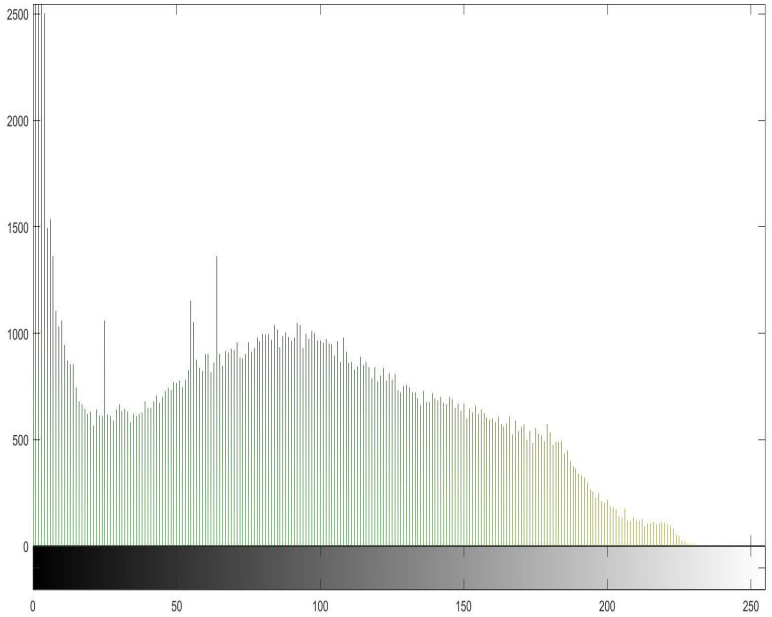
When matlab imports an image with imread(), it saves the image to a 3D array, with the 3rd dimension RBG values representing the range of colors. We convert to grayscale so that each element of the array has just one value, and that value represents the intensity of the pixel.

Step 3: “Segment” Image

The image is already organized into pixels, so we symbolically organize the image into segments by running our analysis code on one segment of the image at a time. The user may determine the horizontal and vertical resolution they desire.

Step 4: ‘imhist’, get a value for each segment

Using the ‘imhist’ MATLAB function, and a peak detection algorithm, we determine what the most common shade present in a given segment, and assign that value to the segment, in an array named ‘value.’ Either darkness or lightness can be made to represent ‘higher’ values.



Step 5: Generate audio

Initially, a sine wave is generated at each center frequency, for the duration of the clip (corresponding to the width of the image).

Step 6: Affect the amplitude

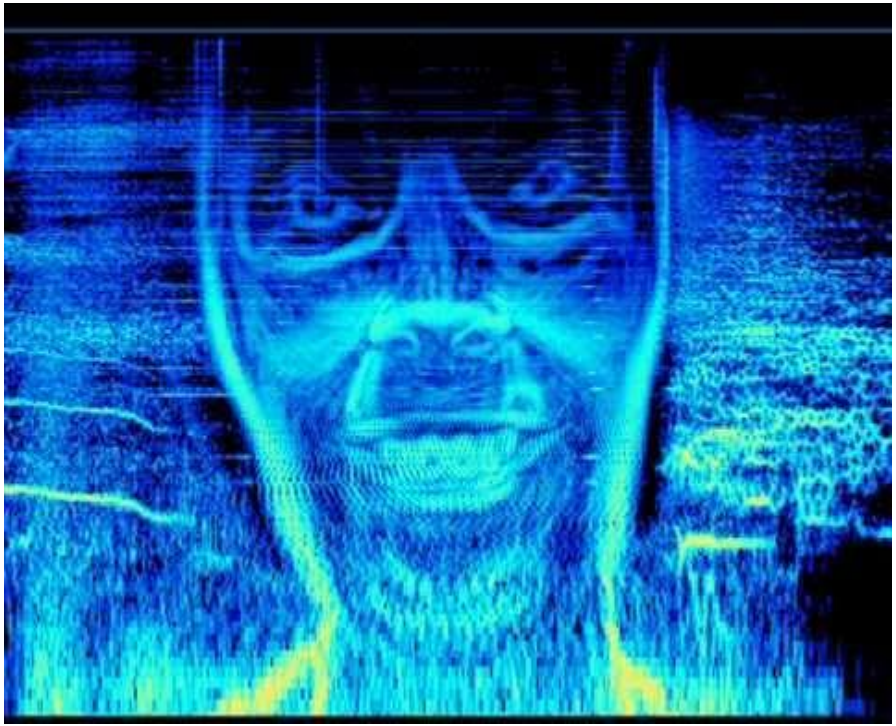
The sine waves are then attenuated according to the intensity values of their corresponding block.

Step 7: Sum Audio

Generate a single waveform by summing all frequencies in each time slice.

Step 8: Compare

Original



Spectrogram of output audio