Evan Fouche
CSCI 4964
Final Project Report
May 12, 2016

Displaying Digital Music Using Physical Familiarity

This project is an attempt to show the entirety of a digital music collection in one browse-able image, the inspiration for which comes from the familiarity of browsing music in one of its many physical packagings, vinyl records in particular [figure 1].

As music listening has shifted to digital platforms, ownership of music has gone from a physical object to a digital file. With this, the interactions that occur while exploring a music archive have completely changed. I, personally, do still like to go to record stores and dig through the crates of vinyl records, and from this perspective the design choices for this project were made.
Inherently a physical piece of music (e.g. vinyl, CD, cassette) takes up space and anyone could see in plain sight how much music they owned by looking at their stack of CDs, cassettes, etc. A digital collection just sits waiting to be cued, invisible until sent to one's favored digital music player.

The visual aspect of a piece that is meant to be listened to lies within its cover art. Cover art, in a physical form, is the second visual cue to mark a piece of music. The first being the recognition of the physical object: vinyl, cassette, etc.
I feel that some appreciation for cover art has been lost in the shift to digital. What used to be *the mark* of a piece of music has become an associated image file packaged in a folder of music files, or it may not even be present at all. I wanted to bring back some love for cover art.

Cover images are the data I used to represent the digital music collection. The issue is that a downloaded digital piece of music does not always come with an associated image file. I used an application called Automatic Cover Tool that crawls through a music directory and downloads cover art from discogs.com an places them in the album folders. It was able to grab most of my cover art, but it was not feasible to check and see if the cover art was, in fact, the official cover art. I noticed at least a few of them were not the official cover art and an additional few were photographs of a CD, vinyl, etc.

I chose to use a familiar vinyl record collection to model my project after [figure 1] and drew additional inspiration from a “movie barcode” visualization that we encountered earlier in the semester [figure 2] [image]. In this form, a collection is comprised of music by album[1]. In this format, records are filed upright in rows, with only their edges being viewable. My visualization follows this. I had to create edges for view and file them in rows. I used two different methods for creating edge images. One is simply taking the cover art image file and squishing it to a fraction of its width, following an edge to face ratio similar to that of a vinyl record or CD [figure 3]. The other method for marking an album edge was to find the dominant color found in its image. I borrowed a piece of Python code from github user, Stefan Zollinger that grabs the dominant colors of an image and outputs them as a palette display for the image. I modified this code to output a vertical display of palette colors as a representation of a record's edge [figure 4].
The Python code that I wrote crawls through my music directory organized by album, grabs the image in the album folder (this should be the cover art) and creates a squished edge image and a palette edge image. Data about the album (i.e. artist and album title) is pulled from the album's folder name. The

1 For the most part. There are singles in the form of 45rpm records.

Python code creates a dictionary of edge images as keys with a list of artist name, album title, and original cover art as their value. These dictionaries are used to generate json objects. All images found and generated are sent to an “images” folder for use by Javascript. The Javascript code displays all edges in rows like record shelves. When hovering an edge, its associated, original form (cover art) and associated data (artist and title) are displayed below the rows. This is akin to choosing a record from a shelf to view its front. Figures 5-12 show the current working version using my music library. It shows 384 album cover “edges”.

I want to do more for this display. Currently only the cover art is displayed on hover and not its associated data. I would like to implement functionality to organize the collection alphabetically and by color. Ideally it could be organized by genre like a record store or for my interest, by record label, but for that I would need to use some other data archive with that information available. As of now, it can output the squished version of album edges or the dominant color version. I would argue for the dominant color version as I think it is nicer to look at and maybe easier to discern where one album ends and another begins. The dominant color palette display runs into trouble where similar color displays may make it harder to notice where one album ends and another begins.

I like the visualization although it begs to have a comparison. I would like to see how someone else's music display compares to my own. With the added functionality as described above I could see it being more useful. It is fun to run the mouse over the edges and watch the covers change rapidly. Some of the cover art images do not load [figure 8]. I have not found a fix for this yet.

I plan on making this a usable product. A user would submit their music directory and an interactive image of their digital music collection would be displayed for them.

Figures:

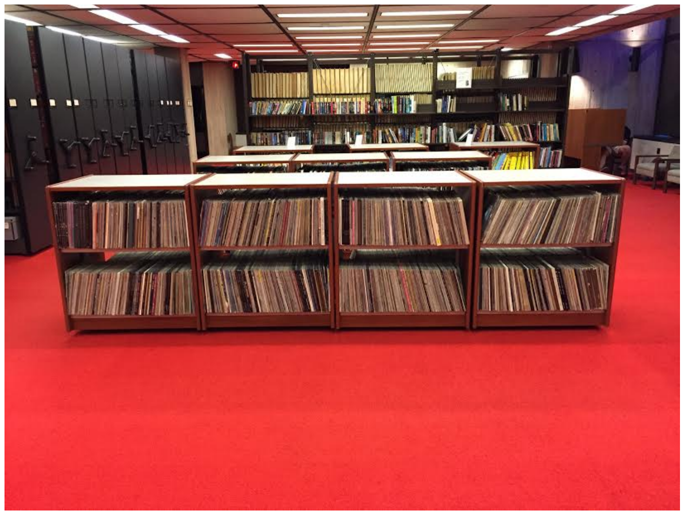

Figure 1: Record shelves on the first floor of Folsam Library

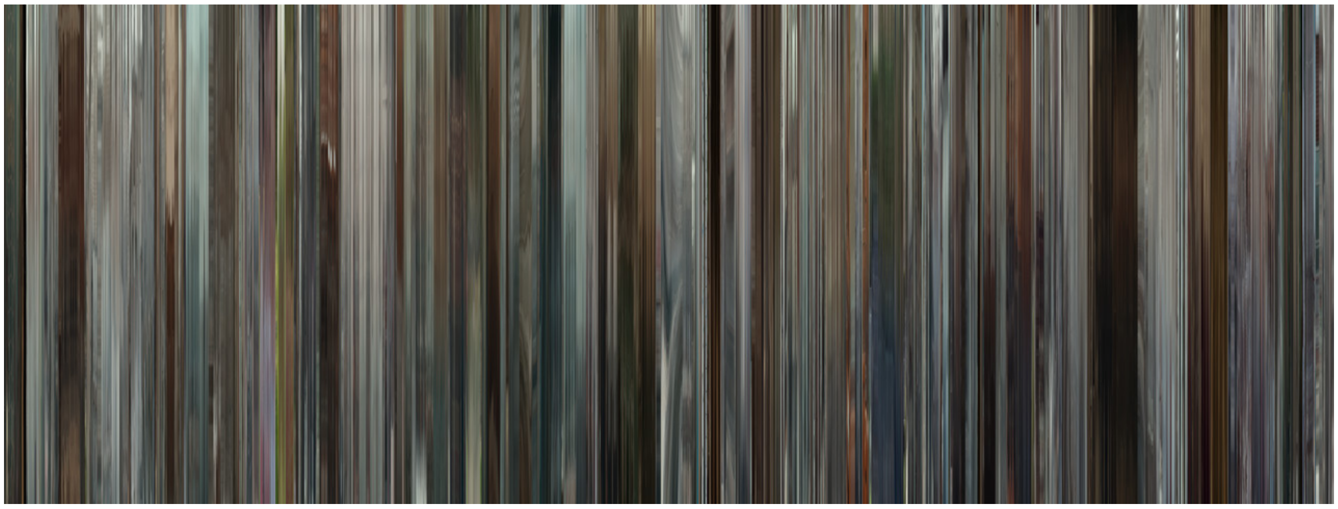

Figure 2: A 'Movie Bar Code' image

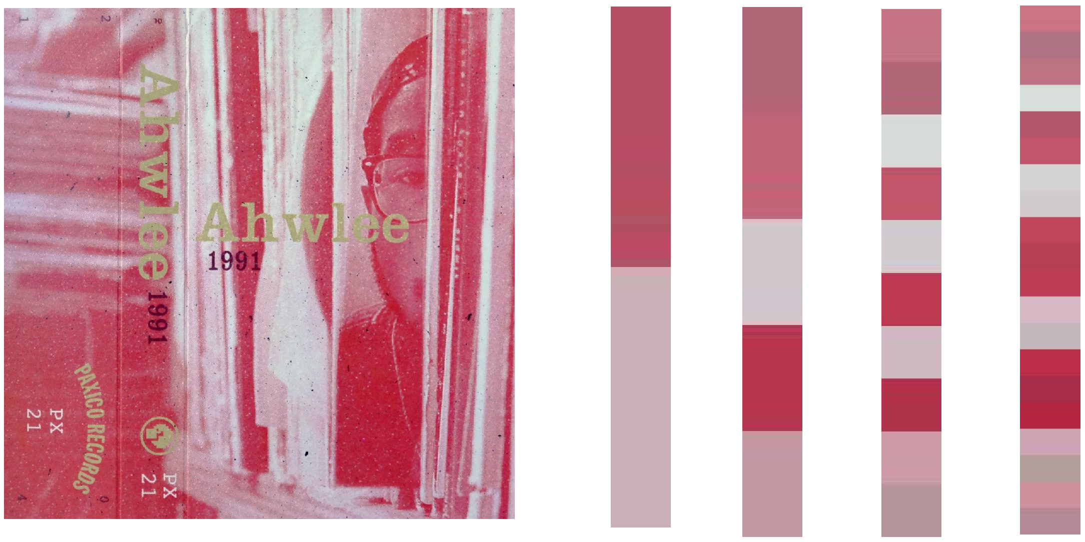

Figure 3: left to right: Cover art, 2 color dominant palette edge, 5 color dominant palette edge, 10 color dominant palette edge, 20 color dominant palette edge

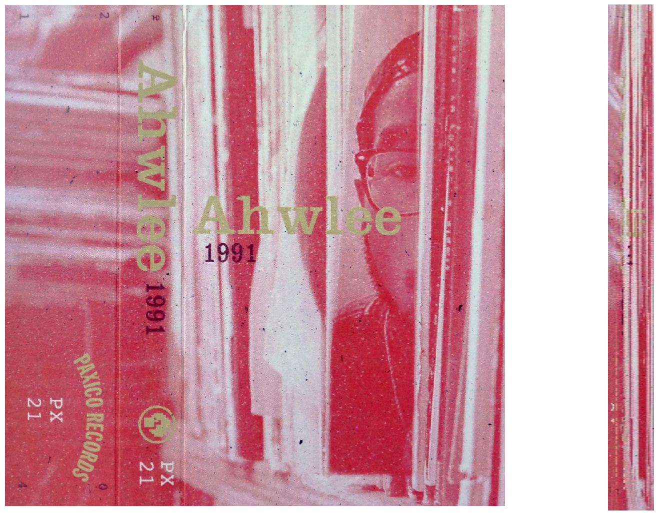

Figure 4: Cover art and squished edge

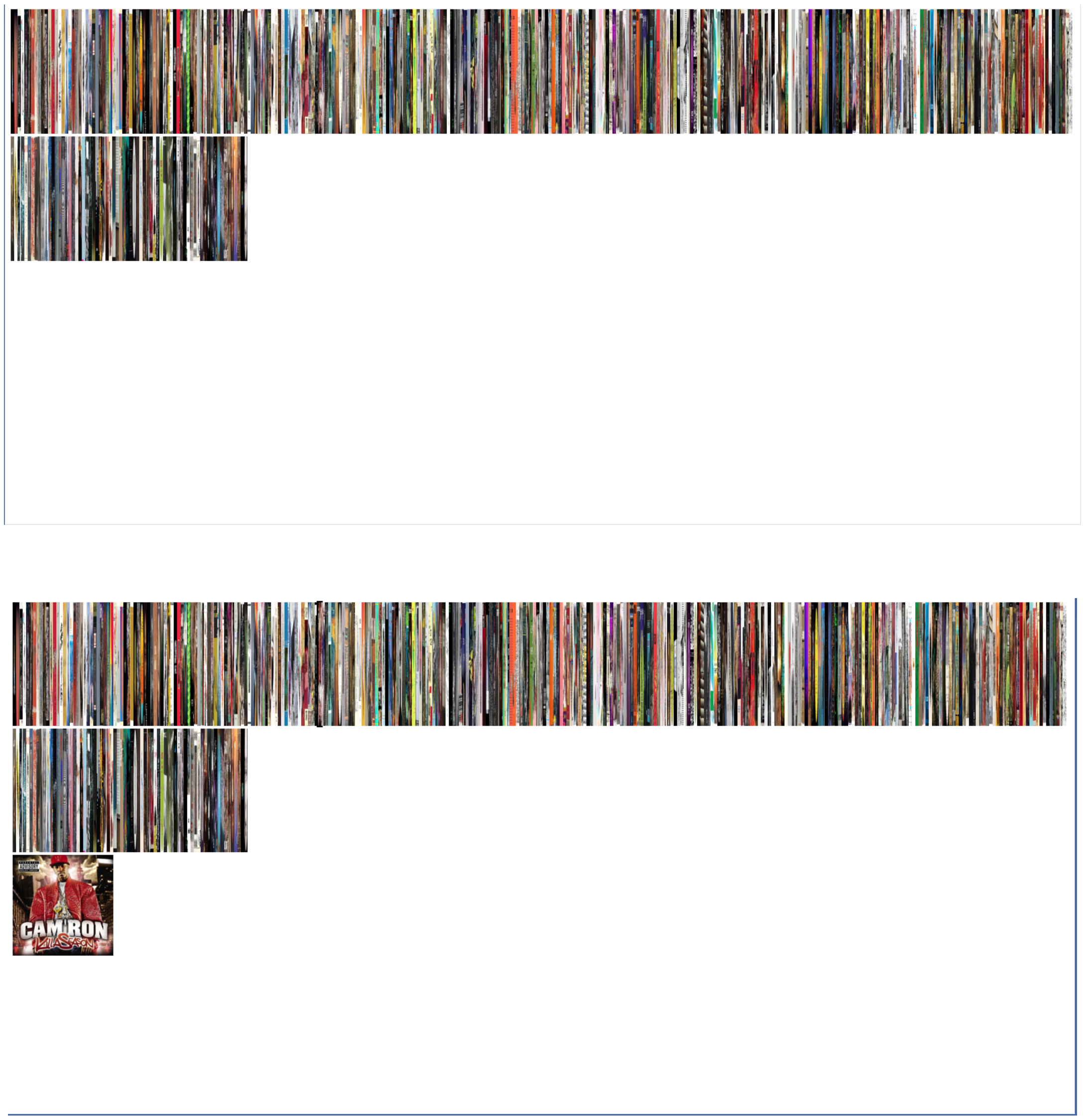

Figure 5: squished edge display without (top) and with hovering

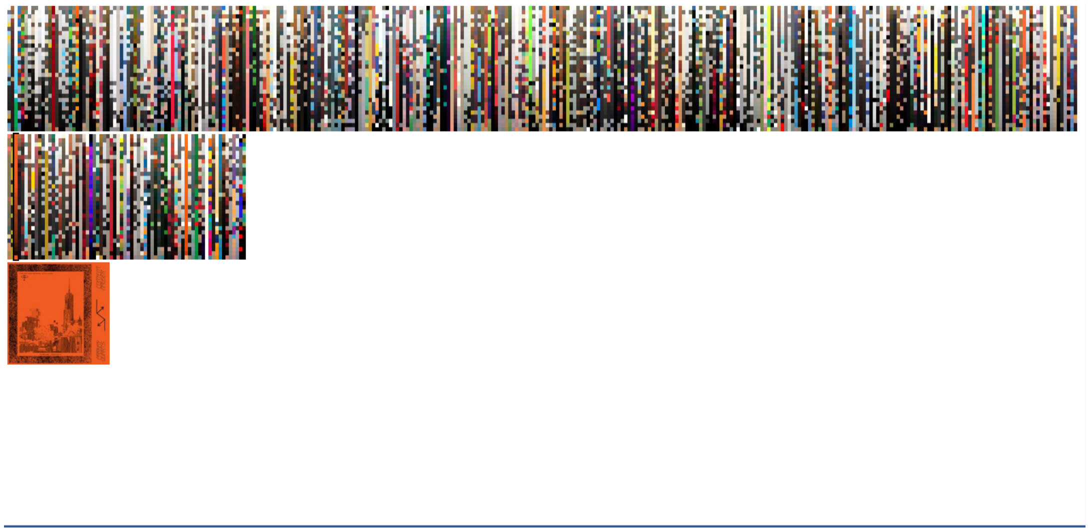

Figure 7: Color palette edges using 30 dominant colors

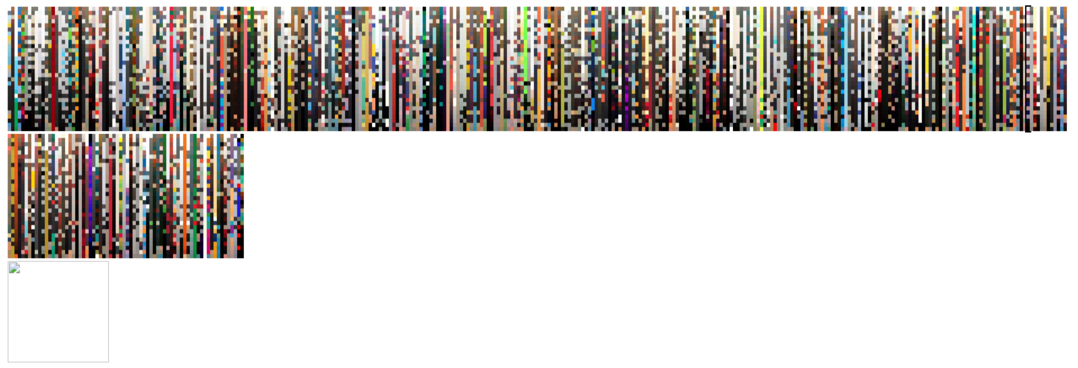

Figure 8: Cover art loading error

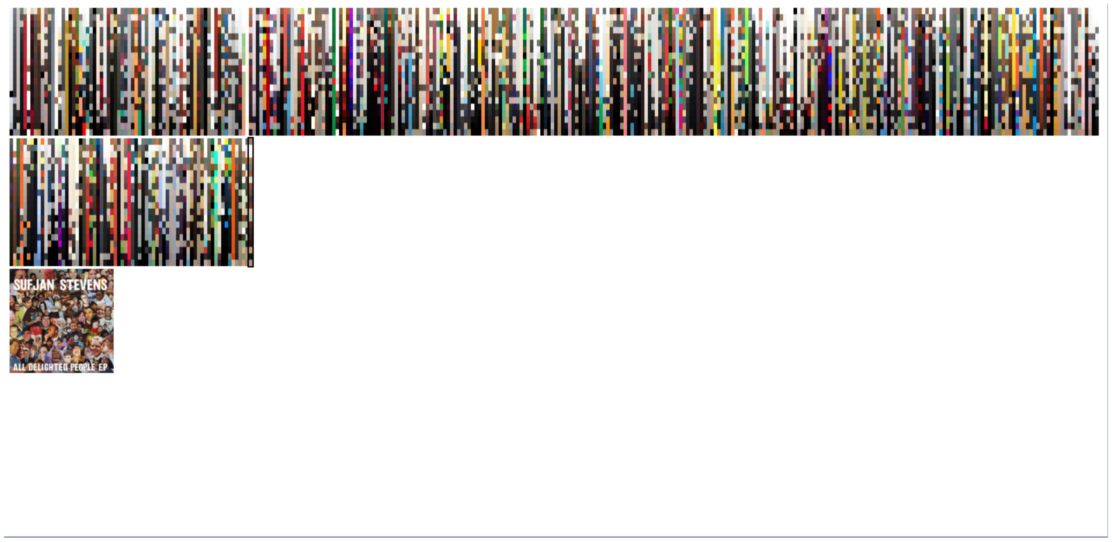

Figure 9: Color palette edges using 20 dominant colors

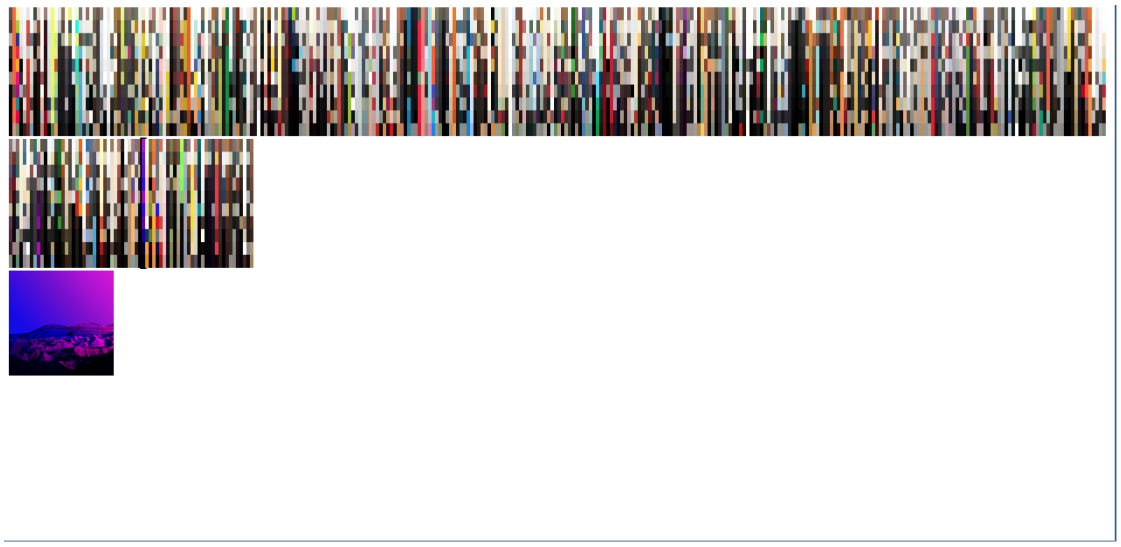

Figure 10: Color palette edges using 10 dominant colors

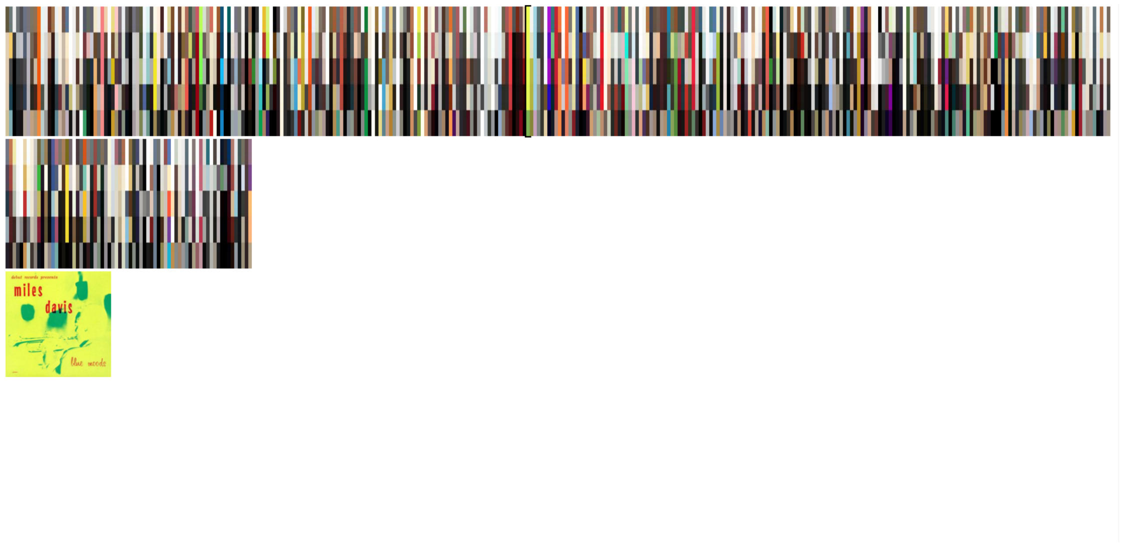

Figure 11: Color palette edges using 5 dominant colors

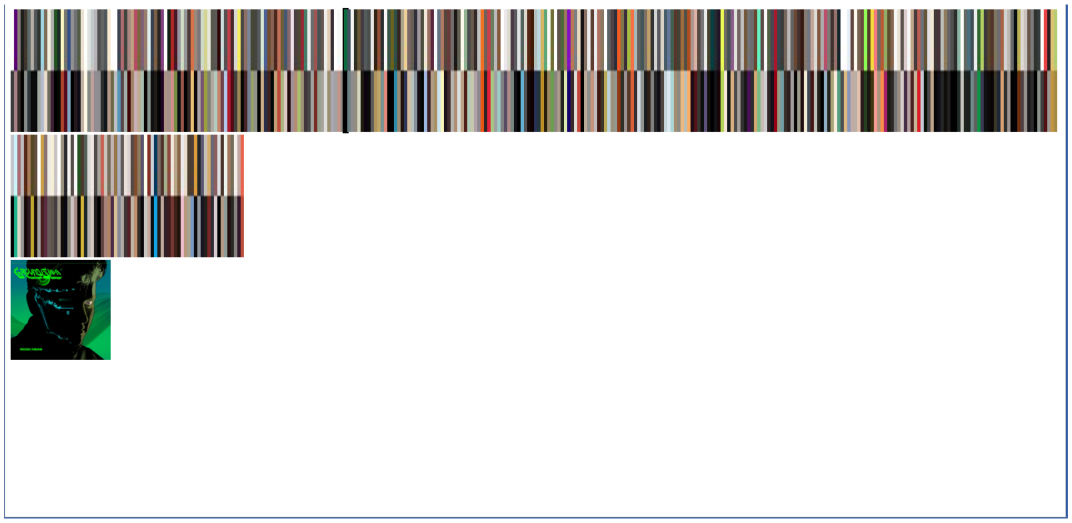

Figure 12: Color palette edges using 2 dominant colors

Works Cited

Stefan Zollinger's color palette:
https://gist.github.com/zollinger/1722663

Automatic Cover Tool:
https://sourceforge.net/projects/act-mp3/

Movie Bar Code:
http://moviebarcode.tumblr.com/image/144254683578