AUTOMATIC MASHUP CREATION BY CONSIDERING BOTH VERTICAL AND HORIZONTAL MASHABILITIES

Chuan-Lung Lee[1] **Yin-Tzu Lin**[1] **Zun-Ren Yao**[1]
Feng-Yi Lee[2] **Ja-Ling Wu**[1]
Communications and Multimedia Laboratory, National Taiwan University, Taiwan
[1]{kane0986,known,yyy110011,wjl}@cmlab.csie.ntu.edu.tw, [2]milkycc1111@gmail.com

ABSTRACT

In this paper, we proposed a system to effectively create music mashups – a kind of re-created music that is made by mixing parts of multiple existing music pieces. Unlike previous studies which merely generate mashups by overlaying music segments on one single base track, the proposed system creates mashups with multiple background (e.g. instrumental) and lead (e.g. vocal) track segments. So, besides the suitability between the vertically overlaid tracks (i.e. vertical mashability) used in previous studies, we proposed to further consider the suitability between the horizontally connected consecutive music segments (i.e. horizontal mashability) when searching for proper music segments to be combined. On the vertical side, two new factors: "harmonic change balance" and "volume weight" have been considered. On the horizontal side, the methods used in the studies of medley creation are incorporated. Combining vertical and horizontal mashabilities together, we defined four levels of mashability that may be encountered and found the proper solution to each of them. Subjective evaluations showed that the proposed four levels of mashability can appropriately reflect the degrees of listening enjoyment. Besides, by taking the newly proposed vertical mashability measurement into account, the improvement in user satisfaction is statistically significant.

1. INTRODUCTION

A Mashup is a kind of popular music what is made by overlaying, connecting, digitally modifying parts of two or more existing audio recordings [29]. The most common way to create a mashup is to overlay the vocal track of one song on the instrumental track of another [29]. With the aid of high-speed Internet, users are more easily to trade music materials and find related information through social websites [1, 3]. The development and availability of digital audio editing techniques and software also reduced the entry barrier for creating mashups. For example, the loop-based music sequencers such as Sony ACID Pro and Ableton Live make it easier for users to match the beats and shift the keys of the audio samples. As a result, mashups are now more often created by music lovers without formal musical training [29]. However, with the aforementioned tools, users still need to rely on their own experiences and musical training to find out proper music clips to be combined together. As the amount of currently available digital music explosively goes up, finding suitable clips becomes time-consuming and labor-intensive. How to automatically find out and create pleasant mashups becomes an challenging and interesting issue.

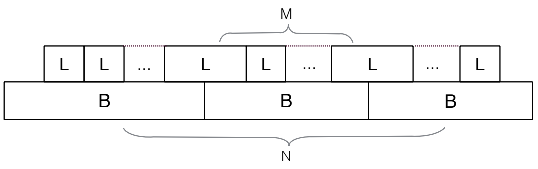

Figure 1. Common strucuture of mashup songs. Each block labeled with "L" or "B" represents a segment in lead track (e.g. vocal track) or the background track (e.g. instrumental track) from the same songs, respectively. M and N denote the number of lead track segments per background segment and the total number of background track segments in the resultant mashup, respectively.

Some previous studies have proposed automatic schemes to create mashups. But those approaches [8, 9] merely focused on the vertical suitability of the chosen music segments, that is, "they considered only about how suitable are the music segments to be overlaid", which was defined as the term "mashability" in [8]. By observation, many human made mashups [1] are not created just by overlaying different music segments on one single base track, as proposed in [9]. A Mashup can also be composed of segments of multiple background tracks (e.g. instrumental tracks) segments from different songs. Each background track segment is overlaid with several lead track segments (e.g. vocal tracks). As shown in Figure 1, when the background track segments changed, the lead tracks on top of them may still remain in the same song. So, while finding proper segments for generating mashups, we need to consider not only the vertical mashability between lead and background track segments but also the horizontal relation-

[1] https://www.youtube.com/watch?v=If5MF4wm1T8

ships between consecutive lead/background segments – we defined this relation as the "horizontal mashability".

In this work, a framework is proposed to automatically create mashups by considering both the vertical and the horizontal mashabilities. Besides, two additional factors: "harmonic change balance" and "volume weighting" to the vertical mashability are also considered and investigated. Subjective evaluation shows, by taking these factors into account, the users' listening pleasance of the created mashups will be enhanced as compared with that of the original counterparts created in [9]. Moreover, by integrating with the horizontal mashability, various degrees of listening enjoyment of mashups can be achieved. As a result, given a set of multitrack songs with structural segment labels, the first background track users want to extracted from and some desired structure factors (such as the number of background track segments N, and the number of lead track segments per background segment M), the system will then automatically generate a pleasant mashup with the structure as illustrated in Figure 1.

We assume that the multitrack songs should at least contain two kinds of tracks: background and lead. This assumption is reasonable because multitrack songs can be easily retrieved from mashup-related social websites [1,3]. The unit of input segments depends on the granularity of user specified song changes in a mashup. The unit could be as large as a structural section (e.g. verse, chorus), or be as small as a musical phrase (e.g. half or quarter of a verse), but we assume that all segment boundaries are aligned with bars. If users are not willing to provide segment boundaries, we can still detect the boundaries by using current structural segmentation techniques [14]. These input segments are regarded as the basic units to create mashups. To distinguish "input segment" from the generally used term "segment", in the rest of the paper, we will term the it as "unit". Therefore, in this paper, a lead unit stands for a segment in the lead track of an input song, and so on.

2. RELATED WORK

As compared with other music genres, mashup music is still young, so there is still a few academic studies focused on automatic mashup creation. Griffin et al. [13] proposed an efficient way to adjust the tempi of user-specified tracks and combine them after synchronizing their beats. The commercial software – Mixed in Key Mashup [2] uses the global harmonic compatibility among tracks in the users' music collection as the cue for track screening and provides tools to help users match the beats of the chosen tracks. In other words, users still need to find out proper segments in the chosen tracks by themselves. AutoMashupper [8, 9] is the first study that provided a thorough investigation on measurement for finding proper music segments to be overlaid together and an automatic mashup generation scheme. In AutoMashupper [9], an input song is regarded as a base track, and is segmented into short segments. For each segment, segments from other songs that are with the highest mashability–on the basis of chromagram similarity, rhythmic similarity, and spectral balance – will be overlaid with the corresponding segment in the base track to create the final mashup. The subsequent studies [7, 27] also followed this structure. In [7], a live input audio is regarded as the base track, and the accompanied music segments are overlaid upon the input audio. Tsuzuki et al. [27] focused on helping users overlay voices from different singers who had sung the same song along the common accompanied track. The proposed system, in contrast, is capable of creating mashups from multiple background and lead segments.

Besides mashup creation, there are other studies focused on mixing parts from existing music recordings, by means of concatenating instead of overlaying the music segments, such as the automatic DJ [6, 15] [16, p. 97-101], the medley creation systems [18, 20] and concatenative synthesis [4, 24] [16, p. 101-102,109-111]. The former two types of studies focused on concatenating longer audio segments such as phrases or sections, and studies of concatenative synthesis focused on audio snippets that are as short as musical notes/onsets. To select proper units to be concatenated, the existing systems may pick up proper candidates by comparing the similarity/distance between the candidates and the given unit according to various audio features (e.g. tempo, rhythm, pitch, harmonic, and timbre) [15, 16, 18], pre-cluster the all the units and then choosing among them according to some statistical models [4, 20, 24], or align them with user specified conditions [4, 6, 20]. For short units (e.g. notes), the units may be concatenated directly or accompanied with short cross-fade. For long units (e.g. sections or phrases), the above-mentioned systems may first decide the transition positions between consecutive music segments on the basis of rhythm [16] or chroma [18] similarity. And then, they adjusted the tempi (e.g. by phase vocoder [12]) and aligned the beats in the music segments with various methods and then concatenated the segments by cross-fading. In this study, the pre-described methods used to find proper segments and to smoothly connect them will be well-incorporated in the horizontal stage of the proposed system.

3. PROPOSED FRAMEWORK

The proposed system framework is illustrated in Figure 2. In the preprocessing step, the system will first extract audio features and pre-compute vertical and horizontal mashabilities for each possible pair of units in the given music set. Then, according user specified structure factors (e.g. the first song, the number of background track segments N, the number of lead track units per background segment M, etc.), we will determine ***(i)*** which and where the audio segment should locate in the resultant mashup – mashup composition ***(ii)*** how these segments are transformed to generate the resultant mashup. – mashup generation. In the "mashup composition" step, we will first pick M consecutive background units from the user specified song. We termed these units as a group of background unit (GBU). If users did not specify the first song, our system will randomly choose a GBU for them. Then, in the verti-

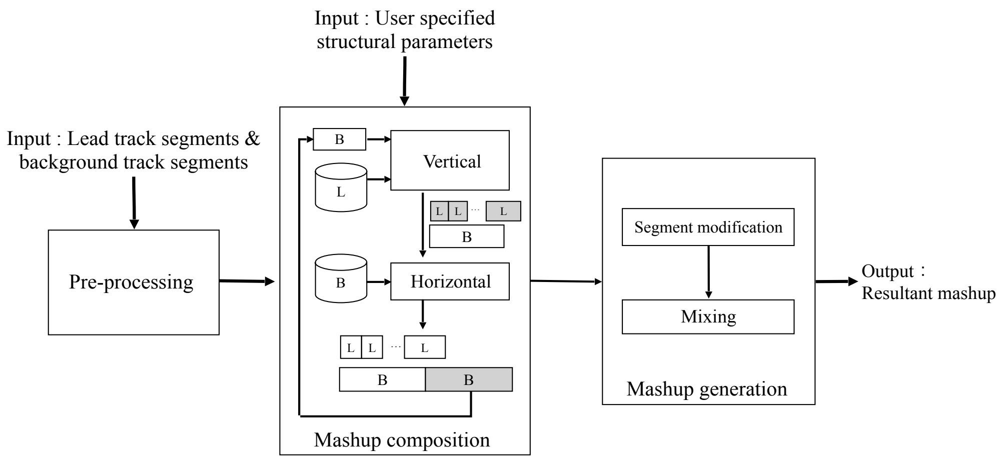
Input : User specified structural parameters
Input : Lead track segments & background track segments
Pre-processing
B
Vertical
L
L L ··· L
B
B
Horizontal
L L ··· L
B B
Mashup composition
Segment modification
Mixing
Mashup generation
Output : Resultant mashup

Figure 2. Proposed system Framework.

cal stage, we focus on finding proper lead units (the gray blocks marked with "L" in Figure 2) to be overlaid with the input GBU via vertical multiple mashabilities. After that, in the horizontal stage, we aim at finding a proper subsequent GBU (the gray block marked with "B" in Figure 2) by considering both vertical and horizontal mashabilities. The two processes, vertical and horizontal mashup stages, will be run iteratively until the resultant mashup reaches user desired length – the number of GBU N. Finally, in the mashup generation step, the tempo, loudness and pitch of each unit will be first modified to the desired values, and then the units will be mixed and concatenated to generate the final mashup song.

4. PREPROCESSING

In this step, the system will first extract audio features and pre-compute vertical and horizontal mashabilities for each possible pair of music units. The used features are beat/tempo [11], beat-synchronous chromagram [21], chord [22], MFCC [10], and volume [23]. For the ease of understanding, we will describe the used mashabilities, and how the above mentioned features are combined with each mashability in the following sections.

5. MASHUP COMPOSITION

In mashup composition, our system will determine which and where the basic units should locate. First, we will pick a GBU as our starting point (user specified or randomly picked by the system). The GBU should be with M consecutive background units in a song. Besides, all the background units in a GBU should contain exactly 2^{κ} beats, for $\kappa \in \mathbb{N}, \kappa \geq 2$. The reasons are ***(i)*** most popular songs are in 4/4 meter – 4 beats in a bar. ***(ii)*** most musical phrases in pop songs are multiples of four bars long [28]. ***(iii)*** most verse or chorus sections contain 2 to 4 phrases [28].

5.1 Vertical Stage

In the vertical stage, our system will find proper multiple lead units for each of the background unit in the input

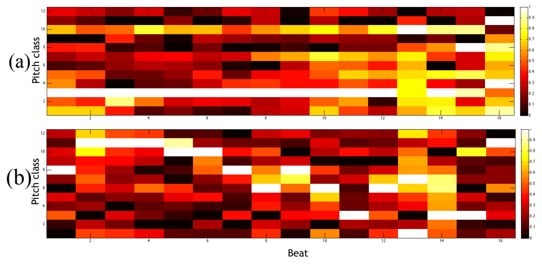

Figure 3. Chromagrams of the segment unit with (a) simple texture and (b) complex texture .

GBU based on vertical mashabilities. Mashabilities used in previous studies [9] include, harmonic matching, rhythmic matching and spectral balance. We do not use rhythmic matching in this stage because most lead tracks have no kick or snare sounds, so rhythmic pattern becomes unreliable in finding lead units. Spectral balance is also eliminated because the sounds in lead tracks often spread in the mid-band (220-1760 Hz), then spectral balance becomes indistinguishable. As a result, we adopt harmonic matching, and propose two new vertical mashabilities: harmonic change balance, and volume weighting.

5.1.1 Harmonic Matching

In harmonic matching part, we use a similar method to that of the AutoMashUpper [9]. The major difference is that we directly calculate the chroma similarity between each lead unit and background unit instead of shifting a window in the whole song. The reason is that the original method can not guarantee to get a complete lead unit. This may cause problems for the subsequent horizontal stage process, especially when it is the last unit in a GBU, because we will need to find consecutive lead units near GBU boundaries (please refer to Section 5.2 for details). The mashability score calculated by harmonic matching is denoted as S_c.

5.1.2 Harmonic Change Balance Weighting

Harmonic change balance is a newly proposed mashability. The idea comes from the observation that chroma similar-

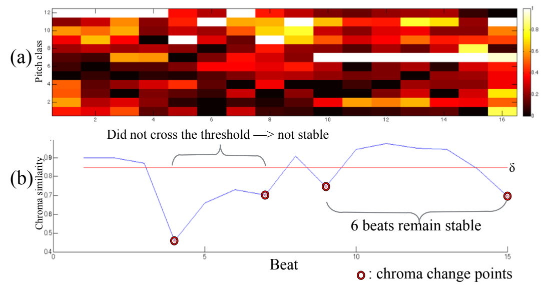

Figure 4. (a) Chromagrams of a segment unit and (b) the corresponding plot of chroma similarity between consecutive beats.

ities are not always proportional to the suitability for overlaying two segments. For instance, a given GBU composed of only one long note or long chord, the highest chroma-similar lead unit to it will highly probably be lead units that are also composed of the same texture. Then, the picked lead units for the GBU will all sound alike – long notes or long chords, which makes the resultant mashup sound boring and meaningless, even it sounds quite harmonic because of high chroma-similarity. As a result, we proposed to match the input unit to the one that is composed of opposite harmonic change rate, e.g., a background unit with simple texture (such as the chromagram illustrated in Figure 3(a)) should match with a lead unit with complex texture (c.g. Figure 3(b)), and vice versa. The harmonic change rate can be calculated according to how many beats remain stable on the chroma in a unit. Figure 4 illustrates the chromagram and the chroma similarities between consecutive beats in a unit. The local minima of the chroma similarity plot below the threshold δ can be defined as the chroma change points. Then, the beats lie between any two change points and contain exactly two crossing points to δ are regarded as stable beats. The percentage of stable beats will be mapped to a sigmoid function to get a smooth score from 0 to 1 (0% stable beat is mapped to 1 while, 100% stable beats are mapped to 0), i.e. the harmonic change rate ξ. Then, the harmonic change balance weights w_t can be calculated as:

$$w_t = 1 - |\xi_p - (1 - \xi_q)|\,, \tag{1}$$

where ξ_p and ξ_q are the harmonic change rate of units p and q, respectively. If harmonic change rate of a background unit is 0.7, we tend to find a lead unit whose harmonic change rate is closer to 0.3.

5.1.3 Volume Weighting

There are many inaudible (less than -40db) lead units in a lead track because the lead vocal or instruments often rest in sections such as intro, intermezzo, and outro. To eliminate lead units contain too many inaudible parts, we included the volume weighting w_v in our vertical mashability computation. w_v can be calculated according to the portion of the lead units that can be heard. That is,

$$w_v = \begin{cases} 1 & , \quad if\ a \geq \frac{1}{2}\eta \\ \frac{1}{2} + \frac{a}{\eta} & , \quad if\ a < \frac{1}{2}\eta, \end{cases} \tag{2}$$

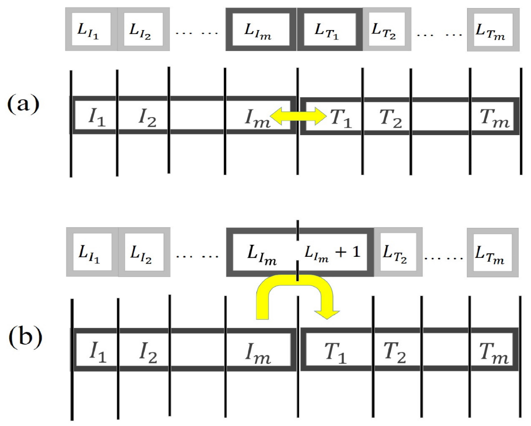

Figure 5. Schematic diagrams showing how to generate mashups by considering (a) horizontal mashability and (b) vertical mashability, respectively.

where a and η are the numbers of audible and total beats in a unit, respectively.

Finally, we combine the aforementioned measurements together to find the final vertical mashability S_v, that is

$$S_v = S_c \cdot w_t \cdot w_v + w_\tau, \tag{3}$$

where w_τ is an additional bonus to the pair of units with close tempo, it is similar to the parameter α adopted in Eqn. (9) of [9].

5.2 Horizontal Stage

In this stage, we aim at finding a proper subsequent GBU **T** for the input GBU **I** by considering both vertical and horizontal mashabilities. A perfect subsequent GBU **T** should satisfy two properties: ***(i)*** it can smoothly be concatenated with the previous GBU **I** (cf. Figure 5 (a)), and ***(ii)*** one can find a proper leader unit on top of the first background unit in this GBU **T** and the found leader unit can be smoothly concatenated with the previous leader unit (cf. Figure 5 (b)). To achieve property (i), we incorporated an approach similar to the concept described in [20, Sec. 7.2] and [19, Sec. 4.1]. We adopted the same similarity measurements and weights as [20] to compute the similarity between the GBU subsequent to **I** in the original track and all the candidate GBUs, which was defined as the horizontal mashability S_h. Then, sort according to S_h, we can get a rank list, R_h. A threshold α is applied to cut off the rank list: the GBUs with S_h that are lower than α are eliminated. For dealing with the pre-described property (ii), we take the opposite direction. That is, we first check if the next unit in the original track of the lead unit L_{I_m} exists. If it is, we will temporally choose it as the first lead unit for GBU **T**. Then, we can get another rank list R_v of GBUs by sorting the vertical mashabilities S_v between the first background units of the GBUs and the picked lead unit. A similar threshold β is also applied to the rank list to eliminate inappropriate GBUs. After the above steps, we may encounter four cases in the transitions between two GBUs.

Case 1. Both R_h and R_v exist, and $\{R_h \cap R_v\} \neq \varnothing$. This is the perfect case. Then, we can pick the first GBU in $\{R_h \cap R_v\}$ as our result.

Case 2. Only R_v exists. We pick the first GBU in R_v. In case 2, the two background units at the transition have no correlation but they are bridged via the lead units taken from the same song on top of them.

Case 3. The opposite of case 2. Only R_h exists. We choose the first GBU from R_h. In this situation, two background units at the transition have high correlation but the lead units on top of them cannot stay in the same track.

Case 4. Both R_h and R_v do not exist. We randomly choose a GBU. In case 4, the two background units have no correlation and the lead units on top of them cannot stay in the same track. We can also provide an optional self-repairing mechanism for case 4. That is, instead of random selection, we choose a GBU that its next transition will fit the condition of case 1 via pre-computation of all the possible cases of all the GBUs in the collection.

A more complex situation is that, both R_h and R_v exist, but $\{R_h \cap R_v\} = \varnothing$. Which rank list should we choose from? According to the user evaluation results in Section 7.2, most users prefer case 2 than case 3. So we will choose a GBU from R_v first.

6. MASHUP GENERATION

Mashup generation can be divided into two steps: segment modification and mixing. In segment modification, we first shift the pitches of the lead units to a target key, found in the harmonic matching step (Section 5.1.1). The same as [9], we use Rubberband library [5] to shift the pitches. Then, the volume of the lead units are also re-scaled to match that of the background unit by Replay Gain [23]. After that, to match to beats of the units, we apply phase vocoder [12] to stretch the beats. Finally, we extend all of the units one beat long and apply cross fade technique to all of the transitions to create the resultant mashup.

7. EXPERIMENT: SETTINGS AND RESULTS

We conducted two subjective listening tests. The first test is to evaluate the impact and the user's acceptability of the four approaches, we proposed to deal with various transition conditions and also find the proper-connecting priority of these four cases. The second test is to compare the compatibilities of lead units which are provided by the mashability in AutoMashUpper [9] and by the vertical mashability in our system. The generated mashups can be found in `http://cmlab.csie.ntu.edu.tw/~kane0986/ISMIR2015.html`.

7.1 Dataset

We use the multi-track audio dataset in [14] with structural segment labels. The dataset contains 104 pop songs, each song contains about 5 tracks on average. In the experiment, for each song, we take the lead vocal track as the lead track, and then we mix all the rest tracks into a single track and regard it as the background track. Examples of background tracks are drum and bass tracks or chordal instruments such as piano, guitar, or string. The vocal chorus track is eliminated because it has different properties to either the lead or the background track. We take 0.8478 as the threshold δ in the harmonic change balance weighting step. The threshold is obtained by finding the intersection of distribution of the chroma similarity values of consecutive beats in 73 simple textured units and 156 complex units. The other two thresholds, α and β we used in the horizontal stage are set to 0.6422 and 0.5740, respectively. These two thresholds are obtained through observations on the first derivatives of the sorted scores of all the unit pairs.

7.2 Subjective Evaluations on Horizontal Mashability

In the first test, given the same background unit $\mathbf{B}$ and lead units on top of $\mathbf{B}$ as inputs, we then got four mashups which have dedicated configurations as those of pre-described case 1 to case 4, respectively. We also added the original track of unit $\mathbf{B}$, which has no transition, as a reference, and is denoted as case 0. The user evaluations are conducted through the aid of a web interface, and the tested mashups are presented in random order. For each participant, he or she needs to listen five groups of mashups, and each group contains five mashups which respect to the five cases we mentioned above. The questionnaires are designed based on a 7-point Likert scale [17]. Users are asked to report their opinions about the degrees of enjoyment of the mashups from the following options: very pleasing (7), pleasing (6), somewhat pleasing (5), neutral (4), not so pleasing (3), not pleasing (2), and very unpleasing (1). 21 males and 6 females aged around 20∼60 participated in this test. All of our participants have listening test experience, but most of them are not major in music (less than five participants have educational background in music) since our target consumer is general public.

Figure 6 shows the mean opinion score of each case. The paired Wilcoxon signed rank test [25] is applied to analyze the results, where the corresponding p-values are reported in Figure 6. The overall result shows that the four proposed cases did impact the human feeling of the resultant mashups under a confidence level of 95% [2]. Case 1 is rated second only to case 0. The score of case 2 is lower than case 1, but commonly higher than case 3. This indicates that users commonly prefer case 2 to case 3, i.e., bridging two GBUs with no relation via one lead track is more acceptable than concatenating two GBUs with high correlation but with the lead units do not stay in the same track. Case 3 is rated higher than case 4 commonly, but not as significant as other cases. Case 4 gets the lowest score generally, this verifies that when there is significant change in both of the lead and the background transitions, a great impact to user's acceptability will result.

[2] By Bonferroni correction, to preserve the total confidence level as 95 %, the p value for each paired comparison should be $< \frac{0.05}{C_2^5} = 0.005$

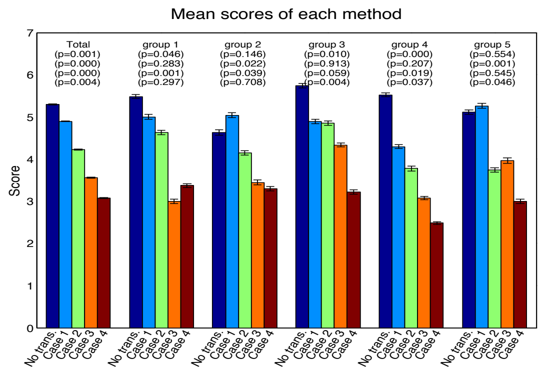

Figure 6. Mean opinion scores of total and each test sample, in which the relevant p-values of paired Wilcoxon signed rank test [25] on "case 0 (no transition) vs. case 1", "case 1 vs. case 2", "case 2 vs. case 3", and "case 3 vs. case 4" are displayed above the corresponding bars of each one of the experiments, respectively.

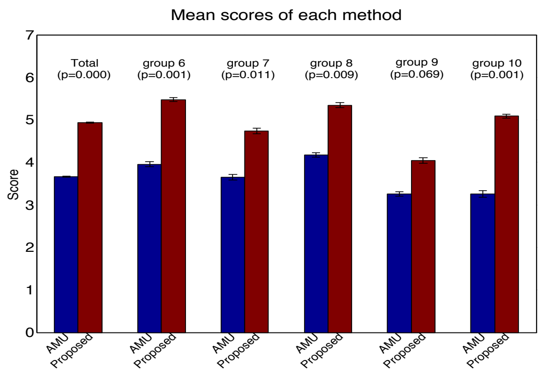

Figure 7. Mean opinion scores of total and each test sample, in which the relevant p-values of paired Wilcoxon signed rank test [25] on "AMU vs. our system" are displayed above the corresponding bars of each experiment.

Counter-intuitively, there are two groups in our listening test showing that case 1 is rated higher than case 0 slightly though they did not statistically significant. Possible reason would be that the two GBUs in case 0 happen to be from verse and chorus sections of different styles, respectively. Then our system may have chance to find another background unit which can be concatenated after the verse segment more smoothly than it's own chorus counterpart.

7.3 Subjective Evaluations on Vertical Mashability

In the second test, we aim at comparing the vertical mashability provided by our system and the AutoMashUpper [9] (denoted as AMU). We create the mashups by our method and AMU's method from the same input GBU **I** of 6 background units. Besides, we force the chosen lead units to be picked from different songs. As we mentioned in Section 5.1, rhythmic matching and spectral balance are not reliable for the current dataset, so we only used the harmonic matching part in AMU[3], i.e. the version in [8]. Then, it is easily to pick lead units that are nearly inaudible since only harmonic matching is considered in the adopted AMU version. To make a fair test, we also apply our volume weighting (Section 5.1.3) to AMU. As a result, the target component we compared here is the harmonic change balance weighting. A similar evaluation procedure to the previous experiment was conducted. Users are invited to listen to five groups of mashups per time, and each group has two mashups – generated by AMU and by our systems, in random order. 18 males and 6 females aged around 20∼60 with similar background to the previous experiment participated in this test. The result of this test is given in Figure 7, and the corresponding p-values are also reported. The lead units generated by our system are commonly rated higher than those created by AMU, under a confidence level of 95%. This again verified the advantage of taking the harmonic change balance weighting into consideration. In fact, most of the GBUs are simple textured. So the lead units generated by AMU is more likely to pick simple textured lead units.

[3] We implemented AMU's methods by ourselves.

8. CONCLUTION AND FUTURE WORK

In this paper, a novel system is proposed to effectively create music mashups. There are two main contributions done in our system. First, both vertical and horizontal mashabilities are taken into consideration. Through this, our system can create a mashup with multiple background and lead track segments, which provides much higher flexibility in making mashups than the systems proposed in previous studies. Second, by taking the newly proposed vertical mashability measurement into account, user study shows that the improvement in user satisfaction is statistically significant. The subjective evaluations also show that the four concatenation cases we analyzed play a critical role in generating enjoyable mashups.

Many aspects of our system can be extended. First, in the vertical stage, we could alternatively match the unit based on the compatibility of pitch of lead unit and the chord of the background unit [26] instead of the chroma similarity between the lead and the background units directly. Second, sometimes we found that the lead units chosen by our system are too different from one another, so that the created mashups would sound very abrupt. To prevent this situation, we may restrict the chosen lead units to be with certain characteristics analyzed in advance, e.g. timbre, style, and emotion. Finally, we could further investigate the effect of overlapping the lead units and the chorus track units. Even more, the background units can also be separated into instrumental track units and drum track units, etc.. Toward the study about how to combine all kinds of units reasonably may provide true solutions to create music mashups in all conditions.

9. ACKNOWLEDGMENTS

The authors are grateful to all the participants (CMLab DSP group members) who helped evaluate the results.

10. REFERENCES

[1] DJ Mix Generator. `http://www.djprince.no/site/DMG.aspx`.

[2] Mixed in Key. Mashup. `http://mashup.mixedinkey.com/HowTo`.

[3] Mixter. `http://ccmixter.org`.

[4] Gilberto Bernardes. *Composing Music by Selection: Content-Based Algorithmic-Assisted Audio Composition*. Phd thesis, University of Porto, 2014.

[5] C. Cannam. Rubber Band Audio Time Stretcher Library, 2012. `http://breakfastquay.com/rubberband/`.

[6] Dave Cliff. Hang the DJ : Automatic Sequencing and Seamless Mixing of Dance-Music Tracks. Technical report, HP Labs, 2000.

[7] Matthew Davies, Adam Stark, Fabien Gouyon, and Masataka Goto. Improvasher: A Real-Time Mashup System for Live Musical Input. In *Proc. NIME*, pages 541–544, 2014.

[8] Matthew E P Davies, Philippe Hamel, Kazuyoshi Yoshii, and Masataka Goto. AutoMashUpper : An Automatic Multi-Song Mashup System. In *Proc. ISMIR*, Curitiba, PR, Brazil, 2013.

[9] Matthew E. P. Davies, Philippe Hamel, Kazuyoshi Yoshii, and Masataka Goto. AutoMashUpper: Automatic Creation of Multi-Song Music Mashups. *IEEE Trans. ASLP*, 22(12):1726–1737, December 2014.

[10] Steven B. Davis and Paul Mermelstein. Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. *IEEE Trans. ASLP*, 28(4):357–366, 1980.

[11] Simon Dixon. Evaluation of the Audio Beat Tracking System BeatRoot. *J. New Music Res.*, 36(1):39–50, 2007.

[12] Mark Dolson. The Phase Vocoder: a Tutorial. *Computer Music Journal*, 10(4):14–27, 1986.

[13] Garth Griffin, YE Kim, and Douglas Turnbull. Beat-Sync-Mash-Coder: a Web Application for Real-Time Creation of Beat-Synchronous Music Mashups. In *Proc. ICASSP*, pages 2–5, Dallas, Texas, USA, 2010.

[14] Steven Hargreaves, Anssi Klapuri, and Mark Sandler. Structural Segmentation of Multitrack Audio. *IEEE Trans. ASLP*, 20(10):2637–2647, 2012.

[15] Hiromi Ishizaki, Keiichiro Hoashi, and Yasuhiro Takishima. Full-Automatic DJ Mixing System with Optimal Tempo Adjustment based on Measurement Function of User Discomfort. In *Proc. of ISMIR*, pages 135–140, Kobe, Japan, 2009.

[16] Tristan Jehan. *Creating Music by Listening*. Phd dissertation, Massachusetts Institute of Technology, 2005.

[17] Rensis Likert. A Technique for the Measurement of Attitudes. *Archives of Psychology*, 22(140):1–55, 1932.

[18] 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 *Proc. ISMIR*, Kobe, 2009.

[19] Yin-Tzu Lin, Chuan-Lung Lee, Jyh-Shing Roger Jang, and Ja-Ling Wu. Bridging Music via Sound Effect Insertion. *IEEE Multimedia*. (to appear).

[20] Yin-Tzu Lin, I-Ting Liu, Jyh-Shing Roger Jang, and Ja-Ling Wu. Audio Musical Dice Game: A User-preference-aware Medley Generating System. *ACM TOMM*, 11(4), 2015.

[21] Matthias Mauch and Simon Dixon. Approximate Note Transcription for the Improved Identification of Difficult Chords. In *Proc. ISMIR*, pages 135–140, 2010.

[22] Yizhao Ni, Matt McVicar, Paul Santos-Rodriguez, and Tijl De Bie. An End-to-End Machine Learning System for Harmonic Analysis of Music. *IEEE Trans. ASLP*, 20(6):1771–1783, 2012.

[23] D. Robinson. *Perceptual Model for Assessment of Coded Audio*. PhD thesis, University of Essex, 2002.

[24] Diemo Schwarz. *Data-driven concatenative sound synthesis*. Phd thesis, University of Paris 6 – Pierre et Marie Curie, 2004.

[25] S. Siegel and N.J. Castellan. *Nonparametric statistics for the behavioral sciences*. McGraw-Hill, Inc., 1956.

[26] Ian Simon, Dan Morris, and Sumit Basu. MySong: Automatic Accompaniment Generation for Vocal Melodies. In *Proc. SIGCHI*, pages 725–734, 2008.

[27] Keita Tsuzuki, Tomoyasu Nakano, Masataka Goto, Takeshi Yamada, and Shoji Makino. Unisoner : An Interactive Interface for Derivative Chorus Creation from Various Singing Voices on the Web. In *Proc. ICMC*, number September, pages 790–797, 2014.

[28] Stephen Webber. *DJ Skills: The Essential Guide to Mixing and Scratching*. Focal Press, 2007.

[29] Melissa Hok Cee Wong. Mash-up. In Charles Hiroshi Garrett, editor, *The Grove Dictionary of American Music*. 2 edition, 2013.