AUTOMATED TIME-FREQUENCY DOMAIN AUDIO CROSSFADES USING GRAPH CUTS

Kyle Robinson
University of Waterloo
kyle.robinson@uwaterloo.ca

Dan Brown
University of Waterloo
dan.brown@uwaterloo.ca

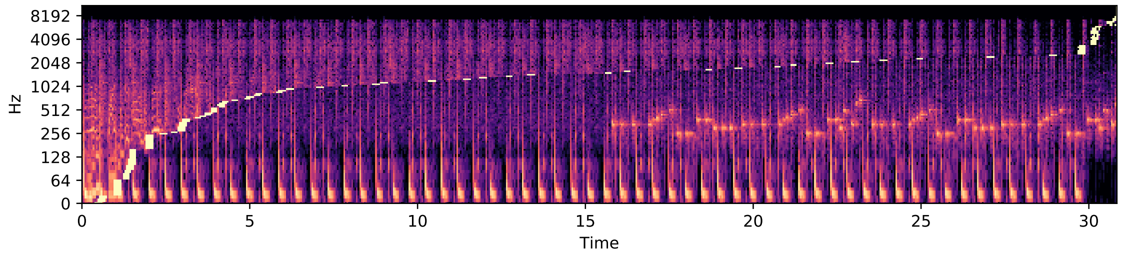

Figure 1: This spectrogram shows overlapped segments of two music tracks after being combined and reconstructed along a per-frequency seam (bright yellow). The tracks were beat and tempo matched, then overlapped by 64 beats.

EXTENDED ABSTRACT

The problem of transitioning smoothly from one audio clip to another arises in many music consumption scenarios; especially as music consumption has moved from professionally curated and live-streamed radios to personal playback devices and services. Classically, transitioning from one song to another has been reliant on either pre-mixed transitions on recorded digital or physical media, hardware or software crossfading on the playback device, or professional transitions by a host or disk jockey (DJ). While options for software crossfading are ubiquitous on music streaming platforms and media players alike, these transitions pale in quality when compared to those manually applied by an audio engineer or DJ who can harmonically and rhythmically align tracks—and importantly—manually apply equalizer (EQ) filters during transitions. The application of EQ filters specifically allow for different transitions in different audio spectrums. For example, the bass register of one track can be made to replace the bass register of another track before transitioning the higher frequencies. Typically the task of deciding how and where to apply transitions in the frequency domain has been completed manually using a limited number of EQ filters.

There is much research on creating, sorting, and extending playlists so as to have tracks naturally flow into each other, as well as on determining optimal times to transition between similar tracks [1, 3, 4, 6, 8]. Both of these research areas play a key role in synthesising a human DJ. To our knowledge, however, all of these approaches still rely on classical methods of transitioning tracks using amplitude in the time domain (crossfading).

Through the application of an existing visual texture extension algorithm borrowed from computer vision, we present the first steps toward a new method of automatically transitioning from one audio clip to another by discretizing the frequency spectrum into bins and then finding transition times for each bin. [5].

We begin by phrasing the problem of transitioning from one song to another as a graph optimization problem: the graph represents the two songs in the transition range, and a cut happens when we transition from one song to the other at a particular time point. To obtain these representations we first apply a short term fourier transform (STFT) to each song, and then convert the resulting complex time-frequency mapped amplitude values into real decibel values. In order to align the tracks we apply rudimentary tempo matching and beat alignment using the libROSA Python library, and overlap the tracks by a number of beats [7]. We call the resulting STFT transformed data of the first and second song's overlapped segments matrix A and matrix B

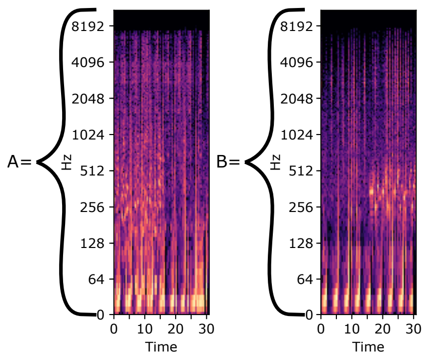

Figure 2: Overlapping song segment spectrogram's used to compute graph weights using eq. (1). Each segment represents 64 beats.

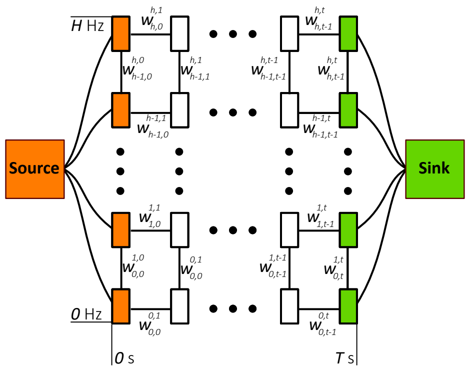

Figure 3: Flow graph model showing adjacent weight calculations. The orange and green nodes are fixed to the first and second songs respectfully. Nodes represent time-frequency bins.

respectfully, as seen in Figure 2. Next, we define a simple loss function:

$$w^{k,l}_{i,j}(A,B) = ||A_{i,j} - B_{i,j}|| + ||A_{k,l} - B_{k,l}|| \tag{1}$$

We apply the loss function to each adjacent time-frequency bin and use the resulting values to assign weights to edges in an undirected graph with dimensions equal to A and B. Finally, the resulting graphs left-most nodes are anchored to a source node, and the right-most nodes are anchored to a sink node. Figure 3 shows a representation of the completed flow graph. In order to obtain a min-cut, we apply the Boykov-Kolmogorov algorithm [2]. The indices of this min-cut are the seam where each frequency bin transitions.

In order to apply the found transition to the audio tracks, we concatenate the complex time-frequency song representations found earlier along the seam, and apply an inverse STFT to obtain the final audio transition seen in Figure 1.

The work here presents an initial foray into automatically transitioning between songs in the time-frequency domain. The loss function described in 1 does not well characterize the inherent qualities found in music, but there is good reason to believe such a cost function can be found through further development. On harmonically similar tracks with similar tempi, the current implementation produces acoustically pleasing results.

REFERENCES

[1] Rachel M Bittner, Minwei Gu, Gandalf Hernandez, Eric J Humphrey, Tristan Jehan, P. Hunter McCurry, and Nicola Montecchio. Automatic Playlist Sequencing and Transitions. *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, pages 442–448, 2017.

[2] Yuri Boykov and Vladimir Kolmogorov. An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision 1 Introduction. Technical Report 9, 2004.

[3] Roman B. Gebhardt, Matthew E.P. Davies, and Bernhard U. Seeber. Psychoacoustic approaches for harmonic music mixing. *Applied Sciences*, 6(5), 2016.

[4] Tatsunori Hirai, Hironori Doi, and Shigeo Morishima. MusicMixer: Computer-Aided DJ system based on an automatic song mixing. In *ACM International Conference Proceeding Series*, volume 16-19-Nove. Association for Computing Machinery, nov 2015.

[5] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: Image and video synthesis using graph cuts. In *ACM SIGGRAPH 2003 Papers, SIGGRAPH '03*, pages 277–286, 2003.

[6] Heng Yi Lin, Yin Tzu Lin, Ming Chun Tien, and Ja Ling Wu. Music paste: Concatenating music clips based on chroma and rhythm features. In *Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009*, pages 213–218, 2009.

[7] Brian McFee, Colin Raffel, Dawen Liang, Daniel Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and Music Signal Analysis in Python. In *Proceedings of the 14th Python in Science Conference*, pages 18–24, 2015.

[8] Jaume Parera. *Dj codo nudo: a novel method for seamless transition between songs for electronic music*. Master's thesis, Universitat Pompeu Fabra, Barcelona, 2016.