MedleyAssistant – A system for personalized music medley creation

Zhengshan Shi
CCRMA, Stanford University
Stanford, USA
kittyshi@ccrma.stanford.edu

Gautham J. Mysore
Adobe Research
San Francisco, USA
gmysore@adobe.com

ABSTRACT
In this paper, we present MedleyAssistant, a system to assist in the creation of music medleys from segments of existing music. Our goal is to make medley creation more accessible to novices, while still allowing them to express their own creative style. Our system addresses two key challenges in medley creation – determining which segments of music sound natural when transitioning to which other segments of music, and determining specific transition points between two given segments of music. This constrains the problem so that medleys created with our system tend to sound natural while allowing the user to be creative with music selection. We also provide a music visualization that helps users understand the musical principles of medley creation.

ACM Classification Keywords
H.5.5. Sound and Music Computing : Methodologies and Techniques; I.5.5. Implementation : Interactive systems

Author Keywords
music medley; creative MIR; personalized music creation.

INTRODUCTION AND MOTIVATION

A music medley is a piece of music that is composed or arranged from a series of songs or musical segments. In a high quality medley, each segment tends to naturally flow into the next segment, and the transitions typically sound seamless.

Medleys provide a way to create new variations of music starting from existing music. They can be used for music playback by itself or as backing tracks for media such as videos and video games. They provide a way to customize a piece of music so that different sections of the media correspond to different segments of music.

Manual creation of high quality medleys can be a challenging task and typically requires a background in music and audio editing. They are often created by musicians and DJs. The typical sequence of steps to create a medley is as follows:

1. Select a number of candidate musical segments from various pieces of music.
2. Determine a musically natural sequence of segments for the proposed medley from the above candidate set of segments.
3. Determine the exact transition points from a given segment to the following segment, crop the segments accordingly, and use a crossfade to stitch the segments together. This step is crucial for a seamless transition between segments.
4. Adjust tempos or keys when necessary by using traditional audio editing tools.

Steps 2 and step 4 above require a keen musical ear or a background in music theory. Step 3 requires a certain amount of skill in audio editing. All three of these steps can be quite tedious. Step 1, however requires less of a prior background in music and audio editing and can simply be based on music preference.

We present MedleyAssistant, a system to help people easily create personalized music medleys with little or no background in music and audio editing. Our system assists in step 2 and automates step 3. It allows users to be creative with song selection and step 1. Moreover, it visualizes certain musical features to help guide users with these steps and can help them better understand the underlying musical principles. We believe that this can make medley creation a more accessible process for novices, and allow experts to speed up their workflow.

RELATED WORK

Recent advances in Music Information Retrieval (MIR) techniques have given rise to intelligent musical interfaces [3, 9], making certain aspects of music creation more accessible to non-experts. This includes applications such as an automatic DJ [4], a song mixing tool [2], an automatic mashup system [1], and a loop creation system [11]. All of these applications help reuse parts of existing music to create new music.

To the best of our knowledge, the work that is most closely related to our proposed medley creation system is Music Cut and Paste [5, 6], a personalized music-cut-and-paste system, which is also used to create medleys. The key difference is that this system only allows users to specify the sequence of segments in terms of vocal and instrumental sections, whereas our system provides the user with significantly more flexibility in terms of choosing and adjusting segments.

MILC'18, March 11, 2018, Tokyo, Japan

SYSTEM OVERVIEW

In this section, we describe the workflow and interface of MedleyAssistant. A user creates a medley using our system as follows:

1. The user provides a number of candidate songs based on their preference in music.
2. Our system provides a visualization of these songs as shown in Figure 1. Specifically, it visualizes the chord structure and tempo. This visualization could help guide the user in the subsequent steps.
3. The user chooses the first segment of the medley based on the music that they would like to be the introduction. This forms the foundation of the medley because it in turn dictates subsequent segments. The user can choose this first segment based on listening as well as using the visualization as a guide. The beginning and end of the selection snaps to the nearest beat.
4. Based on the first segment, the system assists in choosing the second segment. It estimates multiple potential candidates for the second segment (over all songs) and highlights them, as shown in Figure 2. The opacity of each gray box indicates a confidence level of how good the segment will sound. The system attempts to choose candidate segments that will musically flow well from the first segment based on chord structure, tempo, and timbre. The chosen segments will be 16 beats or less. We use a relatively small size so that the user can adjust the length to whatever they desire in the next step. The user can choose to use one of these segments, and the system will then perform the low level edits to concatenate the two segments. However, the user can alternatively choose any part of any song (even if it is not highlighted) and use that as the second segment.
5. After choosing a segment, the user can drag the segment ending boundary to adjust the length of the segment.
6. The user chooses the next segment as outlined in the previous two steps and continues this process until the medley is complete.

The interface in our system is realized as a web application using wavesurfer[1], an API built on top of Web Audio API[2] and HTML5 Canvas.

Visualization

As shown in Figure 1, our system visualizes both the chord structure as well as the tempo for each song. This visualization provides a tool for users to better understand the segments highlighted by the system, helps users choose new segments that are not highlighted, and can serve as an educational tool to better understand the music theoretical principals of medley creation.

We visualize the chord structure of each song by overlaying the waveform on colored blocks where each color corresponds to a different chord. We use only Major and Minor chords,

[1]https://wavesurfer-js.org/

[2]https://www.w3.org/TR/webaudio/

Figure 1. MedleyAssistant Interface. For each song, we visualize chords (as colored blocks on the waveform) and tempo (as a color density bar below the waveform).

so we have a total of 24 chords. The color scheme we used in visualizing each chord is inspired by Alexander Scriabin's "Clavier à lumières" ("keyboard with lights") [10], a keyboard instrument with colors assigned to different keys. For example, we use intense red for C and orange for G. We utilize this color scheme because similar color produces a desirable chord progression, such as the proximity in RGB color value of the chord C and G. The color mapping of the chords are shown in a chord visualization colormap in Figure 1.

We visualize tempo below the waveform, with a bar representing rhythmic density of the music as a function of time. For the rhythmic feature, we first extract the dynamic tempos of the song, and then group the dynamic tempos into different sections. We use color interpolation to indicate the tempo density (i.e: deeper color for a faster tempo).

Our system demonstrates the visualization of two features (chord structure and tempo), but this can be extended to various other features. We think that different kinds of musical features could be used to assist in different forms of medley creation styles.

ALGORITHMS

In this section, we describe the algorithms that we use for visualization and automatic segment selection (as described in steps 2 and 4 of the workflow in the previous section).

We start with a pre-processing step in which we extract features at each beat of each song. These features are used both

Figure 2. When a query segment (the gray section in the first song) is selected, a menu with a segment selection option appear at the bottom of the song such that the user can add the segment into the medley editor (bottom row). The system then calculates and suggests the user potential next segments (the gray boxes in the second and the third song.)

in the visualization and computation of the cost function described below.

Given a specific segment of the medley, which we refer to as the *query segment*, the goal of our algorithm is to estimate potential candidates for the next segment. We select the next candidate segment based on specific criteria as follows:

1. Acoustically smooth and seamless transitions between consecutive segments. We compute this smoothness based on a cost function between the query segment and every other segment in every song based on a sliding window.

2. Consecutive segments should conform to a harmonic progression specified by music theory rules. Based on the acoustically smooth candidates (as determined by the previous step), we select a subset of candidate segments based on a harmonic progression factor.

Feature Extraction

We first estimate beat locations in each song by applying beat tracking on the onset envelope of the audio signal. At every beat of every song, we compute timbral features (Mel-frequency cepstrum coefficients), signal energy level (root mean square), and harmonic features (chroma vectors). We also compute tempo over a window around each beat. We compute these features using librosa [8]. Additionally, we estimate the chords in each song using the chordino plugin based on NNLS Chroma [7]. We compute this in the VamPy Host. These estimated chords are used in the visualization under the waveform.

Acoustic Smoothness

Given the query segment, S_1 of length N, and the potential candidate segment S_2 of length $M = 16$, our goal is to compute the optimal transition point from S_1 to S_2. We do this by computing a cost function between four-beat windows sliding over both S_1 and S_2. We define the optimal transition point by the location of the sliding windows with the lowest cost, which we refer to as the highest acoustic smoothness.

We consider four beat sliding windows starting from beat $\frac{1}{4}N$ to beat $N-3$ of S_1, and beat 1 to beat $\frac{3}{4}M$ of S_2. We do not consider the beginning of S_1 and the ending of S_2 in order to ensure that the resulting combination of S_1 and S_2 retain at least a part of each segment.

Our cost function over a four-beat sliding window of S_1 to S_2 is:

$$\begin{aligned} C(S_{1i}, S_{2j}) = \alpha \sum_{k=0}^{3} D_c(C_{S_1}[i+k], C_{S_2}[j+k]) \\ + \beta \frac{\sum_{k=0}^{3} D_m(M_{S_1}[i+k], M_{S_2}[j+k])}{\sigma_m} \\ + \gamma \frac{D_r(R_{S_1}[i-4:i-1], R_{S_2}[j:j+3])}{\sigma_r} \\ + \delta \frac{D_t(S_1, S_2)}{\sigma_t} \end{aligned} \quad (1)$$

Where D_c denotes the cosine distance of the chroma features, D_m denotes the Euclidean distance of the timbre features, D_r denotes the difference of root mean square energy, and D_t denotes the tempo difference. α, β, γ, and δ are tuning factors, whereas σ_m, σ_r, and σ_t represents the standard deviations. Figure 3 illustrates the computation of the cost function.

We compute our tuning factors apriori as follows. The goal is to define the tuning factors that help indicate acoustic smoothness. By definition, the transitions between consecutive segments of a given song are maximally smooth. Therefore we compute the cost function between all consecutive segments of a number of songs using a number of different combinations of tuning factors (each tuning factor can vary from 0 to 1.0). We choose the combination of tuning factors that on average yields the lowest cost.

When we compute the cost function for segments S_1 and S_2, we choose the optimal transition point between S_1 and S_2 based on the i and j that yield the lowest cost. Given the optimal i and j, the optimal transition between the segments is to go from beat $i-1$ of S_1 to beat j of S_2.

The query segment S_1 has a cost with respect to each segment S_2 (associated with the optimal transition point) of each song. We choose all of the segments S_2 that have a cost under a threshold as candidate segments for the next step.

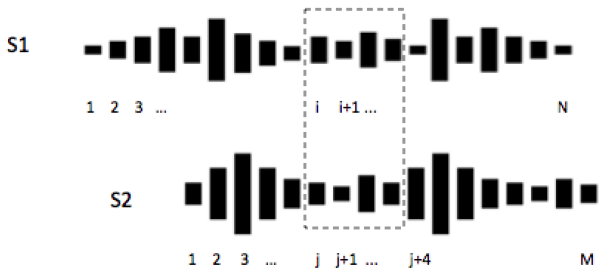

Figure 3. Illustration of the computation of the cost function over a four-beat sliding window (as in dashed rectangle) between beat i in segment S_1, and beat j in segment S_2.

Harmonic Progression Factor

After obtaining a set of candidate segments with acoustically smooth transition points from S_1, we determine which of these candidates yield a music theoretically valid harmonic progression when transitioning from S_1.

Given candidate beat b_1 in S_1 that connects to beat b_2 in S_2, we analyze the four-beat harmonic progression from $S_1[b_1-1:b_1]$ to $S_2[b_2:b_2+1]$, namely the last two beats of the transition point in S_1 and the first two beats in transition point in S_2, as well as the two-beat harmonic progression from $S_1[b_1]$ to $S_2[b_2]$. We assign a score Q_{b_1,b_2} as:

$$\begin{aligned} Q_{b_1,b_2} &= P_{5th}(S_1[b_1], S_2[b_2]) \\ &\quad + P_{pop}(S_1[b_1-1:b_1], S_2[b_2:b_2+1]) \end{aligned} \tag{2}$$

Where P_5th is the chord transition probability from the last beat of the transition point in S_1 to the first beat of the transition point in S_2 based on a circle of fifth, and P_{pop} is the four-beat harmonic transition probability from the last two beats of the transition point in S_1 to the first two beats of the transition point in S_2, trained from the SALAMI dataset [12].

We choose all segments with a score Q_{b_1,b_2} above a threshold as candidate segments that are displayed as gray boxes in the interface, as shown in Figure 2. The confidence value of a given segment is based on the score Q_{b_1,b_2} of that segment and mapped to an opacity value of the corresponding gray box in the interface.

CONCLUSION

We present medleyAssistant, an interactive music medley creation system that enables users to create personalized music medleys. Our informal pilot study showed that our interface makes medley creation significantly easier for novices. We believe that it could be a useful tool for experts as well as it could help them create medleys more quickly.

REFERENCES

1. Matthew EP Davies, Philippe Hamel, Kazuyoshi Yoshii, and Masataka Goto. 2014. AutoMashUpper: Automatic creation of multi-song music mashups. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22, 12 (2014), 1726–1737.
2. Tatsunori Hirai, Hironori Doi, and Shigeo Morishima. 2015. MusicMixer: Computer-aided DJ system based on an automatic song mixing. In *Proceedings of the 12th International Conference on Advances in Computer Entertainment Technology*. ACM, 41.
3. Eric J Humphrey, Douglas Turnbull, and Tom Collins. 2013. A brief review of creative MIR. In *Proceedings of the International Conference on Music Information Retrieval, Late Breaking Demo*.
4. Hiromi Ishizaki, Keiichiro Hoashi, and Yasuhiro Takishima. 2009. Full-Automatic DJ mixing system with optimal tempo adjustment based on measurement function of user discomfort. In *ISMIR*. 135–140.
5. Yin-Tzu Lin, I-Ting Liu, Jyh-Shing Roger Jang, and Ja-Ling Wu. 2015. Audio musical dice game: A user-preference-aware medley generating system. *TOMCCAP* 11 (2015), 52:1–52:24.
6. I-Ting Liu, Yin-Tzu Lin, and Ja-Ling Wu. 2013. Music Cut and Paste: A personalized musical medley generating system. In *ISMIR*. 463–468.
7. Matthias Mauch and Simon Dixon. 2010. Approximate Note Transcription for the Improved Identification of Difficult Chords. In *ISMIR*. 135–140.
8. Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. 2015. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th Python in Science Conference*. 18–25.
9. Markus Schedl, Emilia Gómez, Julián Urbano, and others. 2014. Music information retrieval: Recent developments and applications. *Foundations and Trends® in Information Retrieval* 8, 2-3 (2014), 127–261.
10. Aleksandr Scriabin and Leonid Sabaneev. 1913. Prométhée, le poème du feu pour grand orchestre et piano avec orgue, choeurs et clavier à lumières. Op. 60. Transcription pour 2 pianos à quatre mains par L. Sabaneiew. (1913).
11. Zhengshan Shi and Gautham J Mysore. 2018. LoopMaker: Automatic creation of music loops for pre-recorded music. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM.
12. Jordan Bennett Louis Smith, John Ashley Burgoyne, Ichiro Fujinaga, David De Roure, and J Stephen Downie. 2011. Design and creation of a large-scale database of structural annotations.. In *ISMIR*, Vol. 11. 555–560.