Semi-Automatic Synthesis of Videos of Performers Appearing to Play User-Specified Music

Tomohiro Yamamoto
UEC
yamamoto@onailab.com

Makoto Okabe
UEC/JST PRESTO
m.o@acm.org

Yusuke Hijikata
UEC
hijikata@onailab.com

Rikio Onai
UEC
onai@cs.uec.ac.jp

2-11, Fujimicho, Chofu, 182-0033, Tokyo, Japan

ABSTRACT

We propose a method to synthesize the video of a user-specified band music, in which the performers appear to play it nicely. Given the music and videos of the band members as inputs, our system synthesizes the resulting video by semi-automatically cutting and concatenating the videos temporarily so that these synchronize to the music. To compute the synchronization between music and video, we analyze the timings of the musical notes of them, which we estimate from the audio signals by applying techniques including short-time Fourier transform (STFT), image processing, and sound source separation. Our video retrieval technique then uses the estimated timings of musical notes as the feature vector. To efficiently retrieve a part of the video that matches to a part of the music, we develop a novel feature matching technique more suitable for our feature vector than dynamic-time warping (DTW) algorithm. The output of our system is the project file of Adobe After Effects, on which the user can further refine the result interactively. In our experiment, we recorded videos of performances of playing the violin, piano, guitar, bass and drums. Each video is recorded independently for each instrument. We demonstrate that our system helps the non-expert performers who cannot play the music well to synthesize its performance videos. We also present that, given an arbitrary music as input, our system can synthesize its performance video by semi-automatically cutting and pasting existing videos.

Keywords
Video Synthesis Music Analysis Multimedia

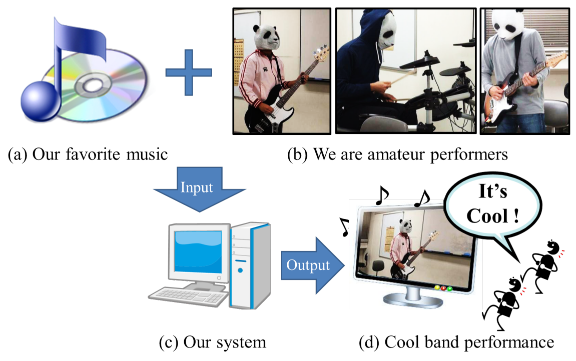

Figure 1: We want to synthesize the video of our favorite music (a) played by our band members. We have recorded the performance videos of our band members (b), but they were not perfect. So, we developed a system (c) that takes the music and the videos as inputs, and synthesizes the video composite (d) in which all the members appear to play the music nicely.

1 INTRODUCTION

Synchronization between sound and footage is an important task when producing a high-quality movie. For this task, the designers must spend a lot of time 1) to add sound effects or music to a silent footage or 2) to edit the footage so that it synchronizes with the given sounds or music. In this paper, we focus on this second demand, i.e., we develop a system to help the designer, who creates a musical performance video, to efficiently synthesize it.

Today, more and more people have become interested in synthesizing an original video using their favorite music. Especially, in video-sharing services such as YouTube, we can find a lot of music videos created by the professional and amateur designers, which a lot of people enjoy every day. Note that watching the music video is the totally different experience from only listening to the music. For example, we can tend to pay more attention to the sounds of the currently watching instrument: watching the bass performance allows us to more easily capture the low-pitched bass sounds. It is also reported that the movie affects how we, audiences, feel the rhythm and moods of the music [10].

As related researches, there are several methods that synthesize videos synchronizing to user-specified music, e.g., the summarization of home videos based on a user-specified music [4], the creation of the slide-show video of photographs [6], or the dance performance videos using 3D human models [5, 11] or the 2D performance videos [8]. However, since all of these

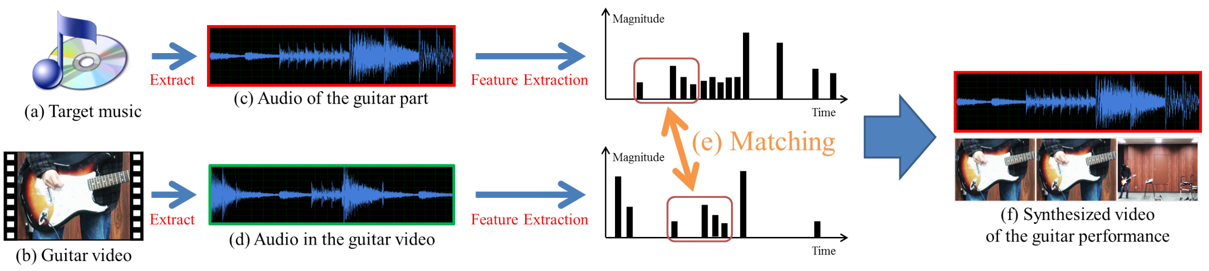

Figure 2: The overview of synthesizing a performance video of a single instrument (Guitar, here).

approaches analyze only the tempos and moods of the input music, we cannot apply these techniques to our problem: since we want to synthesize a performance video in which the performer looks like playing the music, we require finer synchronization.

Given a user-specified music and videos of performers, our method semi-automatically synthesizes the music video. We usually record the video for each instrument independently and then create the video database. We then estimate the timings of the musical notes by analyzing the audio signals of the music and video database. Using the estimated timings as the feature vector, our system retrieves the candidates of footages for each part of the input music. The output is the project file of Adobe After Effect that has multiple candidates, in which the user can further edit the result.

We asked the two types of performers to join in our experiments. In the first experiment, we asked amateur performers who can play each instrument well. We recorded the performances of playing viola, drums, and bass. In the second experiment, our performers are novices, i.e., who didn't know how to play each instrument at first: we then asked them to practice each instrument for one hour. In each case, we demonstrate that our system can synthesize a reasonable performance video in short editing times.

2 SYSTEM OVERVIEW

Our system consists of 2 parts: the first part synthesizes a solo performance video of a single instrument (Fig. 2), and the second part synthesizes the band performance video by mixing all the synthesized solo videos (Fig. 3). In Fig. 2, The user begins to synthesize a solo performance video by specifying a favorite music (Fig. 2-a). If the input music includes multiple instruments, we first separate it into solo audio signals using sound source separation technique (Sec. 3.2).

Here, we show the guitar signal of the input music (Fig. 2-c), and the video of playing the guitar (Fig. 2-d). We then apply our technique of extracting the timings of musical notes (Figs. 2-e and f). In each feature vector, the peak position corresponds to the estimated timing of a musical note, and the distance from it to the next peak corresponds to the duration of the note.

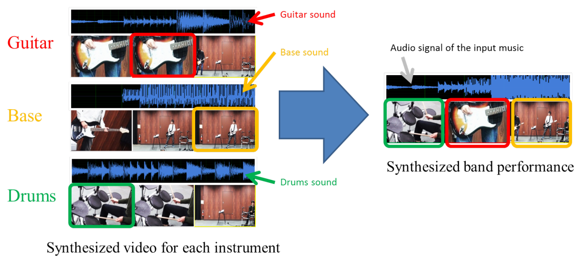
Guitar
Guitar sound
Base
Base sound
Drums
Drums sound
Audio signal of the input music
Synthesized band performance
Synthesized video for each instrument

Figure 3: To synthesize the band performance consisting of multiple instruments, our system selects the adequate part from each solo performance video and concatenate them. Here, we selects red, orange, and green parts and concatenate them.

The system then computes the part-by-part matching between the feature vectors of the input music and the video (Fig. 2-e). Finally, We synthesize the solo performance video by concatenating the videos based on the matches (Fig. 2-f). After synthesizing solo performance videos for each instrument, our second part synthesizes the band performance video by further cutting and concatenating them (Fig. 3).

3 FEATURE EXTRACTION

In this section, we describe the feature extraction, i.e., the process of estimating the timings of musical notes from the audio signal of a solo performance (Sec. 3.1). When the audio signal has sounds of multiple instruments, the sound source separation is required, which separates an audio signal of multiple instruments into audio signals of each single instrument (Sec. 3.2).

3.1 Feature Extraction on Solo Performance

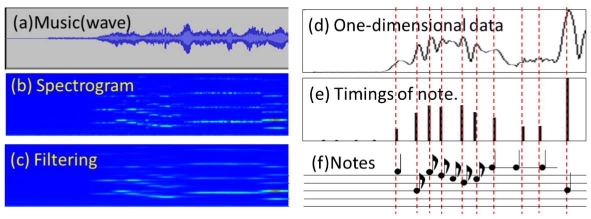

Figure 4: The extraction of feature vector.

To extract the feature vector from an audio signal, we develop a simple technique of onset detection. Our idea is similar to [1].We extract the feature vector from input music and video database.

To estimate the timings of the musical notes, our method begins to compute short time Fourier transform (STFT) to the audio signal (Fig. 4-a), and obtains the spectrogram (Fig. 4-b). Since the spectrogram is usually noisy, we smooth it out but preserve strong edges that correspond to the beginnings and endings of each musical note. For this purpose, we apply a horizontal bilateral filter [14] to obtain the smoothed spectrogram (Fig. 4-c). We differentiate the smoothed spectrogram horizontally to extract the beginning and end of each musical note. We then integrate the spectrogram vertically to obtain one dimensional (1D) signal (Fig. 4-d). Finally, we extract the feature vector by finding local maxima of the 1D signal (Fig. 4-e). We show the score of this audio signal that the performer actually played in Fig. 4-f. Note that the peaks match the timings of the score.

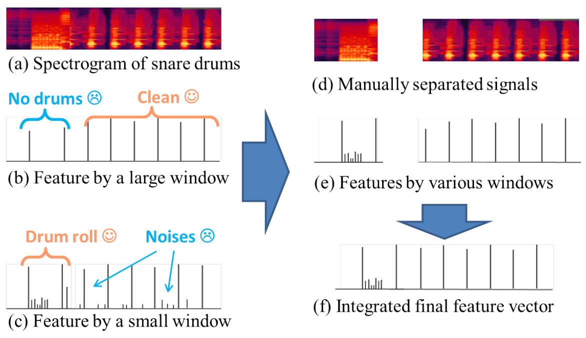
(a) Spectrogram of snare drums
No drums ☹
Clean ☺
(b) Feature by a large window
Drum roll ☺
Noises ☹
(c) Feature by a small window
(d) Manually separated signals
(e) Features by various windows
(f) Integrated final feature vector

Figure 5: The technique to extract the feature vector using multiple bilateral filters with various window sizes.

To extract high quality feature vectors, we must choose an adequate window size for the bilateral filter. For example, Fig. 5 shows an audio signal of drums, which starts by the high-frequent drum roll. A large window produces clean peaks but makes the peaks around the drum roll disappear. On the other hand, a small window produces the peaks of the drum roll well but also too many small peaks that are noises. So, we need to change the window size adaptively through the signal. However, since it was difficult to implement such a smart adaptive filter, we decided to manually cut the signal into segments based on the observed frequencies, and apply the adequate bilateral filter to each of them. This process increases the user's burden but is important to create the high quality feature vector.

We also use the sound volume values as our additional feature vector. We use volume information to detect whether the instrument is being played or silent: if it is silent, we don't assign any footage of the instrument to that part. Our feature vector of the volume is binary, i.e., 1 (there is sounds) or 0 (silent).

We also use the changes of pitch values as our additional feature vector. To estimate the pitches of an audio signal, we use STRAIGHT [7] to estimate the fundamental frequency (F0). Since we are interested not in the absolute pitch values but only in the relative changes of pitch values, we use the differentiation of F0 as our feature vector. This feature vector helps to synthesize the reasonable performance video by preventing from assigning a footage of descending pitch to that part of ascending pitch.

3.2 Sound Source Separation

The algorithm of feature extraction described above works well for an audio signal of a single instrument. However, it is often difficult to prepare the audio track for each instrument independently. For example, the drum set that we used to record the drummer's performance gives us only the mixture of sounds of four drums, i.e., snare drum, bass drum, cymbals, and hi-hats. In such a case, we want to separate the audio track for each instrument. For this purpose, we use the semi-automatic sound source separation technique. Fig. 6 shows an example of the separation of the drum audio signal.

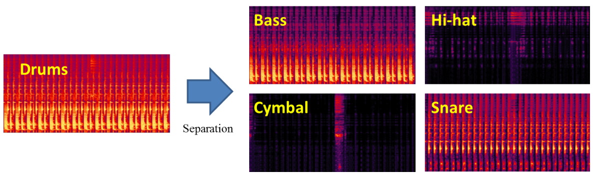

Figure 6: Sound source separation of drums.

We use the probabilistic latent component analysis (pLCA) [12], which is developed based on pLSA algorithm [2]. pLCA is the algorithm for non-negative matrix factorization that separates a spectrogram $S(t,f)$ as follows:

$$S(t,f)=\sum_{i}^{N}B_i(t,f), \tag{1}$$

where $B_i(t,f)$ is non-negative, t represents the time, f represents the frequency, and N is the number of the bases. Fig. 7 shows an example of the separation of an audio signal: the original spectrogram having the violin and drums sounds is separated into the twelve bases at first. We then manually classify them into two classes in this case, i.e., violin class and drums class. For the violin class, the user's selection is visualized as the green boxes. We sum these bases to obtain the spectrogram of the violin. These separated sounds are demonstrated in the supplementary video.

4 VIDEO RETRIEVAL BY FEATURE MATCHING

For each part of the input music, we retrieve a footage so that the feature vector of the part of the input music

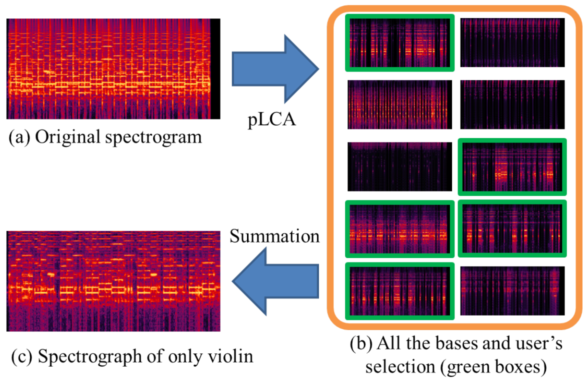
(a) Original spectrogram
pLCA
Summation
(c) Spectrograph of only violin
(b) All the bases and user's selection (green boxes)

Figure 7: Sound source separation by pLCA.

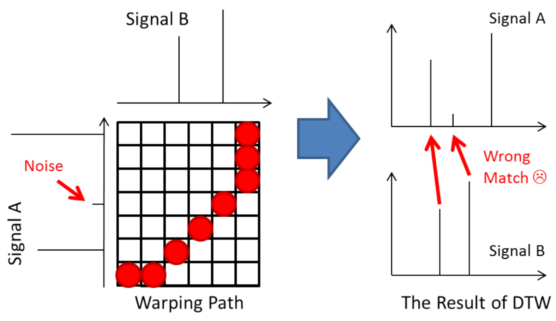

Figure 8: DTW algorithm is not suitable for our case, where the signal consists of peaks.

matches to the feature vector of the footage. However, it is unusual that a pair of feature vectors match exactly, and if we consider only such exact matches, we cannot retrieve any footage for most parts of the input music. So, we consider not only exact matches but also similar matches. Fig. 9 shows some examples: each of red, green, and blue boxes show the match of feature vectors. The peaks of the same number are matched, but the timings of them are slightly different. We complement these differences by changing the playback speed of the footage in the video synthesis process.

4.1 Our Algorithm

As a method for matching the signal while stretching, dynamic time warping (DTW) is well-known [9]. DTW works well for computing similarities between smooth, continuous signals, but does not work for our case, i.e., each signal consists of discontinuous peaks. More precisely, DTW works well for our case, only when the number of peaks is the same between the comparing signals; however, since the method of feature extraction described above is not always perfect, it often produces noises. Fig. 8 shows an example. Signals A and B share two peaks of similar magnitudes, but signal A has a small peak as noise between them. When we compute the warping path of DTW, it becomes as the path of red circles, i.e., it makes the second peak of signal B match to the noisy peak. Based on this observation, we concluded that DTW is not suitable for the comparison of peaky signals.

To solve the problem, we propose a simple technique to efficiently retrieve a footage, allowing the temporal stretching. Our idea is inspired by the RANSAC algorithm [3] used in many computer vision applications, which efficiently finds the set of matches that are consistently explained by a transformation.

Fig. 9 shows how our algorithm finds a local match. The red line of the first step shows 1-to-1 match of peaks: In this step, we find such a 1-to-1 match by simply comparing the magnitudes of peaks using a threshold parameter, α. In the second step, we check the neighboring peaks: here, peaks '3' and 'C' can be a match, because the distance between '3' and '1' and the distance between 'C' and 'A' are similar. The same thing can be said for the match of '2' and 'B'. In the third step, we further check the neighboring peaks: here, peaks '4' and 'D' can be a match because the distances are similar. However, peaks '5' and 'E' cannot be a match: the distance between '5' and '3' are too smaller than the distance between 'E' and 'C'. In the next step, we further check the neighbors toward right-hand side, but we don't check the neighbors toward left-hand side anymore. We continue this process until the difference of the distances becomes larger than a threshold parameter, β. We apply this algorithm to all the possible 1-to-1 matches between the input music and the video database.

Our algorithm solves the weakness of DTW, and inherits the strength of DTW at the same time. The weakness of DTW is that it is too sensitive to noises as shown in Fig. 8. However, in our algorithm, we can prevent a healthy peak from matching to such a small noisy peak by appropriately specifying the threshold α. On the other hand, the strength of DTW is that it can take the temporal stretch into account, which can be achieved by appropriately specifying the threshold β. Fig. 9-right shows the three resulting local matches of blue, red, and green boxes. For example, in the blue match, the distances of features of the input music are larger than the distances of features of the video database, which means that we can fit the video by making the playback speed slower. In the green match, we make the playback speed faster and the video would fit to the part of the input music.

5 SYNTHESIS OF BAND PERFORMANCE VIDEO

Given the input music and a set of videos of playing each instrument, the methods described above extract the feature vectors, retrieve an appropriate footage and assign it to each part of the input music. Each part of the input music usually has multiple candidates of footages. Our system saves this result of retrieval and assignment

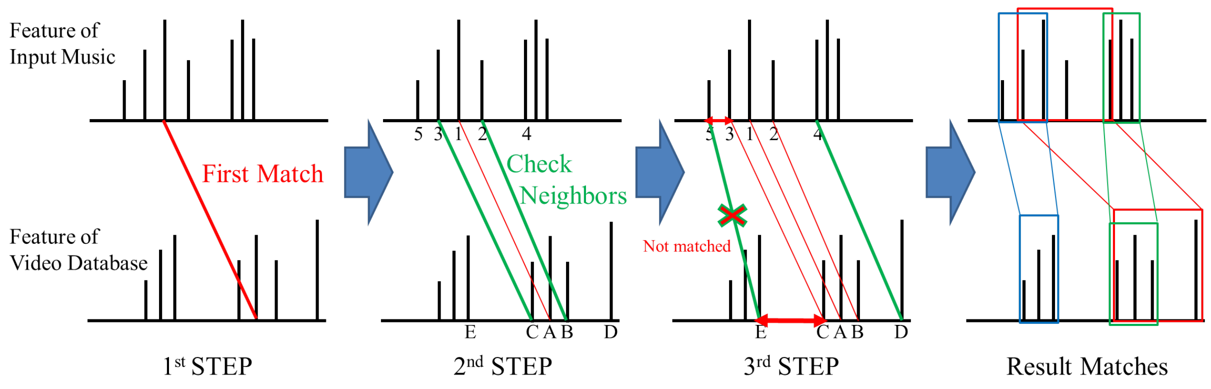

Figure 9: Our algorithm to find local matches.

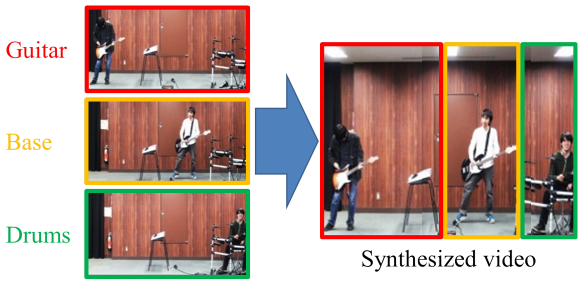

Figure 10: Masking and lapping the solo performance videos to synthesize the band performance video.

as the project file of Adobe After Effects: the user can interactively select the best set of footages from the candidates and render the final movie. The temporal stretch of each footage is easily achieved by the time remapping function of Adobe After Effects: when generating the project file, our system automatically specifying the parameters of this function for each candidate footage.

This editing process is usually as shown in Fig. 3: since our method assumes that each video is recorded for each instrument independently, we cannot synthesize a scene where all the band members appear at the same time. One idea to synthesize such a scene is to record each performer also from a fixed camera as shown in Fig. 10. Each video has just a solo performance, but we can synthesize all the band members by spatially masking and lapping all the videos into the resulting video.

5.1 Automation of Candidate Selection

Manually selecting the best footage for all the parts of the input music is often hard work and time-consuming. To reduce this user's burden, we develop an automatic method to support this process. We consider this as the problem of label assignment for each video frame: each label corresponds to each candidate footage, i.e., the number of labels L is the same as the number of candidate footages. We formulate this as Markov Random Field (MRF) as follows:

$$\text{argmin}_i E = \sum_p V_p(l_i) + \lambda \sum_{p,q} W_{p,q}(l_i, l_j), \tag{2}$$

where labels of l_i and l_j are assigned to the neighboring frames of p and q. V_p is the data term, and $W_{p,q}$ is the smoothness term. $V_p(l_i)$ is defined by the Euclidean distance between the feature vector of the input music at p frame and the feature vector of the candidate footage l_i. $W_{p,q}(i,j)$ defines the cost of transition from footage l_i at p-th frame to footage l_j at q-th frame. We can solve the energy minimization using the graph-cut algorithm with α-β swap [13].

Fig. 11 shows an example of the energy minimization. Here are five candidate footages, A, B, C, D, and E. Here are nine frames. For the 1st and 2nd frames, candidate A is assigned, because the data term of A, 0.1 is smaller than the data term of B, 0.2. For the same reason, candidate E is assigned to the 8th and 9th frames. On the other hand, for the 3rd frame, candidate B who has smaller value, 0.2, is not assigned, but candidate C is assigned. This is because, if we assign B here, it produces a short footage whose duration is just one frame at the 3rd frame: the resulting video that has many such short footages looks annoying because of many scene transitions. We use the smoothness term to avoid this, i.e., $W_{p,q}$ is set to a large value for the transition that might cause too short frame. More concretely, $W_{2,3}(A,B)$ is set to a large value, here.

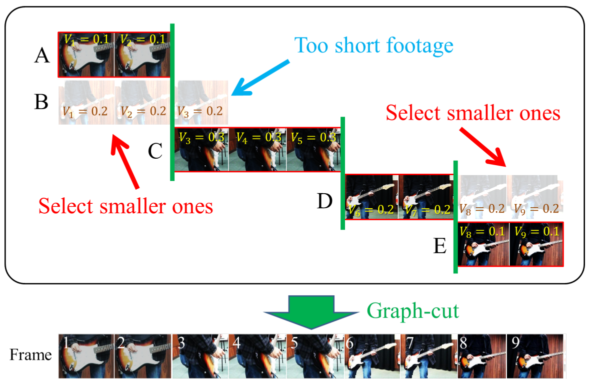

Figure 11: The energy minimization of MRF automatically selects the reasonable candidates. Here are five candidates, A, B, C, D, and E, and here are nine frames.

6 RESULTS

We present that our system can synthesize the video of performers appearing to play user-specified music. In the first experiment, we present that our system can synthesize the video of an arbitrary music using the video database of amateur performers. In the second experiment, we try to synthesize the music video using videos of performers who cannot play the instruments. Finally, we perform subjective evaluations to analyze the usability of our system and the quality of the synthesized videos.

6.1 Experiment 1

As the input music, we selected two music. One is “Let it be” of The Beatles, and the other is “Etupirka” of Taro Hakase. As for “Let it be” played with a violin, a bass, and drums, we could prepare the wave files for each instrument in advance. As for “Etupirka” played with a violin and drums, since we could prepare only the original wave file, we applied semi-automatic sound source separation technique to separate it into the violin part and the drums part. As for the video database, we asked the amateur performers of viola, bass, and drums to play each instrument. We did not specify what they should play but asked to play the instrument freely, i.e., any music randomly. We recorded their performance for one hour using four cameras at the same time. These cameras are set at the different viewpoints.

The supplementary video has the results. (Please use an appropriate video player to watch them: in our environment, Windows Media Player did not play our results with nice synchronization, i.e., there were significant time shift between the audio and the footages. QuickTime player was much better.) Note that each sound synchronizes with the movie, especially, drum sounds do.

The result of “Etupirka” demonstrates the limitation of our technique: we can find the scenes where the performer moves but there is no sound (or the performer does not move but there are sounds). This is caused by the failure of the sound source separation. Fig. 13 shows the feature vectors in “Let it be” and “Etupirka”. “Let it be” was actually good, but “Etupirka” is more noisy. Sound source separation is a difficult problem. Our manual selection of pLCA bases makes it possible to obtain relatively reasonable separation results, but still causes such noises. To synthesize a high quality music video, if we could prepare the audio track for each instrument independently, it would be the best.

Since these results are synthesized using graph-cut, we did not spend much time to synthesize the performance video: we took about 15 minutes to synthesize each of these videos.

6.2 Experiment 2

We prepared the input music “untitled” by an amateur composer for this experiment: this music is played with the guitar, the bass, the piano, and the drums. We asked four students from the computer science department to be the performers: these students have had little expериecne of the instruments. We allowed them to practice each instrument for one hour, and recorded their performance using two cameras: one camera was hand-held and moving, and the other was fixed.

The supplementary video has the result. While these performers were novices and just pretending to playing the instrument, our system can synthesize the performance video by retrieving footages and assigning them to the input music. Also, as described above, we synthesized the scenes of all the band members by mixing the solo performance videos recorded from the fixed camera.

The guitar and bass performers always strike the same string, and the fingers of the left hand do not move much. As the result, if the change of the pitch of input music is significant or the audience has knowledge about the instrument, the unnaturalness of the synthesized video would become noticable.

To synthesize this result, we did not use graph-cut to synthesize more high-quality video. We further spend time to edit camera motions digitally. We took 3 hours to synthesize this band performance video.

6.3 Subjective Evaluation

We performed a subjective evaluation of the quality of the synthesized video. We prepared six types of the videos:

v1: video of "Let it be" synthesized using graph-cut

v2: video of "Let it be" synthesized without graph-cut

v3: video of "Etupirka" synthesized using graph-cut

v4: video of "Etupirka" synthesized without graph-cut

v5: video of "untitled" synthesized without graph-cut

v6: video of "untitled" synthesized using After Effects

We asked the 15 students of the computer science department to evaluate “how unnatural each video looks”. Fig. 15 is the result of this user study, where the score 0 means there is no unnatural scene, and the score 6 means that the video is full of unnatural scenes.

v2 and v4 got smaller scores than v1 and v3. This shows that the quality of the video produced by automatic candidate selection tends to be lower. However, this result also demonstrates that the quality of the graph-cut is not bad. Especially, the score of v4 is comparable to that of

Figure 12: The frames from the “Let it be” and “Etupirka” videos.

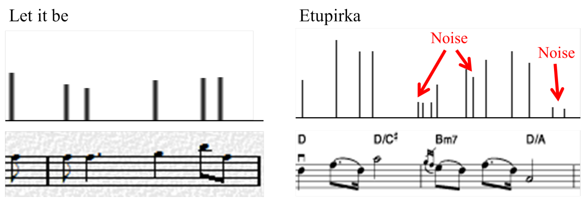

Figure 13: The example of the feature vectors.

v3. We believe the additional manual edit on the result produced by the graph-cut would improve the quality.

v1 got the worst score. The score of v2 is not good, either. The main reason was the drums: in the video of the drum player, he beats the tom. However, since the sound source separation between tom and snare drum was difficult, we treat these sounds as the same. As the result, the evaluators realized this fact, and they assigned the bad score to this video. In v2, since such unnatural scenes could be removed manually, the score is better.

The interesting fact was that the score is affected by the evaluator’s experience of each instrument a lot: several evaluators who had experiences to play the guitar or drums tend to realize the unnaturalness of these instruments in the video. On the other hand, no evaluator had experience to play the violin, and there was few comments about the unnaturalness of the violin. In v3 and v4, the violin is the main instrument, and the drums are simple, which we believe is the reason about why the score of them are better than v1 and v2.

v5 and v6 are for comparison between our system and standard video editing software. We chose Adobe After Effects as the video editing software. To make v5, we manually selected the candidate footages on Adobe After Effects. To make v6, we used only Adobe After Effects, which was completely manual work: we manually search for an adequate video from the database, and edit it. For both videos, we did our best to synthesize as the high quality video as possible. The difference of these scores is very small, which means the qualities of the resulting videos are almost the same. This fact proves that our system can synthesize the video whose quality is similar to the video synthesized completely manually. The performers in these videos had almost no knowledge about how to play each instrument. Most of the evaluators realized the unnaturalness of the performance, e.g., guitar and bass players’ wrong motions. It is difficult to synthesize the perfect music video of these band members: if we want to synthesize it, we must ask them to practice the instruments hard. This is the other limitation of our method.

Finally, we measured and compared the time required to synthesize a short, solo performance video by our system and After Effects. We asked three test users who are the students of computer science department to synthesize a short video. Since all of the test users were not familiar with video editing, we taught how to use our system and After Effects to each test user, and gave 10 minutes to practice it respectively. We then ask them to synthesize a short video whose duration is several seconds consisting of a single footage. Fig. 16 is the average time spent to finish the task: our system took 59$\pm$10 seconds, and After Effects took 578$\pm$157 seconds.

When using After Effects, the user had to check the long video, retrieve an adequate part of it, and then manually assigns it the part of the music: when the assignment process, the manual temporal stretching of the footage is also required. On the other hand, since our system shows the candidate footages automatically, the burden of the user is dramatically reduced: all the user had to do is to check the candidate footages and select one of them. Also, when using After Effects, the qualities of the resulting videos are different from subject to subject. Using our system, the qualities are almost the same.

Through these subjective evaluations and user studies, we have shown that our system can synthesize music videos whose qualities are comparable to the videos manually edited using the existing software. We also demonstrated that the user’s burden to create the similar quality video is much less than the existing software. Also, our system is simple and can be used even by a novice user, after 10 minutes tutorial.

7 CONCLUSION AND FUTURE WORK

In this paper, we proposed a method to synthesize the video of performers appearing to play user-specified music. Technically, our method consists of feature extraction, semi-automatic sound source separation, video retrieval based on local feature mathcing, and automatic candidate selection by graph-cut algorithm. Through the experiments, we have shown that our system can synthesize the music video with the comparable quality but less burden than the standard video editing software.

Figure 14: The frames from the "untitled" videos.

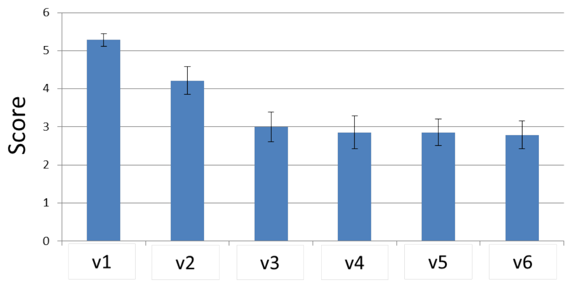

Figure 15: The evaluation of unnaturalness of the six synthesized videos.

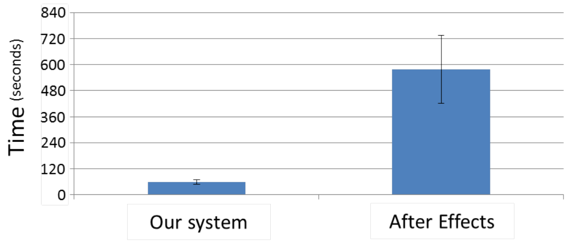

Figure 16: The time required to synthesize a short video.

In the future, we want to synthesize more high-quality video using performance videos of a professional band: currently, the band members are just amateurs or novices. We would like to try to synthesize a really professional music video and investigate how our system helps such a professional demand. The other future direction is to extend the method to take the visual information into account: currently, our synthesis method relies only on the audio information. However, by analyzing the body motion of the performers, we expect that we can exploit more semantic information for the music video synthesis.

8 ACKNOWLEDGEMENT

This work was supported by JSPS KAKENHI : Grant-in-Aid for Scientific Research (C) Grant Number 23500114. We also thank saku for providing the original music.

9 REFERENCES

[1] Bello, J., Daudet, L., Abdallah, S., Duxbury, C., Davies, M., Sandler, M.: A tutorial on onset detection in music signals. IEEE SAP pp. 1035–1047 (2005)

[2] Brants, T., Chen, F., Tsochantaridis, I.: Topic-based document segmentation with probabilistic latent semantic analysis. In: Proc. of CIKM, pp. 211–218 (2002)

[3] Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM pp. 381–395 (1981)

[4] Foote, J., Cooper, M., Girgensohn, A.: Creating music videos using automatic media analysis. In: Proc. of ACM Multimedia, pp. 553–560 (2002)

[5] Goto, M.: An audio-based real-time beat tracking system for music with or without drum-sounds

[6] Hua, X.S., Lu, L., Zhang, H.J.: Automatically converting photographic series into video. In: Proc. of ACM Multimedia, pp. 708–715 (2004)

[7] Kawahara, H., Morise, M., Takahashi, T., Nisimura, R., Irino, T., Banno, H.: Tandem-straight: A temporally stable power spectral representation for periodic signals and applications to interference-free spectrum, f0, and aperiodicity estimation. In: Proc. of ICASSP (2008)

[8] Nakano, T., Murofushi, S., Goto, M., Morishima, S.: Dancereproducer: An automatic mashup music video generation system by reusing dance video clips on the web. In: Proc. of SMC, pp. 183–189 (2011)

[9] Sakoe, H., Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition. In: Readings in speech recognition, pp. 159–165 (1990)

[10] Schutz., M., Manning., F.: Looking beyond the score: The musical role of percussionists' ancillary gestures. In: A journal of the Socirty for Music Theory (2012)

[11] Shiratori, T., Nakazawa, A., Ikeuchi, K.: Dancing-to-music character animation. Comput. Graph. Forum pp. 449–458 (2006)

[12] Smaragdis, P., Raj, B., Shashanka, M.: Supervised and semi-supervised separation of sounds from single-channel mixtures. In: Proc. of ICA, pp. 414–421 (2007)

[13] Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., Rother, C.: IEEE Trans. Pattern Anal. Mach. Intell. pp. 1068 –1080 (2008)

[14] Tomasi, C., Manduchi, R.: Bilateral filtering for gray and color images. In: Proc. of ICCV, pp. 839–846 (1998)