REAL TIME IMAGE PROCESSING BASED ON AUDIO CONTENT

Jay Biernat

University of Rochester

ABSTRACT

This paper explores the connection between audio and visual media by using audio features to control image processing in real time. Anyone involved in the audio-visual industry is aware of the important connection between what we see and what we hear in media. Our goal was to explore ways of mapping audio content to the processing of a static image to cultivate strong connections between the audio being played and the processed image that we observe. Several image processing techniques were explored including color tinting, color contrast, and blurring using a Gaussian filter. The real time processing was implemented in MATLAB using system objects from the DSP and Computer Vision toolboxes. Our implementation was tested using several pieces of classical music, and we believe that some level of success was achieved with much room for improvement and further exploration.

1. INTRODUCTION

Vision and hearing are two important senses that shape how we perceive the world. Often, what we see and what we hear are strongly connected to one another. In media that has both audio and visual elements, creators work hard to make sure that there is a connection between sound and images, and it is most often the audio that is molded to fit the visual element. This paper explores how the opposite might be achieved by manipulating a visual image based on audio features. Although some clear connections between audio and visual representations exist (e.g. the audio's waveform or spectrogram), these are more data-driven representations, and we sought to explore a more aesthetic connection between audio features and the visual representation of them.

To achieve this, we used image processing techniques in real time to manipulate an image based on audio content that is being played. The goal was to manipulate the image in a way that visually connected with what one was hearing in the audio at that moment. Several methods of image manipulation were used: color tinting, contrast control, and blurring. Each method of manipulation is tied to an extracted audio feature: spectral band energy, spectral flux, and signal energy, respectively, which control the amount of processing done on the image in a single frame.

This paper is organized as follows: In section 2, we give an overview of the process of using audio content to manipulate an image in real time. In section 3, we examine in detail how each audio feature is mapped to a specific image processing technique. Finally, section 4 will discuss the achieved results as well as improvements that can be made in future work.

2. PROCESS OVERVIEW

Currently, this processing is being implemented in MATLAB using the AudioFileReader, AudioPlayer, and VideoPlayer system objects from the DSP and Computer Vision toolboxes. An audio file and a JPEG or PNG image are specified, and then feature extraction and image processing takes place in real time. This process is summarized in Fig 1.

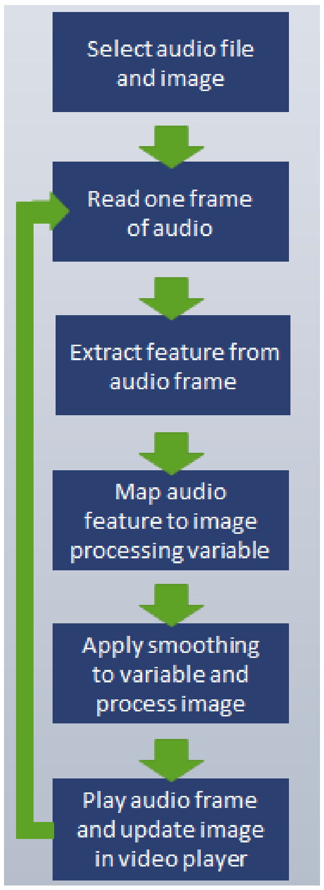
Select audio file and image
Read one frame of audio
Extract feature from audio frame
Map audio feature to image processing variable
Apply smoothing to variable and process image
Play audio frame and update image in video player

Fig 1. An overview of the image manipulation process based on audio content.

The AudioFileReader object first reads one frame of audio (~20ms long) from the specified audio file. The relevant audio features are then extracted from this frame. Each feature gets mapped to a variable specific to its assigned image processing technique. For example: the energy in each spectral band is mapped to a scaling factor used in applying color tinting to the image. After a variable is mapped from an audio feature, a smoothing function must be applied to it. Eq. 1 shows the smoothing function used in our implementation.

$$
\begin{aligned}
& if\ \alpha_n \geq \alpha_{n-1} \\
& \hat{\alpha} = (1-s) * a_{n-1} + s * |\alpha_n| \\
& else \\
& \hat{\alpha} = (1-s) * a_{n-1} - s * |\alpha_n|
\end{aligned} \tag{1}
$$

$\hat{\alpha}$ represents the smoothed variable calculated using the values of the variable from the current and previous frames, and a scaling constant s. This is necessary to prevent rapid fluctuations in the processed images which produce an unpleasant "jitter" effect in successive images.

Once the necessary variables are calculated and smoothed, the image is processed using the appropriate image processing technique. The audio frame is then output to the AudioPlayer, and the processed image is output to the VideoPlayer. The next audio frame is read, and the process is repeated until the end of the audio file is reached.

3. TYPES OF IMAGE PROCESSING

Three types of mappings of audio features to image processing techniques are explored in this paper: spectral band energy to color tinting; spectral flux to image contrast; and signal energy to blurring (using a Gaussian filter). Each of these mappings is explained in detail below. Although each is examined individually, it is possible to combine more than one type of image processing in a single implementation.

3.1. Color tinting

In this paper, we apply a very simplistic form of color tinting to the image. In an RGB image, each pixel has three intensity values associated with it: one intensity value for each of three colors red, green, and blue. To manipulate the color tint of an image, we define three variables α, β, and γ that scale the intensity value of each of these colors, often producing a noticeable red, green, or blue tint based on the values of the three scaling factors.

The scaling factors α, β, and γ are calculated based on the amount of spectral energy within three frequency bands of an audio frame. These frequency bands are chosen to correspond roughly with the generally accepted ranges of the notion of low, mid, and high frequencies [1]. In this paper, the ranges of these frequency bands are defined as: 20-500Hz (low), 500-1500Hz (mid), 1.5-20kHz (high).

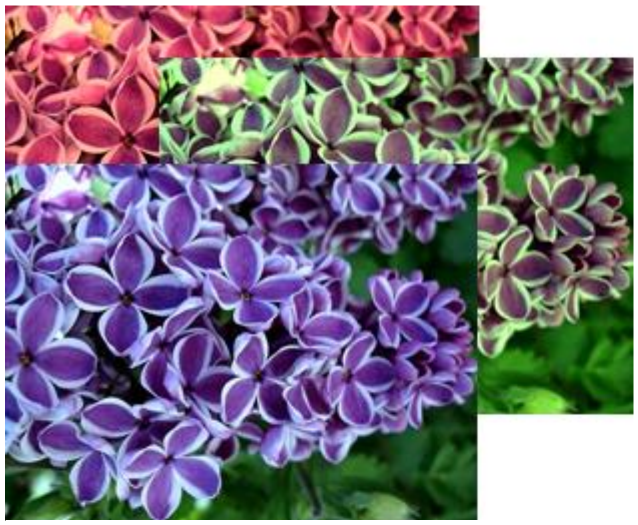

Fig 2. Examples of images tinted with large amounts of red, green, and blue from corresponding large scaling factors α, β, and γ.

Although the boundary between mid and high frequency bands is more commonly referenced around 2kHz, a lower boundary was chosen in this case to compensate for the generally lower-energy high frequencies (compared to the low and mid-range frequencies).

To calculate these scaling factors, we first calculate the amount of energy present in each frequency band

$$E = \sum_{f=f_0}^{F} X^2(f) \tag{2}$$

where $f_0$and F represent the lowest and highest frequencies of a given frequency band. The scaling factor is then the ratio of the spectral energy in a given band to the total spectral energy in all bands:

$$\alpha = \frac{E_{band}}{\sum_{all\ bands} E_{band}} \tag{3}$$

After smoothing, these scaling factors are used to adjust the intensity of each of the RGB values in a JPEG or PNG image.

$$J(i,j,n) = I(i,j,n) * (1 + \hat{\alpha}) \tag{4}$$

where *i* and *j* are the indices of the pixels in an image and *n* takes values 1 to 3, representing the RGB intensity layers in the image. α, β, or γ will be used depending on the value of *n*.

Mapping spectral energy to color tinting in this way will result in a processed image having a large amount of tinting that corresponds to a frequency band with a large amount of energy present. For example, a frame of audio with a large amount of energy in its low frequency band will produce an

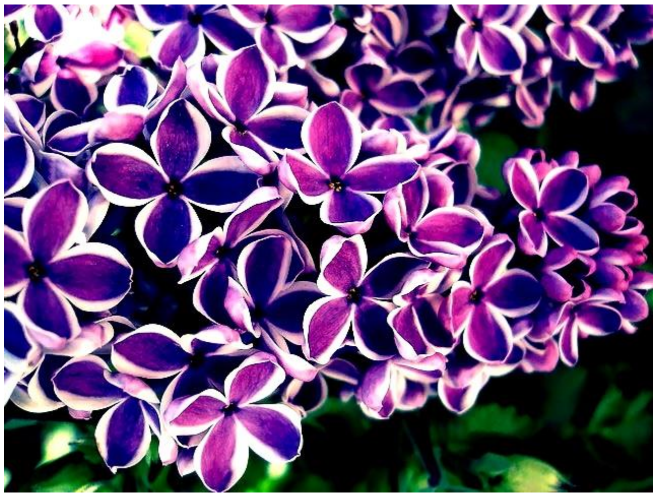

Fig 3. An example of a processed image with a large amount of contrast due to high spectral flux between successive frames.

image with a large amount of red tint to it. Fig 2 shows examples of what an image may look like with large amounts of tinting applied to it.

3.2. Contrast control

To control the contrast of an image, the MATLAB function imadjust() was used to control the contrast limits of the RGB intensities in an image. Contrast limits are specified using a 2x3 array where the first row of 3 elements species the lower contrast limit of each RGB intensity, and the second row of 3 elements specifies the upper contrast limit of each intensity [2]. The values of these limits can take values between zero and one, inclusive. When the range between the upper and lower contrast limits for color intensity is narrow, that color will have high contrast. When the range between upper and lower limits for color intensity is wide, that color will have little contrast. If the lower and upper limits are zero and one, respectively, no contrast is applied.

In order to control the contrast in an image, the spectral flux between frames is used. Spectral flux is a measure of the amount of change in a signal's frequency spectrum [3]. This was chosen with the intention that a frame that has a large amount of spectral flux (such as a frame where a note onset occurs) will produce an image that has a large amount of contrast. By watching the image, we should be able to see visually where note onsets are occurring based on the contrast of the image.

Because three pairs of contrast limits are needed, spectral flux was calculated for each of the same three frequency bands that were used in determining spectral energy for color tinting. Spectral flux was mapped to the upper contrast limit for each RGB intensity using the following formula:

$$lim_{high} = \frac{\rho}{\sum_{f=f_0}^{F}[X_n(f) - X_{n-1}(f)]^2} + 0.5 \tag{5}$$

In Eq 5, ρ represents a scaling factor used to adjust the bound of the upper contrast limit. This is adjusted for each individual audio file and is needed to achieve optimal output. Without adjusting ρ, we may see very little contrast in the processed images or too much contrast depending on the average amount of spectral flux in the audio file. The constant 0.5 is added to ensure that the upper contrast limit is never less than 0.5, which would cause the image to be inverted (like in a photographic negative).

Once the upper contrast limit of an RGB intensity has been calculated, we set the lower limit to be

$$lim_{low} = 1 - lim_{high} \tag{6}$$

After these upper and lower contrast limits for the three RGB intensities are found, they can be used in the imadjust() function to control the color contrast in the image before it is output to the video player. Fig 3 shows an example of an image with narrow contrast limits (and therefore a large amount of contrast) due to large spectral flux between frames.

3.3. Blurring (Gaussian filtering)

A third mapping of an audio feature to an image processing technique explored is using signal energy to determine the amount of blurring applied to an image using a Gaussian filter. The signal energy is mapped to the standard deviation of the smoothing kernel of the Gaussian filter (σ), which can be specified as a parameter in the MATLAB function imgaussfilt() [4]. The mapping for this is simple: the signal energy in the frame is simply scaled by a constant ρ

$$\sigma = \sum_{n=1}^{N}[x^2(n)] * \rho \tag{7}$$

Just as with mapping spectral flux to contrast limits, the scaling constant ρ is user determined for each audio file to achieve optimal output.

The idea here is that a large amount of signal energy will produce a large amount of blurring in the output image, which is most clearly seen in audio that has a large dynamic range. Fig 4 shows an example of an image with a large amount of blurring from an audio frame with a large amount of signal energy.

5. RESULTS AND FUTURE WORK

Each type of image processing was tested using several audio files, most of which were of classical ensemble or solo piano music. Classical music produced image processing that was much more aesthetically pleasing and

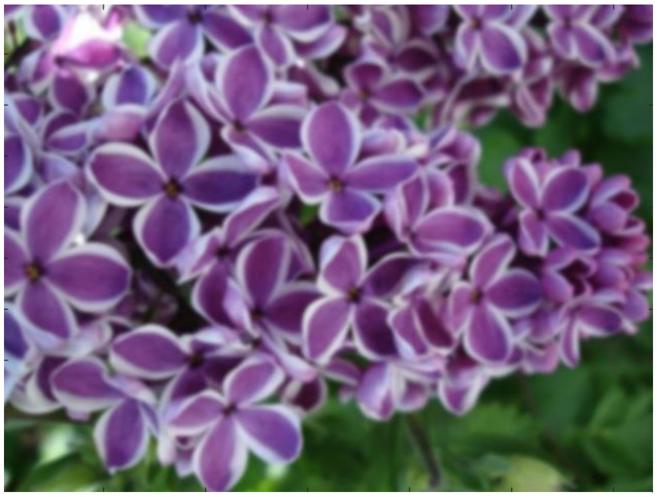

Fig 4. An example of a processed image with a large amount of blurring due to a large amount of signal energy in the audio frame.

less jittery compared to modern pop or rock music. We believe that this is because of the lack of vocals and percussive instruments in the classical music we tested with.

Although the "success" of this research is highly subjective, we will summarize what we felt worked well and what still needs improvement. Of the three audio features explored, the feature that had the strongest correlation visually with the processed image was the spectral band energy. We felt that the color tinting had a consistent and intuitive correlation with the audio that remained true across the several tested audio files. The other two audio features (spectral flux and signal energy) produced images that were less intuitively correlated with the audio.

The mapping of the spectral and energy to color tinting was also a more "successful" implementation because the mapping does not require any adjustment between audio files. The spectral flux and signal energy, however, require adjustment to their scaling constants (ρ) for each audio file to achieve optimal processing. In future work, we would like come up with a way to dynamically update ρ based on the input audio rather than relying on the user to adjust it.

In this paper, we explored only three audio features and image processing techniques, but of course there are many more features, techniques, and combinations thereof that can be explored. To find the "best" audio and visual features to use, it will be useful to look more into psychoacoustics to find which audio features are the most relevant to how we perceive sound – not just the pitch and loudness, but timbre, mood, and more. This will allow us to choose image processing techniques that will (hopefully) produce images that we feel have a very strong and meaningful correlation to what we hear in audio.

6. REFERENCES

[1] R. Dennis. (2000, April). *Equalization by the Octave.* [Online]. Available: http://www.recordingeq.com/EQ/req0400/OctaveEQ.htm

[2] *Adjust image intensity values or colormap*. MATLAB Documentation. [Online]. Available: http://www.mathworks.com/help/images/ref/imadjust.html

[3] P. Herrara-Boyer et al, "Automatic Classification of Pitched Musical Instruments," in Signal Processing Methods for Music Transcription, A. Klapuri and M. Davy, Eds. New York, NY: Springer (Science + Business Media LLC), 2006, pp.163-200.

[4] *2-D Gaussian filtering of images.* MATLAB Documentation. [Online]. Available: http://www.mathworks.com/help/images/ref/imgaussfilt.html