VisualSoundtrack: An Approach to Style Transfer in the Context of Soundtrack Prototyping

Ishwarya Ananthabhotla
Responsive Environments Group
MIT Media Laboratory
ishwarya@media.mit.edu

Joseph A. Paradiso
Responsive Environments Group
MIT Media Laboratory
joep@media.mit.edu

ABSTRACT

In this work, we present VisualSoundtrack, a system designed as a tool for soundtrack composers to experiment with original musical content in differing musical "styles". The system allows a user to rapidly prototype musical ideas with respect to the target media (such as a film or podcast) by having him/ her input original musical motifs, capitalizing on a corpus of existing soundtrack samples to source various styles, and allowing the user to identify the most appropriate style sources for the target media by visually architecting a path through a highly abstracted feature space. The contributions of this paper include the presentation of the novel concept, a description of the fully-functional system design, a focus on the approach taken to achieve the "style transfer", and an initial evaluation of its effectiveness.

1. INTRODUCTION

While research in computer music over the last few decades has produced several intelligent digital tools for music performers, there is still a scarcity in such tools for traditional *composers*. There are even fewer tools available that allow composers to rapidly prototype music in the experimental stages of composition without having to fully realize the final style of the music. Specifically in the context of soundtrack composition, where the development of a final work often entails a set of musical themes that are repeated, varied, and rendered in different styles to suit the visual/ audio cues of the target media [1], there exists scope for a novel tool that allows a composer to experiment quickly by envisioning their own musical content in forms of varying orchestration, instrumentation, etc. Such a tool would need to consider two critical factors– it must employ an input interface that provides operational controls relevant to soundtrack composition, allowing a composer to architecture a sample soundtrack with respect to the target media; and it must perceivably achieve artificial "style transfer" by algorithmically combining a user's content and intended style model.

A few platforms have been developed addressing the first with respect to soundtrack composition. Work by [2] and [3] present models for intuitive and creative interfaces that allow composers to capture musical ideas multimodally at a very early stage. An excellent work by [4] presents a computer-aided soundtrack composition system where musical motifs input by a composer are automatically reorganized to match a specified repetition pattern and aligned with cues in the target media. However, these platforms are at opposite ends of spectrum in terms of automation, and do not incoporate style transfer as a method for experimentation: the final output is exactly a function of what is input by the user. The notion of style transfer and style synthesis has also been explored substantially in signal processing literature. Prior literature has presented "texture" as an analog or sub-problem of "style", and work by [5, 6, 7, 8] present effective statistical and linear-predictive modeling approaches to texture synthesis problems, although operating explicitly on unpitched audio samples. More recently, an approach to audio style transfer using Deep Neural Networks [9], modeled after the approach to image style transfer [10], has been developed- however, the resulting audio is highly input specific with poor audio quality, and has been qualitatively evaluated by study participants to be a weak attempt at style transfer in comparison to the work produced by this system. [1]

This paper seeks to build upon some of the mentioned prior work to create a tool to aid a composer in experimenting with musical ideas in different styles with respect to the target media, by allowing a user to provide their own musical content, capitalize on existing soundtrack samples to source various styles, and allowing a user to identify the most appropriate style sources for the target media by visually architecting a path through a highly abstracted feature space. To this end, we do not attempt to define or model "style", but instead investigate a novel, intuitive approach to mapping "style" from a source on to musical content, which we refer to as "style transfer." The contributions of this paper are as follows:

1. A detailed presentation of the novel, fully-functional VisualSoundtrack system including:
   - A novel approach to architecting a template track (sourcing style material) from existing soundtrack samples using a visual, high-level feature approach

[1] This conclusion is drawn from qualitative feedback provided by the same participants who completed the evaluation; Samples produced by the DNN [9] for comparison can be found at resenv-music.media.mit.edu/VS/samples.

– A simple but effective approach to style transfer in the context of soundtrack composition, using a process to extract and modify "representative samples"

2. A discussion of conclusions drawn from a qualitative study conducted to evaluate the effectiveness of the style transfer approach

2. SYSTEM OVERVIEW

An overview of the system design is shown in Figure 1. The system consists of three main sub-systems– the Sketching Interface, the Template Assembly, and the Audio Processing Pipeline. The system flow can be described at a high-level by the following steps:

1. A soundtrack composer begins with Sketching Interface, where they upload the target audio [2] of their choice. Here, they can "sketch" contours along eight abstract musical features as a description of how they vary over time with the target media.

2. The Template Assembly module then uses these contours to select a sequence of soundtrack segments from a corpus of existing soundtrack music that best match the target media according to the user's input. The Template Assembly module generates a preview "Template Track" that is rendered back to the Sketching Interface, allowing a user to edit and iterate on their contour sketches.

3. A user may then upload a monophonic MIDI file that represents their original content, which is used to build a generative melody model.

4. When the user is ready to envision their content in the "style" of the selected soundtrack segments, the Audio Processing Pipeline operates on each segment, algorithmically combining it with the output of the melody model, and produces a new, synthesized soundtrack segment.

5. These synthesized segments are provided back to the Template Assembly module to be stiched together, and are finally rendered to the Sketching Interface where a user can listen to the final output.

Each of the sub-systems and processes described here will be presented in detail in the sections below.

3. SKETCHING INTERFACE

The Sketching Interface is the user's entry point into the system, and a screenshot demonstrating the interface can be seen in Figure 2. The UI displays an animated audio waveform that allows a user to play, pause, and scrub through the audio for which they intend to craft a soundtrack, which will be referred to as the "target" audio. Additionally, it presents a series of eight "graphs", or sketching canvases where a user can draw and edit a series of connected points. Each canvas represents one of eight high-level audio features that have been selected from Spotify's

[2] The target media has been restricted to audio here for ease of demonstration, but the system can be extended to work with any form of media.

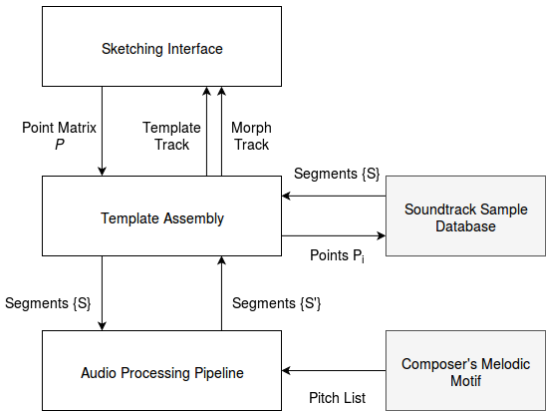

Figure 1. High-level overview of the Visual Soundtrack system and its modules.

Web API, namely "Valence", "Acousticness", "Instrumentalness", "Energy", "Tempo", "Danceability", "Liveness", "Loudness"[11]. As the focus of this work is neither on high-level feature extraction nor on improved classification schemes, the decision was made to select and use features from a third-party API that would be relevant for a soundtrack composer to take into consideration. The vertical location of each point on the canvas represents its value, as specified by the range of that dimension within the Spotify API. Each point also has an associated timestamp, derived from the current playback point of the target audio. A user may draw and edit their "sketch" for each of the eight dimensions to align with the target audio. For example, an emotionally intense sequence in the target audio might lead the user to draw a taller sketch on the "Valence" graph but include a valley in their "Acousticness" graph. The canvases can be edited independantly, simultaneously, or as many times as wanted, while the audio can be replayed or scrubbed for fine tuning. When desired, the user can select the "Generate Template" button, which then renders a "template" track, or an assembly of original soundtrack segments based on the sketch data, on to the screen (see Section 4). This process of adjusting the sketch and generating the corresponding template can be repeated until the user is satisfied with the compatibility between the target audio and the template track.

Finally, the user can upload the MIDI file containing their composed motif by selecting "Upload MIDI File", and then generate the fully-morphed, style-transfered track by choosing "Make Morph".

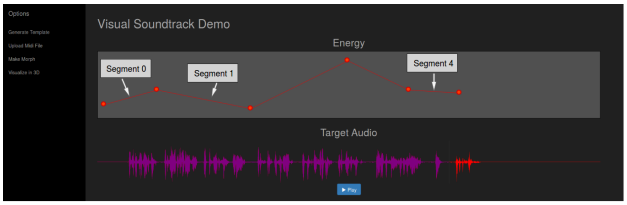

Figure 2. A screenshot showing the VisualSoundtrack Sketching Interface.

4. TEMPLATE ASSEMBLY

The Template Assembly module serves four major functions: acquisition and feature extraction using the Spotify API for the corpus of audio data in the Soundtrack

Sample Database, interpolating the point data returned by the sketching interface, retrieving audio samples from the database that corresponds to the input data, and performing traditional audio editing techniques to stitch together a full soundtrack either from the original soundtrack segments or from the synthesized soundtrack segments.

The corpus of soundtrack samples in the audio database is obtained from instrumental-only soundtrack albums available from Spotify, where identical copies of the audio tracks contained in these albums are stored locally and mapped to the Spotify URIs[11]. For the initial protoype system, this database contains approximately 200 soundtrack samples, and the feature metadata for each track was collected and mapped to the audio in the database.

The next task performed by this module is data interpolation of the point vectors (consisting of the point values and associated timestamps) provided by the Sketching Interface. The data is consolidated by ordering the timestamps across all vectors and combining values across common timestamps to produce the point matrix P. However, given that timestamps do not usually coincide or that users operate on each of the dimensional canvases at different time resolutions, P tends to be sparse. To ensure continuity within a dimension, and to account for incomplete time graphs, P is forward and backwards interpolated through time. This process is illustrated by the graphic shown in Figure 3.

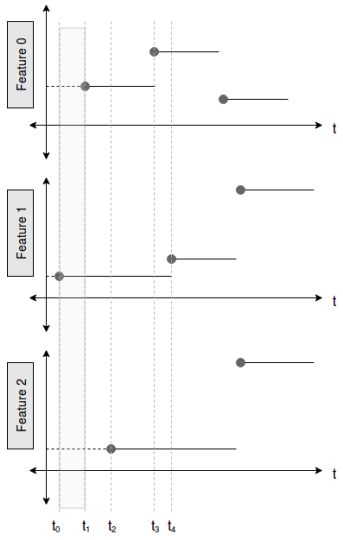

Figure 3. An illustration of the interpolation process on a subset of 3 features performed by the Template Assembly is shown. Each feature retains the value of a new point added by a user until another point is drawn. The feature values for Point p_0 in matrix P with associated timestamp t_0 is given by the shaded box. These values are used to search for soundtrack segment S_0 which is returned with a duration of t_1 - t_0.

Each row in the fully interpolated point matrix P represents the feature data pertaining to a single soundtrack segment, and can be treated as a point p in 8-dimensional feature space. In order to map each point p_i to its corresponding segment S_i in the Soundtrack Sample Database, a standard KD-tree for quick nearest-neighbors look-up is implemented. Once the nearest soundtrack clip is identified, it is shortened (or repeated) until its length matches the duration associated with p_i, given by the difference between its timestamp t_i and t_{i+1}. Finally, for each set of segments S or S' that is returned to the Template Assembly either from the Soundtrack Sample Database or from the Audio Processing Pipeline (see Section 5), volume adjustment and proportional crossfade are programmatically applied to the segments to create a seamless template or morph track.

5. AUDIO PROCESSING PIPELINE

One of the main contributions of this paper surround the presentation of a novel approach to style transfer that has the potential for success within the specific context of soundtrack composition. In order to apply the melodic content from a MIDI file to the style of an existing soundtrack segment, the processing pipeline aims to identify a distribution of "representative texture samples", or samples from the existing soundtrack segment estimated to be stylistically homogenous, which are then pitch shifted or time stretched to align with a generated content model. Finally, these samples are fitted to the original or a generated percussive track to produce the final "morphed" soundtrack. While this approach might seem restrictive, it is well suited to operation on stereotypical soundtrack music content, where there tends to be more variation in dynamics, melody, and pace than in instrumentation over a given composition.

Figure 4 shows a breakdown of the Audio Processing Pipeline, which will be presented in detail below. In order to begin the transformation from S_i to S_i', we first apply Harmonic-Percussive Source Separation (HPSS) to S_i [12]. The resulting harmonic and percussive components are then processed separately.

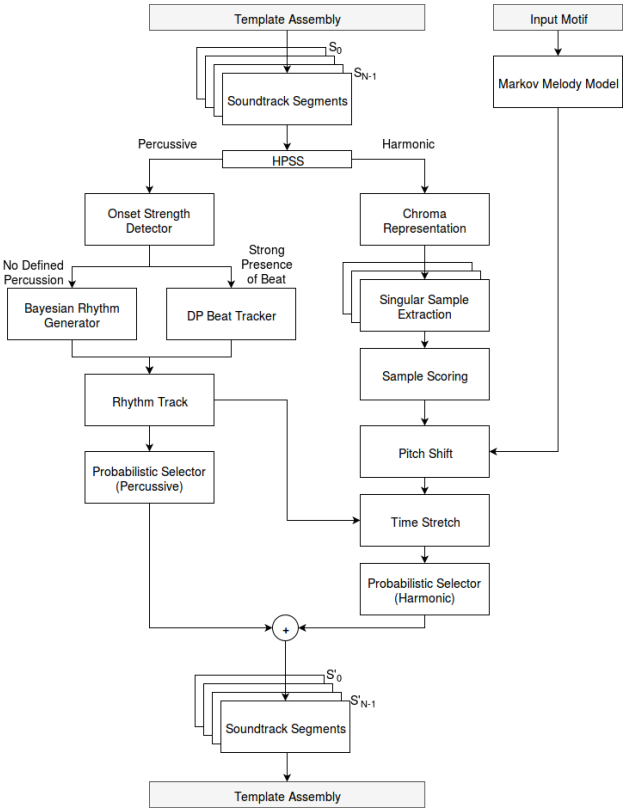

Figure 4. A detailed view of the Audio Processing Pipeline.

5.1 Percussive Pipeline

The first objective of the percussive pipeline is to identify whether there is a strong presence of a consistent beat or rhythm. To do this, a spectral flux onset strength envelope with vibrato suppression [13] is first applied to the percussive component. Next, a dynamic programming beat tracker based on the implementation in [14] is applied to the percussive component. For each beat frame returned by the tracker, its corresponding onset envelope value is compared against an intensity threshold th. Finally, if the number of beat frames in a segment that clear the intensity threshold exceed a beat percentage threshold bp, the segment is considered to have a strong beat present. The thresholds were determined by manually labeling a small subset of 40 soundtracks by the binary classification of having or not having a strong beat, and aggregating the parameters th and bp for each track across the cluster. Figure 5 demonstrates a sample clustering to generate the threshold boundaries.

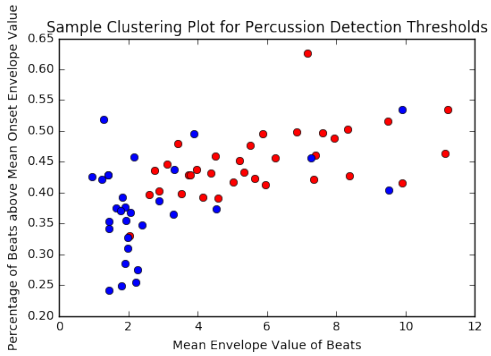

Figure 5. An example of manually labeled soundtrack data that was clustered to determine thresholds th and bp. The red points are labeled as having a strong percussive track, while the blue points are labeled as having a weak percussive track.

If the soundtrack segment was labeled by the thresholding process as containing a defined rhythm, the percussive component was used as the rhythmic track, or the template upon which the pitched samples from the harmonic pipeline would be overlaid (see section 5.4). If not, this rhythmic track was generated by a probabilistic process.

This process works by choosing a series of rhythmic units that together total to the desired length of the track. Each unit is defined by the following properties: it is either a note or a rest; it has a base value that can either be a whole, half, quarter, eighth, or sixteenth note; and it can have a modifier value (which would be considered a tie, grace note, etc) that is a whole, half, quarter, eighth, sixteenth note, or 'None'. The total value of the units that need to be generated is determined by the length of the segment, the segment time signature, the tempo estimated by the beat tracker, and duration value assigned to the smallest unit (a sixteenth note), q. To meet the total value, each rhythm unit is selected from a probabilistic distribution that is reweighted to emphasize unseen units at each time step, until the total number of units required has been produced. The set of units are then sorted by duration value, or the multiple of q that is assigned to the duration of a unit. Finally, a rhythmic sequence is formed by randomly

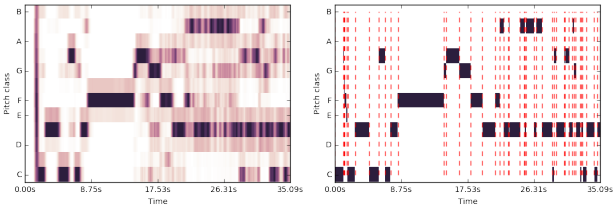

Figure 6. Left: Temporally smoothed chromagram of a soundtrack audio sample for $L = 15$; Right: Max-energy chromogram with onset lines.

sequencing the selected units, weighting longer unit durations higher on downbeats of measures for a more natural feel. Intuitively, this algorithm has been designed such that varied and complex rhythms are produced without predictability, so as to demonstrate the flexibility of the style transfer approach while also providing versatile suggestions to the composer.

5.2 Harmonic Pipeline

In parallel to the percussive processing, the harmonic processing pipeline operates on the harmonic component which results from the HPSS. First, the harmonic component is used to generate a log-scaled chromogram C[15]. In order to identify musically continuous segments, the chromogram is then temporally smoothed. For each chroma vector c_i in C,

$$c_i^s = \frac{1}{L} \sum_{l=-(L-1)/2}^{(L-1)/2} c_{i+l} \tag{1}$$

where L is the length of the smoothing window, and c_i^s is the new, smoothed vector. A binary mask is then applied selecting the maximum energy value across each chroma vector, and "onset lines" with tolerance for the chosen smoothing window are drawn to mark contiguous segments. The transformation of a sample chromagram through these stages is shown in Figure 6. The set of samples s between each consecutive pair of onset lines form the set of candidate samples for the final collection of "representative samples" for segment S_i. Each of the audio samples identified by the onset lines are evaluated by a set of three metrics to identify the most suitable samples for forming the synthesized track. Each metric assigns the sample a score $E_{y,j}$ ranging from 0 to 1, where y is the sequential index of the contiguous sample (or the onset lines) within segment S_i and j is the relevant metric. The first metric discourages short samples to reduce the need for time stretching during the assembly portion of the pipeline (see 5.4), and is given by:

$$E_{y,length} = \begin{cases} 2^{\frac{x}{t_l}} - 1 & 0 \leq n \leq t_l \quad (2) \\ 1 & x > t_l \quad (3) \end{cases}$$

where x is the length of the sample in seconds. The threshold t_l is determined by the performance of the time stretching implementation that is used (see Section 5.4), and is the value of time above which audio quality degrades substantially when stretched. The second metric is used to determine local homogeneity within a sample, by comparing neighboring chroma vectors. The score is given by:

$$E_{y,hom} = \frac{M_{hom}(s_y) - min(M_{hom}(S))}{max(M_{hom}(S)) - min(M_{hom}(S))} \tag{4}$$

$$M_{hom}(s_y) = \sum_{k=1}^{k=N} (c_k \cdot c_{k-1}) \quad (5)$$

where k is the index of the unmasked chroma vectors within the sample being analyzed and $M_{hom}(S)$ is the set of values for the metric M_{hom} evaluated across all samples s_y in soundtrack segment S_i. The resulting score assigned to the sample, $E_{y,hom}$ is a linear function spanning the range of the metric values normalized over the range from 0 - 1. The third metric is used to determine the polyphonic characteristic of the sample, by assessing the amount of extraneous energy in a chroma vector which does not contribute to the fundamental pitch class of the vector. To do this, a set of template chroma vectors representing the energy distribution for the harmonic overtones of each pitch class as the fundamental frequency was generated. The template vectors T_f were generated by assigning energy to each harmonic of the fundamental pitch f as an exponentially decaying series with decay factor of $\alpha < 1.0$. To compute the monophony score for the entire sample, the template vector corresponding to the fundamental pitch of each chroma vector in the sample was compared to the chroma vector, and integrated across the sample. The score is given by:

$$E_{y,mon} = \frac{M_{mon}(s_y) - min(M_{mon}(S))}{max(M_{mon}(S)) - min(M_{mon}(S))} \quad (6)$$

$$M_{mon}(s_y) = \sum_{k=0}^{k=N} (T_{f(c_k)} \cdot c_k) \quad (7)$$

where, again, k is the index of the chroma vectors within the sample being analyzed, $M_{mon}(S)$ is the set of values for the metric M_{hom} evaluated across all samples s in soundtrack segment S_i, and $E_{y,mon}$ is a linear function spanning the range of the metric values normalized over the range from 0 - 1. A visualization of the assigned scores for all samples isolated from the sample soundtrack segment in Figure 6 is shown in Figure 7.

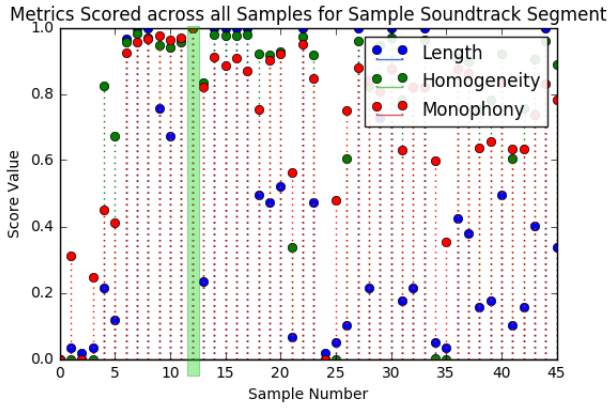

Figure 7. The scoring metrics (with t_l = 1s) applied to the same chromagram as in Figure 6 is shown, where the sample numbers are given by the index of the onset lines. The highest scoring sample which is selected for modification, corresponding to the segment at t=8.75 in Figure 6, is highlighted.

Finally, an aggregate score for every sample is simply computed by weighting the individual metric scores and normalizing:

$$E_{y,total} = \sum_{j \in \{length,hom,mon\}} w_j E_{y,j}$$
$$\sum_{j \in \{length,hom,mon\}} w_j = 1 \quad (8)$$

We choose the set of representative samples R based on the number of samples dictated by the total length of the segment S_i we are interested in morphing:

$$L(R) = \begin{cases} 1 & L(S_i) < p \quad (9) \\ \left\lfloor \frac{L(S_i)}{p} \right\rfloor + 1 & L(S_i) \geq p \quad (10) \end{cases}$$

where L is a function indicating length or count, p is a time parameter set by the user representing the minimum amount of time for a which a single representative sample must play in the synthesized soundtrack. The set R is simply determined by the $L(R)$ highest scoring samples s in segment S_i.

5.3 Melodic Generation and Morphing

With the representative texture samples and rhythmic track prepared, the content must be generated from the user's input. As mentioned previously, a user can upload a monophonic MIDI file, which is modeled by the system as a list of pitches, ignoring rhythm. To generate musical variations on this melody while still preserving the underlying content that should be perceivable by a listener, a Markov Chain with variable order is trained on the MIDI data and used to output a stochastic sequence of pitches. The Markov Chain order determines the similarity between the generated pitch sequence and the original MIDI data, and can be set by the user. The number of pitches that must be generated by the sequence is determined either by the number of beats detected by the rhythm track extraction process, or alternatively, the number of units that have been produced by the probabilistic rhythm generator process (see Section 5.1). Next, the selected representative samples are pitched shifted to match the pitches in the generated sequence. If multiple samples are present in R for a given S_i, boundaries for a change in the representative sample are drawn randomly across the sequence of pitches while preserving the constraint that no section between boundaries is less than p in length.

5.4 Assembly

In the final step of the processing pipeline, the series of pitch shifted samples are overlaid on to the rhythm tracks output by the percussive processing pipeline. In the case where a rhythm track is extracted from the original segment S_i, the pitch shifted samples are time stretched to match the duration between subsequent detected beats. In the case of the generated rhythm track, the samples are time stretched to match the duration of each rhythmic unit. To perform the time stretching without significantly compromising audio quality, a phase vocoder algorithm (based on [16]) was implemented. In the first case, the percussive track and the pitched samples were vertically appended, with consecutive samples stiched together by a fractional

crossfade and equalized for volume amplitude local to a segment. In the second, the pitched samples were simply stiched together in time with the application of a crossfade and equalized for volume amplitude. The result of a single assembly is the new synthesized soundtrack segment S_i'. The process described in this pipeline is applied to every segment S_i from the template track to produce each new, morphed segment, S_i'. These segments are returned to the Template Assembly block, where they are processed and rendered to the user interface as a single morphed track.

6. EVALUATION AND DISCUSSION

For the purpose of demonstration and evaluation [3] , the parameter settings shown in Table 1 were used.

th	bp	t_l	p	γ	α	w_{length}	w_{hom}	w_{mon}
4.0	0.40	1s	30s	0.15	0.70	0.33	0.33	0.33

Table 1. The list of parameter settings used in generating audio samples for demonstrative and evaluation purposes.

In order to draw conclusions regarding the effectiveness of the presented approach to style transfer in the context of soundtrack music, a study [4] was designed and was completed anonymously by 10 individuals. The study intended to examine fundamental principles of the work, including the effectiveness of style transfer, feasibility as a compositional tool, and the side effects of digital synthesis. This was done by asking participants to complete tasks such as selecting the synthesized audio clip amongst a set of choices that best matched a sample MIDI melodic motif, rating the compatibility between accompanying soundtracks and target audio samples, and rating stylistic similarity amongst sets of synthesized soundtrack clips. The Sketching Interface as a tool for composition was not evaluated in this study, and will be presented in further detail in future literature. The results of the study produced several key insights. For example, an excellent accuracy across the matching exercises served as a testament to the style transfer approach taken in this work. Additionally, while target media compatibility ratings did not waver between pairs of sample template tracks and morph tracks, generally poor ratings indicated greater scope for development in the template generation process alone. And lastly, most users perceived a degradation of audio quality and naturalness in the synthesized acoustic samples, which suggests a need for improvement in the morphing pipeline. Ultimately, this work demonstrates the novelty and feasibility of a style transfer-based compositional prototyping tool; future work on this system will focus on development to reflect feedback from the pilot study and on detailed, end-to-end system evaluations.

Acknowledgments

We would like to thank Spencer Russell for the meaningful discussions and feedback, and all of the study participants for volunteering their time.

[3] Sample audio clips: resenv-music.media.mit.edu/VS/samples

[4] Study questionnaire: resenv-music.media.mit.edu/soundtrack

7. REFERENCES

[1] F. Karlin and R. Wright, *On the track: A guide to contemporary film scoring*. Routledge, 2013.

[2] D. MacDonald and T. Stockman, "Toward a method and toolkit for the design of auditory displays, based on soundtrack composition," in *CHI'13 Extended Abstracts on Human Factors in Computing Systems*. ACM, 2013, pp. 769–774.

[3] S. Abrams, R. Bellofatto, R. Fuhrer, D. Oppenheim, J. Wright, R. Boulanger, N. Leonard, D. Mash, M. Rendish, and J. Smith, "QSketcher: an environment for composing music for film," in *Proceedings of the 4th conference on Creativity & cognition*. ACM, 2002, pp. 157–164.

[4] E. Vane and W. Cowan, "A Computer-Aided soundtrack Composition System designed for Humans." in *ICMC*, 2007.

[5] J. H. McDermott and E. P. Simoncelli, "Sound texture perception via statistics of the auditory periphery: evidence from sound synthesis," *Neuron*, vol. 71, no. 5, pp. 926–940, 2011.

[6] M. Athineos and D. P. Ellis, "Sound texture modelling with linear prediction in both time and frequency domains," in *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, vol. 5. IEEE, 2003, pp. V–648.

[7] D. Schwarz, "State of the art in sound texture synthesis," in *Digital Audio Effects (DAFx)*, 2011, pp. 1–1.

[8] S. Dubnov, Z. Bar-Joseph, R. El-Yaniv, D. Lischinski, and M. Werman, "Synthesizing sound textures through wavelet tree learning," *IEEE Computer Graphics and Applications*, vol. 22, no. 4, pp. 38–48, 2002.

[9] D. Ulyanov, "Audio Texture Synthesis and Style Transfer," https://dmitryulyanov.github.io/audio-texture-synthesis-and-style-transfer.

[10] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," *arXiv preprint arXiv:1508.06576*, 2015.

[11] "Spotify Web API." https://developer.spotify.com/web-api/.

[12] D. Fitzgerald, "Harmonic/percussive separation using median filtering," 2010.

[13] S. Böck and G. Widmer, "Maximum filter vibrato suppression for onset detection," in *Proc. of the 16th Int. Conf. on Digital Audio Effects (DAFx). Maynooth, Ireland (Sept 2013)*, 2013.

[14] D. P. Ellis, "Beat tracking by dynamic programming," *Journal of New Music Research*, vol. 36, no. 1, pp. 51–60, 2007.

[15] D.P Ellis, "Chroma feature analysis and synthesis," http://labrosa.ee.columbia.edu/matlab/chroma-ansyn/.

[16] D.P. Ellis, "A phase vocoder in Matlab." http://www.ee.columbia.edu/~dpwe/resources/matlab/pvoc/.