12/12/2016

Image Audification

Submitted to: Dr. M. Mostafa

Ziad Osama Abdel Ghany
900130315

An approach for image sonification

1. Introduction:

Image sonification can be defined as translating the image from the visual domain to the audio domain. This can be used in various fields, starting from applications for the visually impaired [7] to a synthesizer for audiovisual artists [3] in their live performances. Image sonification follows audification of data where we try to provide another method for visualizing a digital signal which in my case is an image. I am trying to utilize the visual information for a musical purpose. This process is being researched in CCRMA (Center for Computer Research in Music and Acoustics) at Stanford University.

2. Target:

In my application, I am targeting artists with no musical background who want to have a piece of music for their art piece. Artists usually have to go through getting the copyrights of an already pre-recorded song which adds a cost to the operation and is considered impractical. The other option is studying music theory to create music pieces, which adds an overhead of being a music expert to compose and also considered impractical. So, if they have a tool which can create the appropriate musical piece for their artwork, it will make the process much easier, reduce costs and no copyright infringements will be faced. I was inspired by this quote by M. V. Mathews "A computer can be programmed to play "instrumental" music, to aid the composer, or to compose unaided." [8]

3. The Tools used:

My coding environment is a tool called processing. Processing is a widely used software within researchers, designers and artists. It's used for prototyping in visual arts because it provides ease of usage in linking audio and visuals. Another major importance, is open source, processing is developed from the community to its community. Any artist can use it and change the code to suit his needs and it has the ability to export the code onto different platforms. The library used is an audio library called minim [6]. Minim is a powerful open source audio library that uses the JavaSound API [5], Tritonus [1] and MP3SPI [2]. It provides ease of usage for developers in processing. Minim is very famous with processing users to the extent, it was integrated in newer processing versions.

4. The Approach:

My approach is based on color segmentation in the color space of HSB (hue, saturation, brightness). HSB provides easier and more accurate color ranges than the normal RGB (red, green, blue) color space. Each musical piece follows a scale with a tempo, the musical scales are assigned to the different ranges of hue, the saturation determines the octave of the note and the brightness plays a chord progression. A normal musical scale consists of 7-8 notes and since we have roughly 8 colors in the hue range; red, orange, yellow, green, cyan, blue, purple and pink. The translation process is based on getting a random pixel from the input image. Based on the hue of this pixel, we assign it to a note from the scale. Now I have a basic knowledge of the frequency of the note that is going to be played. Furthermore, I use the saturation to determine the frequency range (octave) that the note is going to be played in. The lower the octave, the lower the frequency. After that, depending on the brightness of the pixel, I either play a chord or play a note. A chord is the combination of 3 or more notes played together.

5. Documentation:

The declaration in the beginning of the code is for setting up the path for loading the image, the musical scales that can be used, the color ranges in hue are commented after the notes. Then the minim library declarations for playing the notes and saving recordings. We reach the setup part, where we initialize all our data, first the window size (sketch size), and the color space we are going to use, initializing the minim library and the audio output. Audio ouput connects to the output of a computer's sound card and we specify Mono output or stereo, we can also specify the buffer size, the sample rate and bit depth. Then we set the tempo and choose one of the random scales that were declared previously and we create a recorder for recording the sound. In the draw part we call the function separate which performs the segmentation and plays the note at the kth second. The program keeps running until the user quits and it records the output played by the sketch. The last thing by pressing 'r' we save the output in myrecording.wav.

The instrument class uses the instrument interface in minim, it allows the user to create his own instrument. I used the code from their reference [6] and modified it a bit by adding an ADSR (Attack, Delay, Sustain, Release) envelope. ADSR is a UGen (unit generator), a ugen is the basic formal unit in computer music programming languages, they are the building blocks for designing synthesis and signal processing algorithms in software. “The computer sounds are described in terms of the waveshapes produced by the unit generators in the instrument units” [8]

6. Future Improvements:

From the future improvements that can be added, matching scales according to the HSB of the image where we can play sad pieces if it’s a dark image or cheerful pieces, if it’s a bright image. Another improvement is developing a mode for imitating instruments using the instrument class.

7. Notes:

The same approach can be used in security where we can send packets of audio signals instead of sending packets of image and on the other end in a receiver application will perform an FFT on the audio signal and we will try to reconstruct the pixel based on the frequency since each octave has a frequency range. On trying to implement this, a problem was faced due to large size of images and overhead of recording each pixel of an image, so it was discarded.

References

[1] http://www.tritonus.org/

[2] http://www.javazoom.net/mp3spi/mp3spi.html

[3] http://www.imagesonification.com/

[4] https://ccrma.stanford.edu/overview/research.html

[5] https://docs.oracle.com/javase/tutorial/sound/

[6] http://code.compartmental.net/minim/

[7] Yoshida, Tsuba, Kris M. Kitani, Hideki Koike, Serge Belongie, and Kevin Scheli. "EdgeSonic: Image Feature Sonification for the Visually Impaired." *ACM Digital Library*. Proceedings of the 2nd Augmented Human International Conferenc, 13 Mar. 2011. Web. 11 Dec. 2016.

[8] Mathews, M. V. "The Digital Computer as a Musical Instrument*." Proceedings of the American Mathematical Society* 26.4 (1970): 695-98. Web. 11 Dec. 2016.