From Image to Sound: Transforming Digital Image Data into Algorithmic Music

Jacob Richard

Supervisor: Kevin Morse
Independent Summer Research Grant Report
September 15th 2024

This student research projet is generously supported by the JEA Crake Foundation

There is a long-standing tradition linking music and the visual arts. For example, classical composers have composed string quartets inspired by paintings. With the increased digitization of both auditory and visual media, new opportunities have emerged to explore the intersection between these two mediums using their digitized data. In the last century, the rise of electronic music has further expanded this exploration, with composers experimenting with technological and scientific systems, large data sets, and mathematical equations to create sound.

My research aims to explore these opportunities, seeing how the raw data extracted from digital images can be used to create electronic music, developing new creative possibilities and fostering a relationship between digitized images and electronic music. This research will serve as the foundation for an independent capstone project I will undertake this year, a requirement for the joint major in computer science and music in which I am enrolled. I plan to use and build upon the work completed during this research phase, integrating my findings into a musical composition.

While using image data to create music offers innovative possibilities, it raises the question of whether similar results could be achieved by simply tweaking the parameters of preexisting effects for making electronic music. Instead of processing the visual information, one could directly manipulate functions such as reverb, delay, and pitch-shift within a digital audio workstation (DAW) to produce a similar result. This approach might seem more straightforward and efficient, prompting consideration of whether the added complexity of translating the visual data into audio truly offers unique creative advantages or merely replicates effects that can already be achieved through conventional methods.

Unlike conventional audio effects, where changes are made by tweaking parameters through abstract numbers and sliders, using images as a basis for sound manipulation gives more agency to the relationship between the music and the parameters controlling the effects, as it also leaves an immediate impact on the image. As an example, an artist could experiment with altering an image's colour balance, brightness, or contrast in choosing the size, reflections, and delay on a reverb effect. Conventionally this would be done by tweaking these parameters using arbitrary numbers and sliders until the desired sound is reached. However, it can be difficult to perceive the small auditory changes so having the balance, brightness, and contrast of the image control the size, reflections, and delay of the reverb allows the artist to both see and hear immediate feedback as the sound changes. This also could be used as an effective creative tool, making the process more engaging and meaningful.

Linking visual data to sound fosters a more intuitive and exploratory creative process, making it easier to understand the impact different manipulations have on the resulting sound. For example, to achieve a brighter sound, you might adjust the high-frequency bands on an equalizer (EQ) by modifying the arbitrary values that control the filter's various parameters such as its height and width. However, when using an image-based approach, this same effect could be controlled by linking the brightness data to the settings of the EQ. As you increase

the brightness of the image, the corresponding high frequencies in the audio is enhanced, thus brightening the sound. This method not only produces the desired auditory effect but also provides a more intuitive and visual way to understand and control the changes. The direct manipulation of the image creates a tangible link between the visual and auditory allowing for a more intuitive and creative process.

The addition to enhancing the UI experience for controlling parameters, this approach also has creative benefits. In an electronic musical performance, the inclusion of images and image data could transform the performance into a multimedia experience where the interplay between sight and sound would become central to the creative process. As an example, an improvisatory electroacoustic music performance, where live sound from the performer is processed in real-time using effects, could incorporate a program to generate various audio effects using data from a digital image. This approach creates a dynamic three-way interaction between the performer, the images, and the audio effects, where each element continuously responds to and influences the others. This would create an immersive and dynamic experience, inviting the audience and performer to embark on a multi-sensory, interactive journey where each moment is unpredictable and unique.

I set myself three primary goals for the summer. I tailored my goals to serve as the foundational pieces for my capstone project which I will be doing during the upcoming year. I began by familiarising myself with the language and conventions of digital images, and understanding how their data is stored and organized. My next goal was to educate myself on various image manipulation techniques and see how they could be applied to music creation. Lastly, I learned how to implement these techniques, by learning tools such as MaxMSP, JavaScript, and Ableton Live.

Working with digital image data

The first few weeks of the research focused on familiarising myself with digital image data. The two fundamental types of digital images are raster and vector images, each with distinct characteristics and applications. Raster images, also known as bitmap images, are composed of individual pixels arranged in a grid. Each pixel holds information on its colour and together they form the complete image. In contrast, vector images are created using mathematical equations and geometric objects such as points, lines, and curves.[1] Raster images are best for detailed and static graphics like photographs, while vector images are ideal for scalable graphics like logos and illustrations. For this project, I chose to use raster images, specifically JPEG images, because my focus is working with photographs.

---

[1] Adobe, "Vector Image File Formats: The Most Common Types," *Adobe*, accessed May 27, 2024, https://www.adobe.com/ca/creativecloud/file-types/image/vector.html.

All digital media, including images, are fundamentally just a series of 0s and 1s. Every piece of information, whether it's a pixel in an image, or a note in a song is ultimately represented by a unique combination of these binary digits. For example, the following string of bits is the binary representation of a simple JPEG image of a 5x5 grid of arbitrary colours that I created.

Figure 1.1 JPEG image of this 5x5 grid of pixels.

```
010011100001011100110110101010100111101110011101011010100
1000111011011111011001110001101101001101100001010000010001
1110011100011010101100110100011001100101111110100111111100
000000110001001111110101111001001100010100100011011110100
01010011010100000111001010010101011111001010110110011111
00001101010010100010101010011111111111110001110100011 0000
1111000011101010010101100101100101011111111101011101011001
01011111010011011100100011000010100000111000000001000000000
1110001011110110110111011011100101010110010000001011100010
0110011000111101000011010001111011111000011110101100000101
0100111111001011111111000000000111111011
```

Fig 1.2 Binary representation of Figure 1.1.

However, within this seemingly obscure and endless string of bits, there must be a systematic way to distinguish and organize different elements, such as individual pixels in an image. File structures and formats define specific patterns and rules for how the data is stored and interpreted, providing the necessary framework to systematically decode and arrange the data, transforming the raw binary string into a coherent image. By adhering to these predefined rules, the system can accurately distinguish between different elements, such as colour information and pixel placement, allowing each pixel to be precisely identified and displayed within the overall image.

Raster images can be broken into two parts, an image header and a list of pixels comprising the image. The header of a raster image file is structured to always be the same length for easy identification and includes important metadata about the image, such as its dimensions (width and height in pixels) and its bit depth.[2]

[2] "JPEG - Image File Format," *JPEG - Image File Format*, accessed May 23, 2024, https://docs.fileformat.com/image/jpeg/.

Bit depth refers to the number of bits used for each channel, colour channel and each channel represents a different colour component of the image. I decided to use a standard format of alpha, red, green, and blue (ARGB) as my colour channels for this project.[3] With a bit depth of 8, each of the four channels has a binary representation ranging from 0-255 (2^8) which represents that channel's presence within the pixel's final colouration with 0 meaning a complete absence and 255 indicating fullness in that channel. The alpha channel, which represents opacity, is always set to 255. Its inclusion is to ensure that the length of each pixel is 32 bits which coincides with the length of an integer, making traversal and indexing the list of pixels convenient.[4]

As mentioned, pixels in a digital image are stored as a continuous sequence of bits. A digital image organizes these bits into a two-dimensional matrix of pixels defined by the image's width and height. The stride of an image represents the number of bits in each row of this matrix. It's important to distinguish between the stride and the width of an image because the stride often includes extra bits of padding at the end of each row that are not part of the image's visible width. This padding is accounted for in the stride, not in the width.

Understanding the stride is crucial when you need to locate the data of a specific pixel within this 2D matrix based on its x and y coordinates. The starting position of a pixel in memory can be calculated using the following formula:

$$Index = (y * Stride) + (x * Bit\ Length)$$

Here, multiplying the stride by the y value gives the starting position of the row containing the pixel, and adding the product of x and the bit length tells us how many bits to move across to find the specific pixel within that row.[5]

Knowing how pixels are represented by binary bits, how these bits translate into colour values (such as ARGB), and how these values form the visual content of an image is imperative to transforming this data into sound. Without this foundational knowledge, extracting meaningful data and processing it to be used to influence musical parameters would be impossible, as the relationship between the binary data and the visual content it represents would remain obscured.

---

[3] JPEG images use YCbCr as their colour space, separating the image into one luminance (Y) and two chrominance (Cb and Cr) components each with a bit depth of 8, however, MaxMSP, the program I used this summer automatically converts the colour space of JPEG images from YCbCr to alpha, red, green, and blue (ARGB) instead.

[4] *Cycling '74 Documentation*, "Tutorial 5: Matrix Data and Planes," accessed May 26, 2024, https://docs.cycling74.com/max8/tutorials/jitterchapter05.

[5] Sean Riley, "Digital Images - Computerphile," YouTube video, 10:34, uploaded by Computerphile, February 15, 2015, https://www.youtube.com/watch?v=06OHflWNCOE&ab_channel=Computerphile.

Moreover, understanding how to process image data enables one to manipulate it in ways that directly affect the resulting music. Different image processing techniques, such as adjusting brightness, contrast, or applying filters, alter the pixel values and, consequently, the data that can be extracted for musical use. For instance, by transforming the bits into pixels and then into the four colour channels (ARGB), one can map this information onto a 2D graph. This graph can then be used as a function to control various musical parameters, such as volume, pitch, or rhythm over time. This process of translating visual data into auditory output requires not only a technical understanding of how to manipulate image data but also an artistic vision of how these manipulations will influence the final sound, making the study of image data essential for this innovative approach to music creation.

Throughout the summer, I consistently developed applications using Max/MSP, a visual programming language, to explore and implement the concepts related to my project. I chose Max/MSP because it allows for real-time manipulation of digital audio signals without needing specialized hardware. Additionally, it includes Jitter, a library that enables image/video manipulation and offers built-in support for creating JavaScript objects. This combination made it an ideal tool for me to experiment with different methods of translating visual data into sound, while also helping me become proficient with the software.

In Max/MSP, images are stored as matrices, which are like grids where each square, or cell, represents a single pixel in the image. Each cell contains information about the pixel's colour, specifically the values for red, green, and blue. To better understand and work with this data, I created a program (known as a patch in Max/MSP) that moves through each pixel in the image and plays a three-note chord based on the pixel's RGB values. While Max/MSP has a built-in tool that can move through the image, it doesn't allow you to control the speed at which each pixel is processed. My program addresses this by adding a feature that lets you adjust the speed, so there's a pause between each chord, giving the sound more structure and allowing for a clearer interpretation of the image data.

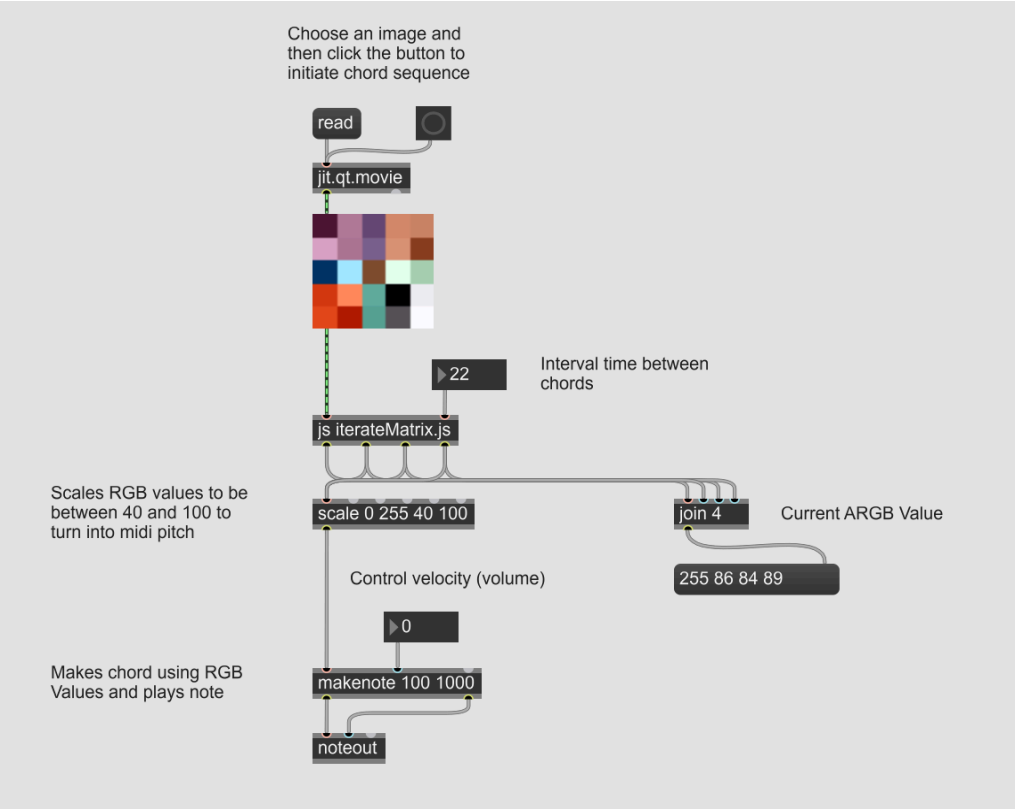


Figure 1.3 MaxMSP patch that plays chords using RGB values. This patch traverses each row from top to bottom. It iterates through each pixel and extracts its RGB values. It then scales this value to be between 40 and 100 to be converted into midi data. This translation is done to put the note within a comfortable range in the piano so that it is audible. It then waits x amount of milliseconds controlled using a parameter before moving on to the next pixel.

Linking image data to music creation

After learning how to extract data from images, I began exploring how to use this data to create electronic music. My goal was to create relationships between the visual elements of the image and the resulting music and to understand how altering the image would affect these relationships and the overall sound. I tackled this by breaking down rhythm, pitch, and timbre and analyzing each one separately. Throughout the summer, I focused on how these aspects of composition could be influenced by visual data, experimenting with different approaches to see how changes in the image would translate into changes in the music.

Data and Rhythm

Rhythm in electronic music has undergone a significant transformation shaped by technological innovations and changing artistic priorities. Early electronic music, emerging in the mid-20th century, often focused more on texture and timbre than on rhythm. As electronic music began to intersect with popular music in the late 20th century, rhythm took on more importance. The development of drum machines in the 1970s revolutionized how rhythm was constructed in electronic music. These machines allowed for the precise programming of repetitive drum patterns, which became a defining feature of genres like house, techno, and hip-hop.

As electronic music continued to evolve, so did the complexity and diversity of rhythm. The advent of digital audio workstations (DAWs) and sampling technology in the mid-late 20th century allowed for even greater experimentation with rhythm. These samples could be looped and manipulated to create fast, syncopated patterns to create rhythmic structures from arbitrary sounds.

A common challenge in electronic music is avoiding excessive repetition, as the music often loops through generated patterns or samples, leading to a monotonous sound. How can data extracted from JPEG images be used to create a logic that manages to solve this problem? In designing a program, my goal was to create a flexible framework for rhythmic structure rather than a fixed rhythmic pattern. By focusing on generating a rhythmic structure using the data extracted from an image, I can establish the overall "feel" of the piece while allowing for the creation of rhythmic elements like drum patterns and melodies with varying degrees of randomization, ensuring that the music remains cohesive yet dynamic.

My goal in using image data for rhythmic creation is to address the challenge of excessive repetition in electronic music by developing a flexible framework for rhythmic structure. Instead of relying on fixed patterns that can lead to a monotonous sound, I aim to generate rhythmic structures directly from the data extracted from JPEG images. By doing so, I can ensure that the music remains cohesive yet dynamic, with the visual data driving a constantly evolving and engaging auditory experience.

I decided to draw on the ideas of Australian composer Matthew Whitley using a rhythmic framework or “clave” as a structure for determining rhythmic content. The principle revolves around creating a rhythmic phrase to be the basis of our sequence. This phrase is created by first determining its length in beats. Downbeats, also referred to as landmarks, are then placed across this series of beats separating them into bars. Further landmarks are then placed to determine rhythmic events within a bar and are used to determine where rhythmic events such as a snare hit or chord change may happen.[6]

---

[6] Matthew Whitley, “Algorithmic Music Generation: Solving the right problem,” presented at the Australasian Computer Music Conference (ACMC) 2020, [July 6th, 2020], https://australasian-computer-music-association.github.io/acmc2020/poster_81.html.

To implement the idea of a rhythmic key or "clave" using the extracted data from our image, we can draw a parallel between the rhythmic structure and a binary string. As mentioned earlier, a binary string contains 0s and 1s. Given a binary string of any length, we can create a bar of the same length, with the rhythmic landmarks corresponding to the positions of the 1s in the string. By repeating this process a couple of times, the rhythmic phrase can be given by adding the bars together serving as the clave. For example, consider a binary string like "1000100010001000." Here, each '1' marks a landmark, and the zeros between them determine the spacing between these key points. This can be directly correlated to the ARGB values of a pixel in a digital image when expressed in binary. The binary representation of each colour channel can be segmented into four bars that together make a complete phrase.

The binary data for each segment needs to be processed due to two key issues. First, if we use the raw data directly, each segment will always be 8 beats long because the channels have a depth of 8 bits. Second, there's a potential complication if the binary sequence starts with a 0, making it difficult to determine the downbeat of the bar. I decided to convert the numerical value (0-255) of each channel to binary instead of using all 8 bits as this can create irregular bars and ensure that the bar has a downbeat. As an example, a pixel with a blue value of 100 is stored as "01100100". Leading 0s are added to preserve the bit-depth of 8. By converting the numerical value separately, the resulting binary string is "1100100". Using this approach, a leading 1 is guaranteed as well as an irregularity in the number of beats per bar.

There is one final issue that must be addressed. The alpha channel in any pixel is always 255. To ensure variability, I opted to use a different channel in its place. Grayscale refers to a range of shades of gray without any colour. In a grayscale image, each pixel represents a brightness or luminance level, where black is the darkest shade (0% brightness), white is the lightest shade (100% brightness), and various shades of gray fall between these two extremes. Grayscale images can be formatted to be composed of 256 levels of gray, represented by an 8-bit value ranging from 0 (black) to 255 (white). Unlike the alpha channel, a pixel's grayscale value is dependent on its RGB values thus making it variable and a great replacement for creating the clave. To convert RGB to grayscale, a weighted average is calculated using the National Television Standards Committee's algorithm which reflects how our eyes perceive luminance:[7]

$$Grayscale\ value\ = 0.299R\ +\ 0.587G\ +\ 0.1145B$$

[7] Riyad Mitieb Mahmood, "Color Spaces Analysis for Luminance & Chrominance Signals AS NTSC-TV System," *Journal of Engineering and Sustainable Development* 15, no. 4 (2011): 123–34. Accessed [July 20th 2024]. https://jeasd.uomustansiriyah.edu.iq/index.php/jeasd/article/view/1312.

In practice, the jit.rgb2luma object can be used to convert a coloured image to grayscale. The object iterates through each pixel, calculates its grayscale value and then stores that value into a matrix.

To extend this rhythmic technique from applying to a single pixel to the entire image, I began by averaging the RGB and grayscale values using the jit.3m object which returns the max, mean and minimum value across the entire image. This approach is more significant as it encompasses the broader visual content of the image, rather than focusing on an individual pixel. By using averaged values, the generated clave serving as our rhythmic structure becomes more representative of the overall image.

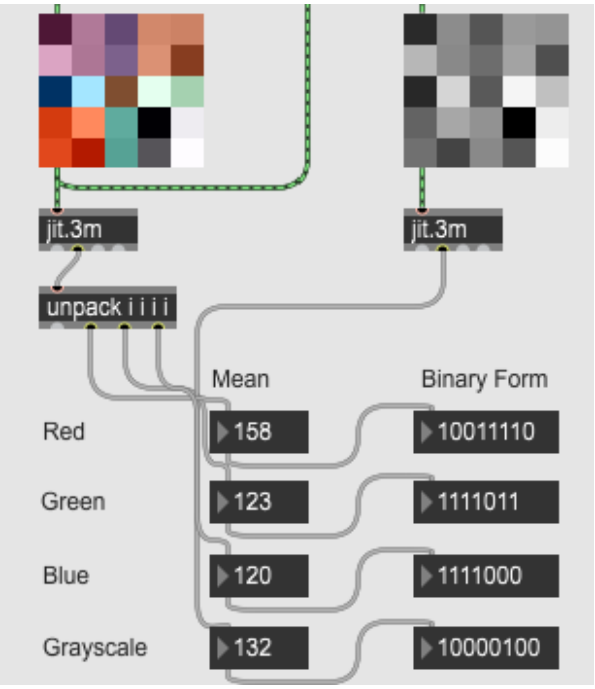


Figure 2.1 MaxPatch that retrieves the binary strings of the average RGB and grayscale values. This patch uses the jit.3m object which returns the max/mean/min values of each colour channel in that image. In the case of the coloured image, the mean ARGB values must be separated into their values by using the unpack object (i i i i indicates that four integer values are to be broken into individual segments).

Finally, the clave is generated by appending the binary strings of each colour channel and the grayscale value. This composite binary sequence forms the rhythmic framework that drives the music, resulting in a sequence that is both cohesive and dynamic, reflecting the image's broader characteristics. The irregularity in the binary data often produces odd

rhythms, such as alternating bars of 8 and 7 beats, which adds an element of unpredictability to the music. For example, as shown in Fig 2.1, the binary strings derived from the image data create the following sequence: "10011110, 1111011, 1111000, 10000100." This can be further visualized in terms of downbeats (D) and rhythmic landmarks (L) as: "D00LLLL0 DLLL0LL DLLL0000 D0000L00." By creating a framework instead of a fixed pattern, this method not only avoids excessive repetition but also maintains a strong connection between the visual and auditory elements of the piece, resulting in a more engaging and integrated musical experience.

As mentioned, the advantage of the clave lies in its ability to serve as a flexible and dynamic rhythmic framework, enabling the generation of various rhythmic elements such as drum patterns or melodic rhythms. This framework allows for the creation of rules based on the clave that dictate how each element behaves, adding layers of intricacy to the composition. For instance, I designed a snare drum pattern that incorporates an element of randomness. Instead of having the snare play at every rhythmic landmark, I wanted it to introduce an element of surprise by playing more frequently in the middle of the bar. To achieve this, I generated a bell curve that increases the probability of a snare hit occurring in the middle of the bar, while reducing the likelihood of it playing at the edges. When the clave's landmarks indicate a potential snare hit, the program rolls a random number and if the result falls within the desired range, the snare is triggered. This approach ensures that the overall feel of the rhythmic framework established by the clave is maintained, while the snare pattern remains dynamic, mitigating the issue of excessive repetition often found in electronic music.

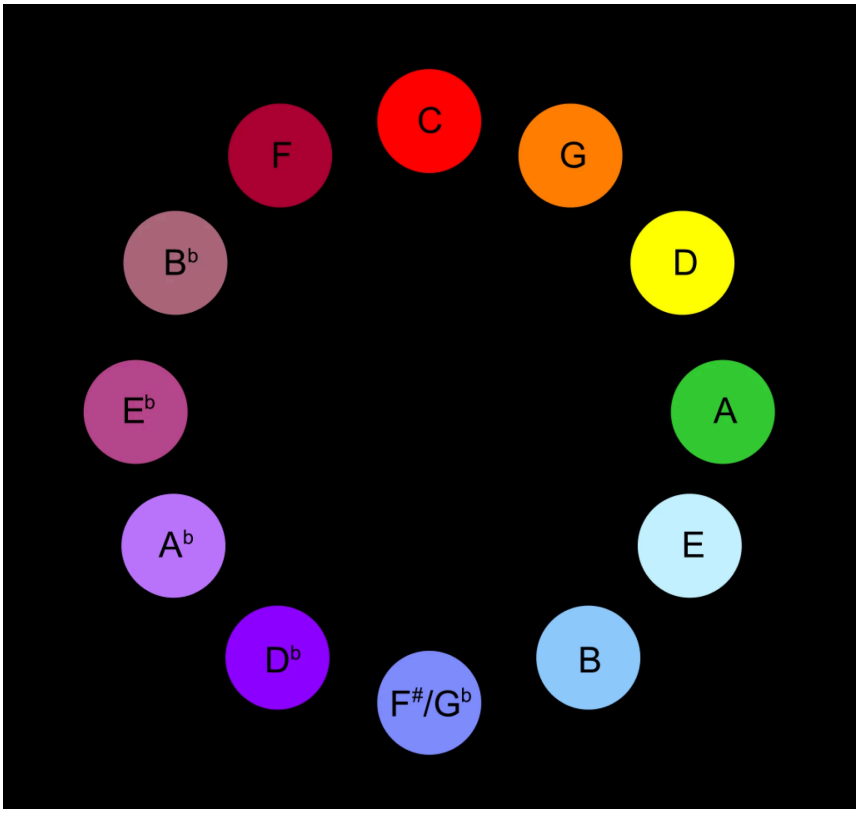
Data and Pitch

Pitch plays a crucial role in shaping both melody and harmony in music, as it defines the notes that will be used to create the musical narrative. In melody, pitch determines the sequence and progression of notes, while in harmony, pitch is responsible for the stacking of notes into chords. There are numerous ways to derive pitches. One approach I explored is determining a key and mode, which provides a specific set of pitches that can be used consistently throughout the piece. By establishing the key, I can select the tonic note, which serves as the central pitch around which the melody revolves. Choosing a mode such as major, minor, or a modal scale like Dorian or Phrygian, further defines the pattern of intervals within the scale, giving the melody and harmony a distinct character. This framework ensures that the pitch set is cohesive and harmonically consistent, allowing for smooth harmonization and integration with the rhythmic structure derived from the clave, ultimately creating a piece that is both melodically and rhythmically engaging.

My goal in determining pitch through image data is to identify the key and mode by leveraging the inherent qualities of the image, fostering a more intuitive and exploratory

creative process. This approach not only anchors the music in a visual context but also encourages a deeper connection between the visual and auditory elements, allowing the creative process to unfold organically. By letting the image data guide the selection of key and mode, the resulting music becomes a reflection of the image's unique attributes, promoting a dynamic and responsive interaction between the visual and musical components.

Synesthesia, the phenomenon where stimulation of one sensory pathway leads to automatic, involuntary experiences in a second sensory pathway, offers a unique perspective on choosing a root note for a home key.[8] For individuals with synesthesia, specific musical notes or keys may evoke distinct colours, shapes, or other sensory experiences. A correlation between this cross-sensory perception and a digital image can be formed by mapping a colour to a pitch. Alexander Scriabin, a late 19th-century composer, famously devised a system relating colour to the circle of fifths.[9]



[8] Zamm, Anna, Gottfried Schlaug, David M. Eagleman, and Psyche Loui. "Pathways to Seeing Music: Enhanced Structural Connectivity in Colored-Music Synesthesia." NeuroImage, Suppl.C 74, (Jul 01, 2013): 359-366. doi:https://doi-org.libproxy.mta.ca/10.1016/j.neuroimage.2013.02.024. http://search.proquest.com.libproxy.mta.ca/scholarly-journals/pathways-seeing-music-enhanced-structural/docview/1668112701/se-2.

[9] Eric William Barnum. “Audition Colorée (Synesthesia).” Epiphany and Catharsis (blog), April 20 2011.

Figure 3.1 Scrabin's colour wheel for artificial synesthesia.

To determine the root note of a scale based on the colours in a digital image, you can start by averaging the RGB values of every pixel within the image. This averaging process produces a single RGB colour value that represents the overall hue of the image. By using Scriabin's colour wheel, which associates specific colours with musical notes based on his interpretation of synesthesia, you can map this averaged colour to a corresponding root note.

To find the closest match between the averaged colour and a note on Scriabin's wheel, you can use a method that involves plotting the RGB values of both the averaged colour and the colours on Scriabin's wheel onto a 3D graph, where the x, y, and z axes represent the red, green, and blue components, respectively. The vector distance formula is then used to calculate the distance between the averaged RGB value and each colour on the wheel. By iterating over all possible colours on Scriabin's wheel, the colour with the shortest vector distance to the averaged RGB value is identified as the closest match, and the corresponding note is selected as the root note of the key.

Here's the basic pseudocode for this process, assuming `colourWheel` is a dictionary containing the twelve musical notes and their corresponding RGB values, and `avgRGB` is the computed average RGB value of the image:

```
Function colourDistance(colour1, colour2)
    // Calculate the difference in each colour component
    rDiff = colour1[1] - colour2[1]
    gDiff = colour1[2] - colour2[2]
    bDiff = colour1[3] - colour2[3]

    // Compute and return the Euclidean distance
    distance = sqrt(rDiff^2 + gDiff^2 + bDiff^2)
    Return distance
End Function

Function findClosestKey(colourWheel, targetColour)
    closestKey = null

    // Iterate over each key in the colourWheel
    For each key in colourWheel
        // Compare distances to find the closest colour
        If closestKey is null OR
colourDistance(colourWheel[key], targetColour) <
colourDistance(colourWheel[closestKey], targetColour)
            closestKey = key
        End If
    End For
```

```
    Return closestKey
End Function
```

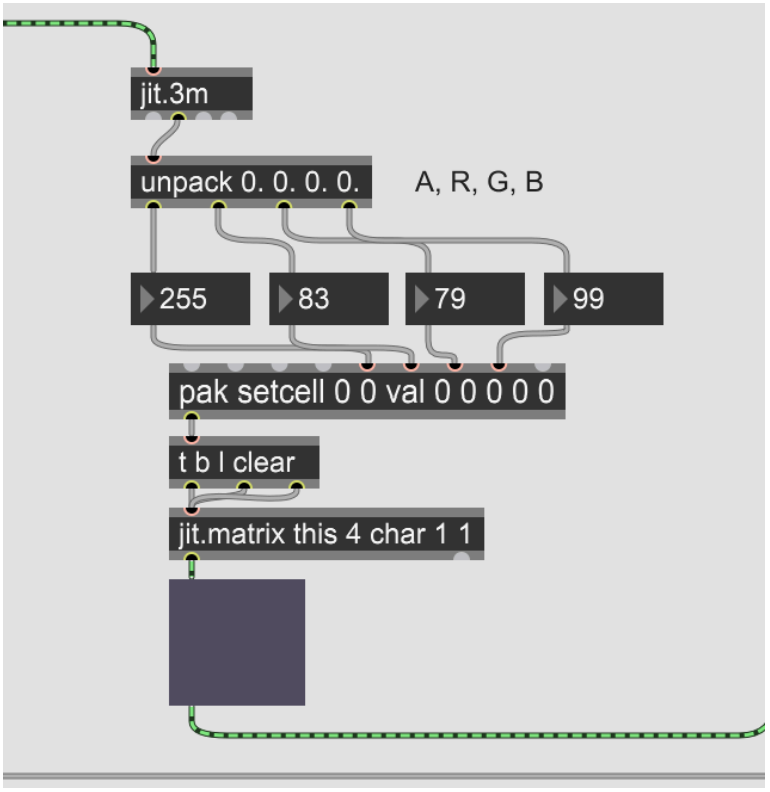


Figure 3.2 MaxPatch finding the average hue of an image. This patch gets the average RGB values from an image and displays the resulting colour onto a pixel. These RGB values are then sent as input to the script above to get the key based on Scrabin's colour wheel.

The average brightness of an image can be intriguingly related to musical modes, which can evoke various emotional qualities in music. Musical modes, such as Ionian (major) and Aeolian (natural minor), each have distinct characteristics that can be perceived as brighter or darker. Brighter modes, like the Ionian mode, often correspond to a more uplifting and vibrant emotional quality, similar to how an image with high average brightness might feel more energetic and positive. Conversely, darker modes, such as the Aeolian mode, evoke a more sombre and introspective mood, like an image with low average brightness, which can appear more subdued or melancholic. Using the same formula for evaluating the average light intensity seen when computing the rhythmic key, this value ranging from 0-255 can be scaled down to be between 0-6 to get the mode from a list ranging from darkest to brightest: [Dorian, Phrygian, Lydian, Mixolydian, Aeolian, Ionian, Locrian]

Once the key and mode are established, various techniques can be employed to generate pitches for the melody or harmony. One approach could involve random or weighted selection from the scale, where pitches are chosen either randomly or with certain notes (such as the tonic or dominant) having a higher probability of being selected. This allows for a

balance between predictability and variation, ensuring that the chosen pitches are both cohesive and engaging. By integrating these pitch-generation techniques with the rhythmic clave derived from the image, the resulting rhythms, melodies, and harmonies are dynamic yet consistent, as the frameworks are intricately connected with the visual data. This creates a sense of unity between the music and the image, ensuring that the auditory elements are deeply reflective of the visual data.

Data and Timbre

Timbre plays a crucial role in shaping the texture, mood, and atmosphere of music. Audio effects are a powerful tool for shaping timbre. They add complexity to a sound's texture and harmonic content beyond the choice of instrumentation and inflection. The origins of audio effects can be found in early acoustic techniques where the physical environment was altered to change the sound.

My goal with timbre was to process the image data to be used as input to preexisting audio effects, enabling a unique method of controlling these effects through visual manipulation. By altering aspects of the image, such as its resolution, brightness, and colour, I can directly influence parameters like reverb, delay, or distortion in the audio. For instance, increasing the brightness of an image might correspond to a higher level of reverb, while adjusting the image's resolution could alter the delay time. Similarly, changes in the colours of the image could modify the tonal quality or intensity of the effects. This approach not only offers a visually intuitive way to interact with and control sound but also creates a dynamic connection between the visual and auditory elements, allowing for a more immersive and exploratory creative process.

One of the earliest known effects is reverb, which naturally occurs when sound reflects off surfaces in a space, creating a sense of depth and atmosphere. Musicians and composers in ancient and medieval times took advantage of this phenomenon by performing in spaces like cathedrals and caves, where the long reverberation time added a unique, ethereal quality to their music.[10]

The development of digital technology in the 1980s revolutionized the world of audio effects. Digital signal processing (DSP) allowed the creation of effects that were not possible with analog equipment. The process begins by converting an analog audio signal into a digital format, where the sound wave is represented by a series of discrete numerical values. DSP algorithms then process these numerical values to achieve the desired effect, such as reverb, delay, distortion, or equalization. For instance, in a reverb effect, the algorithm simulates the reflections of sound waves in a physical space by adding delayed versions of

---

[10] Thomas Wilmering, David Moffat, Alessia Milo, and Mark B. Sandler, “A History of Audio Effects,” *Applied Sciences* 10, no. 3 (2020): 791, https://doi.org/10.3390/app10030791.

the original signal. The processed digital signal is then converted back to analog for playback through speakers or headphones.

These DSP algorithms, although complex, rely on precise and well-structured data to function effectively. It is crucial to process the image data in a way that aligns with the specific parameters required by the audio effects. For instance, certain parameters might only accept simple integer values, requiring the image data to be scaled or quantized appropriately to fit within this scope. Additionally, transforming the image data into a more suitable format, such as a 2D graph, can help map visual characteristics directly to sound manipulation, like plotting brightness against amplitude or colour values against frequency. By carefully processing the data to match the requirements of these parameters, we can ensure that the audio effects respond accurately and meaningfully to the visual input, resulting in a seamless and effective integration of visual and auditory elements.

With this in mind, I focused my attention on exploring the different ways the image data could be transformed into parameter values, specifically targeting integers and 2D graphing data. My goal was to understand how to best convert the visual information into formats that could be effectively used by preexisting audio effects.

There are several ways to derive a single integer value from an image, each capturing different aspects of its visual content. A common method is averaging the RGB values of all the pixels, producing a single value that represents the primary colour tones of the image. Moreover, we've seen how converting the image to grayscale and calculating the average luminance provides an integer that reflects the image's overall brightness. The resolution of the image, defined by its width and height, can also be used as a straightforward integer value that indicates the total number of pixels.

I developed a Max/MSP patch that integrates a preexisting reverb effect with five adjustable parameters, each of which is controlled by manipulating different aspects of an image. For the dry/wet mix, I implemented a slider that adjusts the dimensions of the image, effectively pixelating it. As the image becomes more pixelated, the mix value increases, blending more of the wet (reverb) signal with the dry (original) signal.

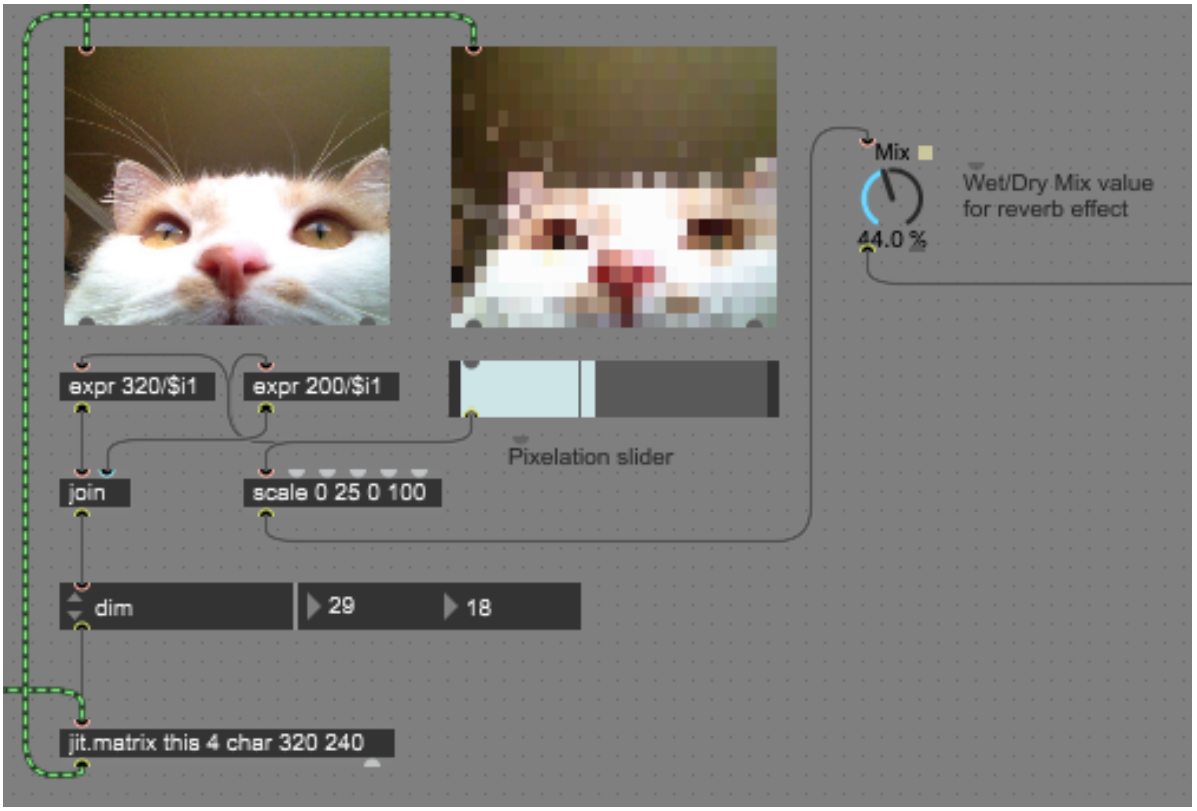


Figure 4.1 Image pixelation to control wet/dry mix. This section of the patch features a slider ranging from 0 to 25 that pixelates an image of my cat, adjusting the resolution (320x240) accordingly. The pixelation value is displayed, and the wet/dry mix increases in increments of 4, corresponding to the level of pixelation.

The decay parameter is determined by applying a binary operation to the image. A binary operation evaluates each pixel and returns either true or false based on whether it meets a specified criterion. In this context, a new image is generated where pixels that satisfy the criterion are rendered as white (true), and those that do not are rendered as black (false). In this case, I checked whether the brightness of each pixel exceeds 50%. The number of pixels that pass this check is counted and used as the parameter value. This count directly influences the reverb's decay time, with a higher number of bright pixels resulting in a longer decay.

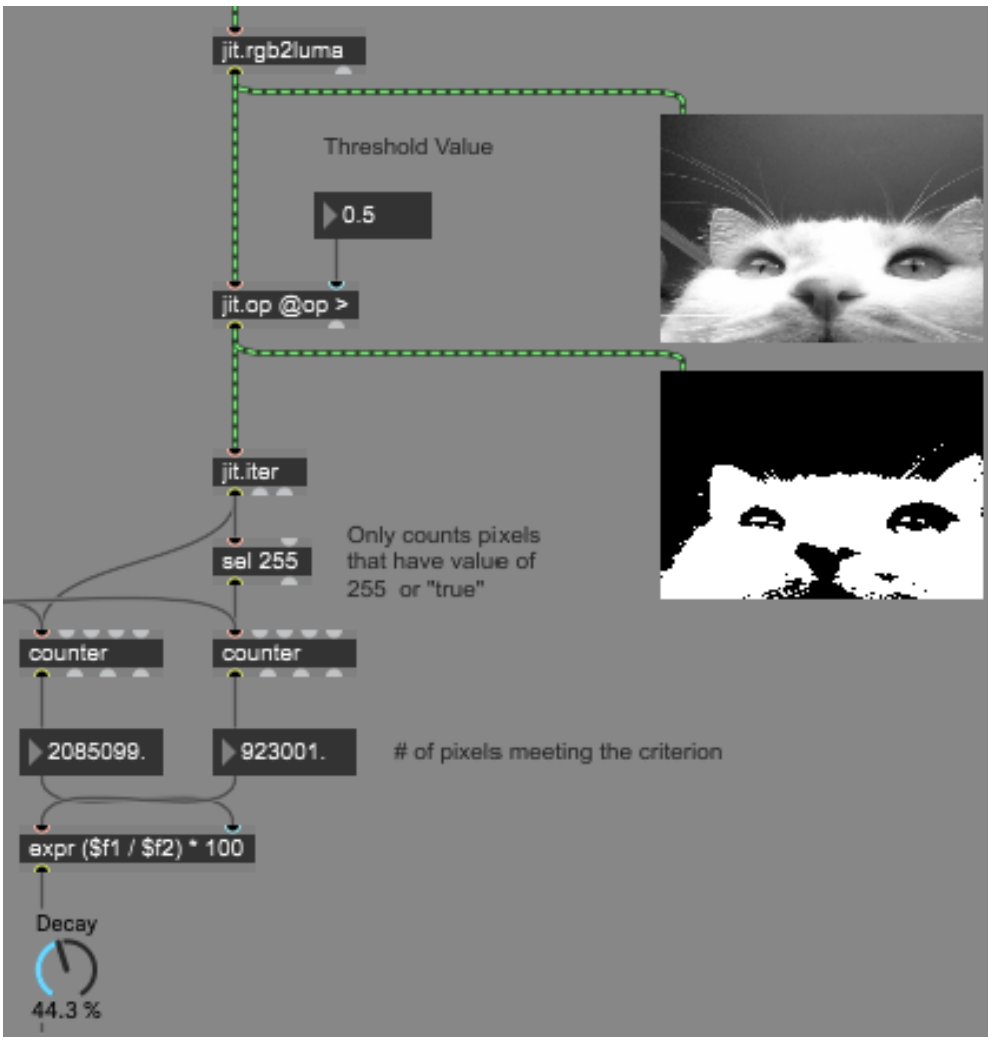


Figure 4.2 Binary operation to get the decay value. This section of the reverb patch first begins by converting the image into grayscale. Then it does a binary operation on the image checking if the brightness value of a pixel is above 0.5 (Controlled by the threshold value). This new binary image then counts the total number of pixels and the number of pixels that met the criterion. The % of pixels that meet the criterion is used as the decay value for the reverb effect.

The size of the reverb is controlled by the average luminance of the image, where a higher luminance results in a larger perceived space. For the diffuse parameter, I randomly select a pixel from the image and use its red value to adjust the diffusion, adding variability based on the chosen pixel's colour. Lastly, the damp parameter is influenced by the average red value of the entire image, providing a contrast to the random pixel selection used for diffusing. This approach ensures that the reverb effect is not only responsive to the visual content of the image but also creates a dynamic and visually driven audio experience.

The 2D grid-like structure of an image naturally lends itself to graphing, offering unique possibilities for creating sound effects using functions such as an EQ curve. To transform an image into a functional graph, you must first determine the x and y coordinates for plotting points. This is achieved by converting the image into binary based on specific criteria using image processing techniques. Once the image is processed, you can iterate through the binary data to graph the corresponding points. It is crucial to ensure that no two points share the same x value, which can be done by adding a small offset to each pixel in a column (Example: 1.0, 1.1, 1.2, … 2.0, 2.1, and so on). This method allows for the creation of intricate, visually driven sound effects that change dynamically based on the plotted data.

So far, my focus has been primarily on electroacoustic applications, where the sound is shaped live in response to real-time image data. This approach allows for dynamic interaction between visual and auditory elements during a performance. However, by plotting image data as a function of time, we can extend this concept to apply effects to prerecorded audio clips, allowing the effect to evolve throughout the sample. By mapping the visual data onto a timeline, the parameters of an audio effect such as reverb or EQ can change progressively as the sample plays. This method adds a new layer of control and creativity, made possible by the inherently grid-like structure of pixels within image data.

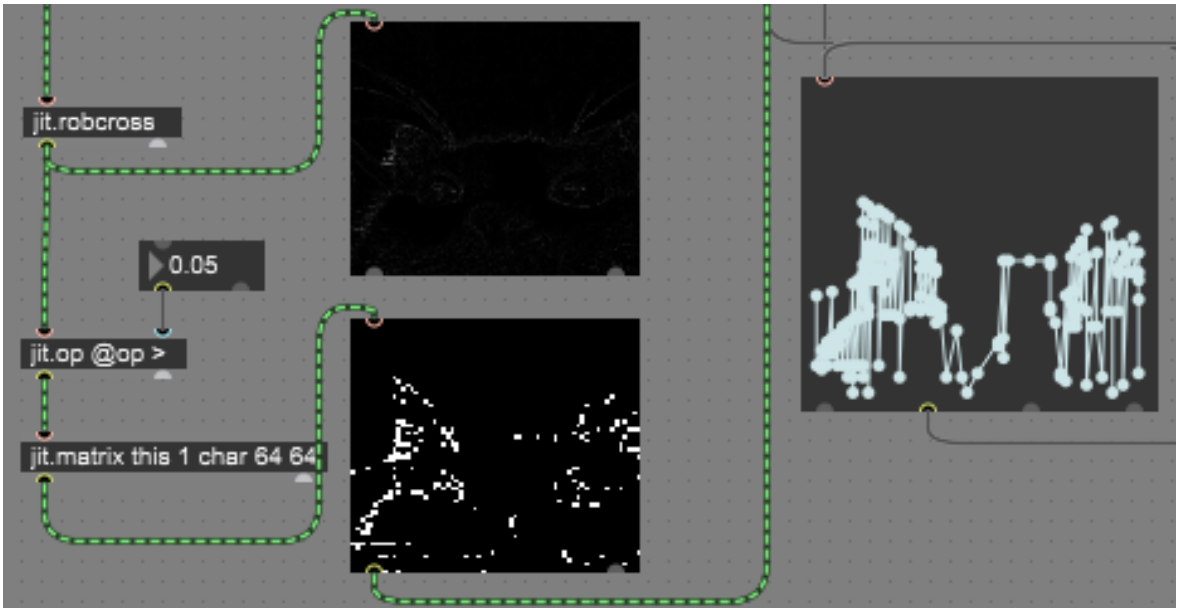


Figure 4.3 Generating a function from edge detection algorithm. A MaxPatch that processes an image of my cat into a 2D graph. It starts by using the "jit.robcross" object, a built-in edge detection algorithm in Max, followed by applying a binary operation to filter prominent edges based on a brightness threshold of 5%. The image is then resized to 64x64 to reduce the number of points on the resulting graph. These points are plotted and traced into a function. Note that the x-values of pixels in the same column are incremented by 1/64 for each new point, though this step is not shown.

Using the function generated from the figure above, I created a patch that shifts the frequency of a sample dynamically. By setting the function's domain to correspond with the duration of the clip, the patch reads the function's values at each point in time and applies these values to the frequency, effectively altering it throughout the playback. To allow for more dramatic shifts, I set the function's range to span from -1000 to 1000, ensuring significant and varied frequency modulation throughout the sample.

Other Fields of Study

In my pursuit to advance my multidisciplinary skills over the summer, I delved into live coding, microcontrollers, and artificial intelligence (AI). Initially, I considered using live coding to create live electronic music by manipulating image data through real-time code performance. However, I chose to work with MaxMSP instead, as its visual coding environment allowed me to both process and display the image, aligning more closely with my goals. I also explored integrating microcontrollers to introduce a dynamic input stream into the project. Since images are inherently static, I experimented with sensors like accelerometers and gyroscopes to inject analog data, adding movement and variability to the otherwise fixed image.

While these explorations were promising, I became more intrigued by the possibility of manipulating the image itself in real-time to alter its underlying data, effectively transforming a static image into a dynamic environment. Additionally, I investigated the potential of AI in computer vision, specifically its ability to discern subtle differences between visually similar yet conceptually distinct elements, such as a lake and a blue volcano. Despite their similar appearance in raw data, AI could detect unique features, and this capability could be leveraged to generate different musical responses based on the visual distinctions, presenting an exciting step toward blending image recognition with creative sound design.

Future Steps

Reflecting on my summer research, the majority of my time was dedicated to deeply understanding image data and how it can be shaped to serve musical purposes. This process involved not only learning how to extract relevant data from digital images but also transforming that data into a format that could be used as input for various algorithms in electronic music creation. Grasping the intricacies of image data was crucial, as it allowed me to explore how visual elements could influence and control different aspects of sound, from rhythm and pitch to more complex audio effects. This foundational knowledge has been essential in developing innovative ways to bridge the gap between visual and auditory art forms.

While the work I've done this summer did not come together to create a complete composition, it represents an exploration of tools and techniques for composing various musical elements using image data. The experiments and patches I've created serve as building blocks for understanding how visual data can shape sound, offering new possibilities for electronic music composition. Although these initial explorations are just the beginning, they have provided valuable insights into how digital image data can be repurposed to generate rhythmic structures, melodies, and sound effects, laying the groundwork for more comprehensive compositions.

Looking ahead, I anticipate that the work I've done this summer will significantly inform my capstone project. The techniques I've developed will serve as the basis for creating a complete composition that integrates visual data with musical expression. By applying these methods to a full-scale project, I aim to further explore the potential of image-based music creation and produce a cohesive piece that not only challenges conventional notions of music but also showcases the dynamic relationship between sight and sound. This summer's research has equipped me with the tools and understanding necessary to push the boundaries of electronic music, and I'm excited to see how these ideas will evolve in my future work.

Bibliography

Adobe. "Vector Image File Formats: The Most Common Types." Adobe. Accessed May 27, 2024. https://www.adobe.com/ca/creativecloud/file-types/image/vector.html.

Acharya, Tinku, Tsai, Ping-Sing, and Tsai, Ping-Sing. JPEG2000 Standard for Image Compression: Concepts, Algorithms and VLSI Architectures. Newark: John Wiley & Sons, Incorporated, 2004. Accessed May 14, 2024. ProQuest Ebook Central

Barnum, Eric William. “Audition Colorée (Synesthesia).” Epiphany and Catharsis (blog), April 20, 2011. https://ericwilliambarnum.wordpress.com/2011/04/20/audition-coloree-synesthesia/.

Cycling '74 Documentation. "Tutorial 5: Matrix Data and Planes." Accessed May 26, 2024. https://docs.cycling74.com/max8/tutorials/jitterchapter05.

“JPEG - Image File Format.” JPEG - Image File Format. Accessed May 23, 2024. https://docs.fileformat.com/image/jpeg/.

Mahmood, Riyad Mitieb.“Color Spaces Analysis for Luminance & Chrominance Signals AS NTSC-TV System”. 2011. Journal of Engineering and Sustainable Development 15 (4): 123-34. https://jeasd.uomustansiriyah.edu.iq/index.php/jeasd/article/view/1312.

Moffat, Alistair. “Huffman Coding.” *ACM Computing Surveys*, vol. 52, no. 4, 2020, pp. 1–35, https://doi.org/10.1145/3342555.

Rowlands, Andy. Physics of Digital Photography (Second Edition), Institute of Physics Publishing, 2020. ProQuest Ebook Central, https://ebookcentral.proquest.com/lib/mta-ebooks/detail.action?docID=31252800.

Riley, Sean. “Digital Images - Computerphile.” YouTube, uploaded by Computerphile, 15 February 2015, https://www.youtube.com/watch?v=06OHflWNCOE&ab_channel=Computerphile.

Whitley, Matthew. “Algorithmic Music Generation: Solving the right problem.” Presented at the Australasian Computer Music Conference (ACMC) 2020, July 6, 2020. https://australasian-computer-music-association.github.io/acmc2020/poster_81.html.

Wilmering, Thomas, David Moffat, Alessia Milo, and Mark B. Sandler. "A History of Audio Effects." *Applied Sciences* 10, no. 3 (2020): 791. doi https://doi.org/10.3390/app10030791. http://search.proquest.com.libproxy.mta.ca/scholarly-journals/history-audio-effects/docview/2533928979/se-2.

Zamm, Anna, Gottfried Schlaug, David M. Eagleman, and Psyche Loui. 2013. "Pathways to Seeing Music: Enhanced Structural Connectivity in Colored-Music Synesthesia." NeuroImage, Suppl.C 74 (Jul 01): 359-366. doi:https://doi-org.libproxy.mta.ca/10.1016/j.neuroimage.2013.02.024.