Vol.2010-CG-141 No.9
2010/11/9

Motion Synthesis for Synchronizing with Streaming Music

Jianfeng Xu, Koichi Takagi and Ryoichi Kawada[†1]

In this report, we present the first system to automatically synthesize human motion that is synchronized with streaming music using both rhythm and intensity features. In our system, a motion capture database is re-organized into a graph-based representation with meta-data (called *meta motion graphs*) beforehand, which is specially designed for the streaming application. When receiving an amount of music data as a segment, our system will search a best path for the segment on a meta motion graph. This approach can compose motions segment by segment, which (1) are synchronized with the music at a beat level, (2) are connected seamlessly with the previous segment, and (3) have the necessary synchronization capacity for the remaining music.

1. Introduction

To enrich the experience of music, automatic generation of a CG music visualizer is attracting ever greater attention. By means of the visualizer, 3D dancing animation is concatenated to synchronize with the music by reusing motion capture data[1)–3)]. However, there are few systems for streaming music, although streaming music such as Internet radio is becoming more popular as portable devices with Internet connectivity become more freely available to the public. The purpose of this paper is to automatically generate 3D dancing animation that is synchronized with streaming music, targeting at the entertainment of nonprofessionals. In such an application, the basic constraint is that only part of the music is available for our system at a time instant to synthesize a dancing animation on-the-fly, which causes the so called *horizon problem*[4)], and there is no chance to further modify any of the rendered animation.

Regarding people's dancing to music, one of the features common to both dance and music is the rhythmic structure, thus this feature is commonly used in music synchronization systems[1)–3)]. Obviously, rhythmic structure provides the basis for synchronization in the temporal domain. Moreover, the matching of intensity between motion and music is considered to be the next most important feature in Shiratori's system[2)], which is related to the spatial domain. As pointed in[2)], it is derived from the fact that people feel quiet and relaxed when listening to relaxing music such as a ballad, and they feel excited when listening to intense music such as hard rock music.

From the literature, the following are the definitions of the basic concepts used in the systems[1)–3)]. *Motion beats*, the concept of which is borrowed from the beat of music and thus reflects the rhythmic structure of dancing motions, are defined as the regular moments when the movement is changed significantly in direction or magnitude[1)]. *Motion intensity* expresses the excitement of motion[2)], which may be expressed as the kinetic energy[5)].

In this paper, we also employ the above rhythm and intensity features as a means to obtain music synchronization. Therefore, two straightforward requirements are the beat and intensity features in the generated motion should be synchronized with those in music respectively. Moreover, the motion should be synthesized in high quality. These three requirements are the main objective in our system. As one of the prevailing approaches to re-use of motion capture data, the concept of so-called *motion graphs*[6)] is extended to a graph-based representation with meta data, which is specially designed for the streaming application. Then, to synchronize motions with streaming music, a best path is searched segment by segment on our graphs. Naturally, the longer the segment is, the better the synchronization is. However, the length of a segment is restricted to the permitted delay, buffer size, and so on. Moreover, the total time of data processing should be no longer than the rendering time of the previous segment, which limits the scale of the motion capture database and the complexity of the synchronization algorithm. In the search stage, besides defining the important objective function for the above three requirements, it is essential to connect to the motions in the previous segment smoothly and guarantee the ability to synchronize with the unknown remaining segments in order to deal with the *horizon problem*[4)]. Our experiments in a user study demonstrate that the generated motion is plausible with a competitive quality to the conventional methods for non-streaming music.

†1 Media Solutions Laboratory, KDDI R&D Laboratories Inc.

2. Related Work and Issues to be Solved

In this report, we focus on the techniques based on reusing motion capture data[1)–3)]. To synchronize human motion with music, Kim et al.[1)] synthesize a new motion from a motion capture database according to the rhythmic pattern by traversing a movement transition graph to match the beat of the music. Due to absence of intensity constraints, it depends on external constraints to synthesize motions such as user's mouse movements as demonstrated in[1)]. Shiratori et al.[2)] employ both beat and intensity features in their system to improve the performance further, where the input music is divided into segments by the repeating patterns, and then candidate motion segments, the rhythm features of which are matched to those of each music segment, are searched for, and finally the motion is found whose intensity is similar to that of the music segments. Our previous work[3)] also uses both beat and intensity features to globally search a best path on a proposed weighted motion graph, which is much faster than Shiratori's method[2)] with comparable performance. From the viewpoint of features, Shiratori's system[2)] and our previous work[3)] are close to the system proposed in this paper. However, both systems are unsuitable for streaming applications, which is the target of this paper, as they use global searching.

3. Proposed System of Motion Synthesis for Streaming Music

3.1 System Overview

Our system uses a double-buffering scheme as shown in Fig. 1 (a), where the back buffer receives the streaming music and feeds an amount of music data (called *a segment*) to motion synthesis for music synchronization, and both the music and motion data are swapped to the front buffer for continuous rendering at the proper time. As shown in Fig. 1 (b), motion synthesis is performed for the current segment while the previous segment is being rendered, which requires that the processing time should be no longer than the rendering time of the previous segment. After an initial delay, our system can generate 3D dancing animation in real time. In the cases of spare time, a best-effort approach is certain to improve performance. However, in our implementation, a fixed-length segment is adopted for simplicity. More technically, our system is composed of two parts as shown in Fig. 2. In the first part, which is performed beforehand, we re-organize the motion capture database into meta motion graphs. In the second part, a human motion is concatenated on-the-fly from the meta motion graph to synchronize with the music segment by segment.

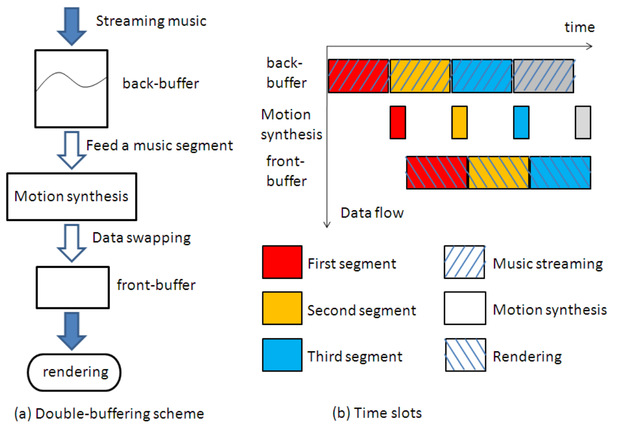

Fig. 1 Double-buffering scheme (a) and its time slots (b) for synchronizing human motion with streaming music.

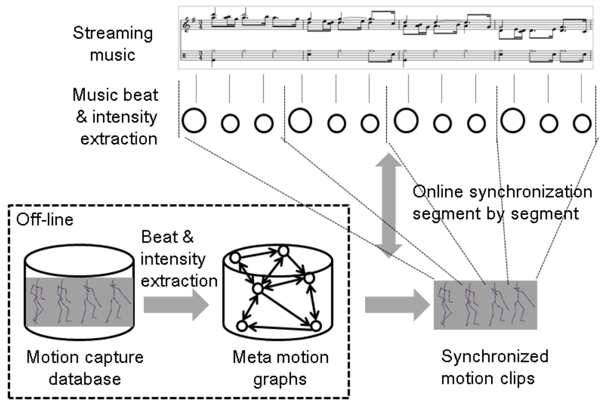
Streaming music
Music beat & intensity extraction
Online synchronization segment by segment
Off-line
Beat & intensity extraction
Motion capture database
Meta motion graphs
Synchronized motion clips

Fig. 2 System framework for generating human motion that is synchronized with streaming music. Meta motion graphs are constructed beforehand and a best path is searched in the graph to generate a synchronized motion segment by segment.

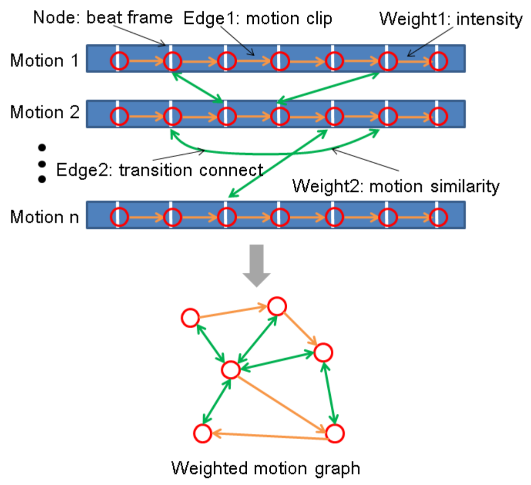

Fig. 3 Construction of a weighted motion graph[3], where only beat frames are checked for transition possibility and edge weights are assigned for music synchronization.

3.2 Novel Motion Graphs for Streaming Application

As described in Section 2, it is necessary to embed the features used in music synchronization and the so-called synchronization capacity for the nodes in the proposed meta motion graphs. First, we separate the entire database into sub-groups by the motion genres from the annotations in the database (recorded during the capturing procedure[7]), and motion tempos which are calculated from motion beat extraction (separating a genre equally with a fixed tempo range). Then, a meta motion graph is constructed in a sub-group. This technique ensures that the motions in a graph belong to the same motion genre and share a range of tempos while limiting the size of the motion graphs.

Motion graph construction: In our previous work[3], a weighted motion graph is proposed for music synchronization in a non-streaming mode. This allows the beat and intensity to be embedded in the graph. Here is a brief introduction. Before construction of our meta motion graphs, it is necessary to extract the features including motion beats and intensity from each motion in the database. A short-term principal component analysis (short-term PCA) method, proposed in our previous work[8], is adopted to detect the frames at beat instants (called *beat frames*) in each motion. The motion intensity can be calculated as the total kinetic energy between two neighboring beats to express the level of excitement of motions[5]. As shown in Fig. 3, a weighted motion graph WMG is constructed in a sub-group as $\{V, E, W(E)\}$ for the set of nodes, edges, and edge weights, respectively. The node set V includes all of the beat frames in the sub-group. The edge set E consists of two subsets including the sets of mono-directional edges E^1 and bi-directional edges E^2, and the edge weight set $W(E)$ has two corresponding subsets, i.e., $W^1(E)$ for E^1 and $W^2(E)$ for E^2.

Two successive beat frames F_B^i and F_B^{i+1} are connected by a mono-directional edge $e^1(F_B^i, F_B^{i+1})$ and the motion intensity between the beat frames is assigned as the edge weight $w^1(F_B^i, F_B^{i+1})$, where F_B^i denotes the i-th beat frame in a motion. For beat frames F_B^i and F_B^j with similar poses that satisfy (1), they are connected by a bi-directional edge $e^2(F_B^i, F_B^j)$ and the edge weight $w^2(F_B^i, F_B^j)$ is calculated as (2).

$$rd \equiv \frac{d(t_B^i, t_B^j)}{d(t_B^i - 1, t_B^i + 1) + d(t_B^j - 1, t_B^j + 1)} < THS \tag{1}$$

$$w^2(F_B^i, F_B^j) = \begin{cases} rd & \text{if } rd > THR \\ THR & \text{others} \end{cases} \tag{2}$$

$$d(t_B^i, t_B^j) = \sum_{m=1}^{M} w(m) \parallel \log(\mathbf{q}_{j,m}^{-1} \mathbf{q}_{i,m}) \parallel^2 \tag{3}$$

where t_B^i denotes the frame index of beat frame F_B^i, the frame distance $d(t_B^i, t_B^j)$ is calculated as the weighted difference of joint orientations[9], THS is a threshold that controls the number of bi-directional edges, THR is a threshold larger than 1, M denotes the number of joints in a frame, and $\mathbf{q}_{i,m}$ is the orientation of joint m in frame F_B^i. Here we compare the frame distance of two beat frames with those of their neighboring frames to decide if the beat frames are similar or not.

Dealing with the horizon problem: As pointed out by Forsyth et al.[4], the horizon problem is one of the main concerns in local search, which is a characteristic unique to streaming applications. A local search like "greedy algorithm" has the possibility of falling local minimum due to the invisibility of future music, which, in the worst case, leads to no path in the motion graph being available for the future music. Therefore, it is helpful to know how many successors the potentially selected node can be followed by. For example, the end node (from which no mono-directional or bi-directional edge exists) has no ability to connect any more. Fortunately, it is possible to calculate the synchronization ability for the node in the stage of motion graph construction, referred to as *synchronization capacity*. Thus, we can guarantee that a path exists for the next segment in the stage of motion synthesis by limiting the last node in the current segment in those nodes the synchronization capacities of which are larger than the threshold.

The calculation of synchronization capacity is based on the following facts.

(1) If a node u is an end node, its synchronization capacity $C(u)$ is zero, i.e., $C(u) = 0$.
(2) If a node v only has the successor u and $C(u) \neq \infty$, then $C(v) = C(u) + 1$.
(3) If a node u is in a cycle, its synchronization capacity $C(u)$ is infinite, i.e., $C(u) = \infty$.
(4) If a path exists from node u to v, then $C(v) \geq C(u)$.

For each node, we first decide if it is an end node or in a cycle (based on the depth-first search method). Then, we find those nodes that reach the above nodes and calculate the synchronization capacity by Property (2) or (4) iteratively.

Once the synchronization capacity of each node is calculated in the weighted motion graphs, our meta motion graphs are constructed by adding the synchronization capacity $C(u)$ as node weight $w(u)$, denoted as $MMG = \{V, E, W(V), W(E)\}$, where $W(V)$ is the set of node weights.

3.3 Motion Composition

Basically, we search a best path in a meta motion graph MMG segment by segment with the constraints from the previous and future segments. At the first segment, as initialization, our system will select a meta motion graph that is compatible with the music genre and covers the music tempo.

For any segment of music data, we first extract music beats using existing techniques such as[10] and the music intensity, which is calculated as the sound pressure level between two beat instants. Then, a best path for the current segment is searched in the selected graph beat by beat. To seamlessly connect with the previous segment, the last node in the previous segment is selected as the first node in the current segment. Moreover, the last node in the current segment is limited to those nodes with enough synchronization capacity to avoid the case where there is no path to synchronize with the remaining music. Furthermore, it is important to properly define an objective function for matching the motion with music.

Basically, three requirements should be met in the generated motion. First, in the temporal domain, beat instants in the generated motion should be synchronized with those in the music. Because the beat frames are and only are in the node set V, we can exactly synchronize the beat instants in any path with those in the music by modifying the frame rate between two nodes in the path. In other words, there is no time difference between the corresponding beat instants in the music and the motion. Therefore, the cost of beat synchronization is controlled to zero, i.e., the granularity of synchronization is at the beat level. Second, in the spatial domain, the motion intensity should be matched to that in the music. Their error is expressed by the absolute difference between the normalized values of motion intensity in the music and the motion. Third, the generated motion should be as smooth as possible, where the source of motion artifacts is the motion inconsistence at bi-directional edges and the temporal modification.

As a result, the cost function of an edge $e(F_B^i, F_B^j)$, which is a mono-directional edge $e^1(F_B^i, F_B^j)$ or a bi-directional edge $e^2(F_B^i, F_B^j)$ with its following mono-directional edge $e^1(F_B^j, F_B^{j+1})$, is defined as:

$$edgeCost(e(F_B^i, F_B^j), k) = \begin{cases} intCost & \text{if } e(F_B^i, F_B^j) \in E^1 \\ s \cdot w^2(F_B^i, F_B^j) \cdot intCost & \text{if } e(F_B^i, F_B^j) \in E^2 \\ \infty & \text{others} \end{cases} \tag{4}$$

$$intCost = \| \overline{w^1}(F_B^i, F_B^j) - \overline{I}(k) \| \tag{5}$$

$$s = \max\left(\frac{rate(F_B^{i-1}, F_B^i)}{rate(F_B^j, F_B^{j+1})}, \frac{rate(F_B^j, F_B^{j+1})}{rate(F_B^{i-1}, F_B^i)}\right) \tag{6}$$

where $intCost$ denotes the cost from intensity matching, s denotes the cost from the change of modification strength, w^2 denotes the edge weight in the set of W^2, which shows the cost from the motion inconsistence at bi-directional edges (see (2)), $\overline{I}(k)$ denotes the normalized music intensity from the k-th beat to $(k+1)$-th beat or the k-th beat interval, $\overline{w^1}$ denotes the normalized edge weight in the set of W^1, and $rate(m,n)$ denotes the modified frame rate between beat frame m and n. Note that standard scores are calculated for the normalization and $s \cdot w^2$ is a penalty for selecting a bi-directional edge. In addition, since the bi-directional edge takes no time, it is only used with the following mono-directional edge $e^1(F_B^j, F_B^{j+1})$ (i.e., no cost definition for just a bi-directional edge).

Then, assuming we have K beat intervals in the music segment, our objective function is $\min \sum_{k=1}^{K} edgeCost(e(F_B^i, F_B^j), k)$, which can be optimized by a method similar to dynamic programming algorithm as shown in (7)-(9). As described previously, the initial node is the last node in the previous segment. For the first segment, we use the first beat frames of all the motions in the graph as initial nodes $InitS$.

$$P(F_B^v, 0) = \begin{cases} 0 & if \quad F_B^v \in InitS \\ \infty & others \end{cases} \tag{7}$$

$$P(F_B^v, k) = \min_{F_B^i \in V} \{P(F_B^i, k-1) + edgeCost(e(F_B^i, F_B^v), k)\} \tag{8}$$

$$P(K) = \min_{F_B^v \in V \,\&\, C(F_B^v) \geq THC} \{P(F_B^v, K)\} \tag{9}$$

where $P(F_B^v, k)$ denotes the cost of a best path for the first k beat intervals with the last node of F_B^v, $P(K)$ is the cost of a best path for the entire segment of music, $C(F_B^v)$ denotes the synchronization capacity of the node of F_B^v, and THC is the beat number of the remaining music if known or ∞ otherwise.

Equation (8) shows that the current best path with the last node of F_B^v is searched based on the previous best path the last node of which is a node F_B^i. In standard dynamic programming, the cost function is the edge weight that remains constant. In our definition, the cost function is dynamic, which is updated with the music intensity for the k-th beat interval as (4). Note that our method is a local searching method from the viewpoint of the entire piece of music. If the average indegree of the nodes is D and the node number is N in the graph, we only need to compare DNK times to search the best path, which is linear to the number of beat intervals K in the music segment.

4. Experimental Results and Discussions

Experimental Conditions: Two pieces of music are tested from the RWC music database[11] including one with a slow tempo (Song004) and one with a fast tempo (Song091). The dancing motions are from the CMU motion capture database[7] including break and Indian dances, resulting in four different kinds of content in our experiments. In order to reduce the evaluation time, only a part (30 seconds) of a song is extracted for the user study with fade in and fade out operations at cut points, which makes the shortened songs more natural and the music intensity unevenly distributed. The global dynamic programming method (baseline)[3] and Shiratori's method[2] are compared with the proposed method, where the beat and intensity features are common to all the methods. The threshold THS in (1) is set as 3.0. The threshold THC in (9) is set as ∞.

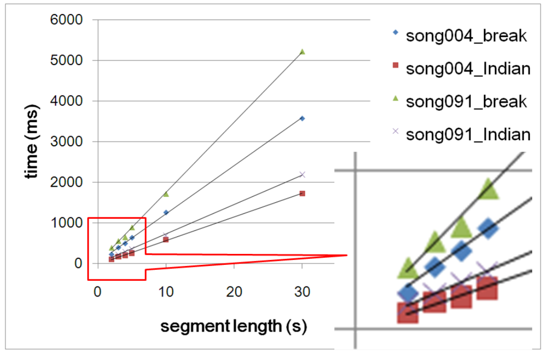

Fig. 4 Comparison of average computational cost (milliseconds) for a segment. Segment length includes 2s, 3s, 4s, 5s, 10s, and 30s (entire length).

Computational Cost: All the experiments are performed on the same PC with an Intel(R) Core(TM)2 Quad CPU (2.83 GHz) and a 3.25 GB RAM memory. As shown in Fig. 4, it takes less than 0.5 seconds on average to synthesize a 5s-segment motion in our method for the Indian dance, increasing the initial delay very little. At the same time, it is much shorter than the rendering time, which is one of the constraints for streaming applications. In addition, it is observed that the computational cost is almost linear to the segment length as discussed in Section 3.3. Moreover, it demonstrates reasonable accuracy in terms of its capacity to predict the computational time t for any segment length L given the average indegree D, node number N, and the music tempo P in a segment. In

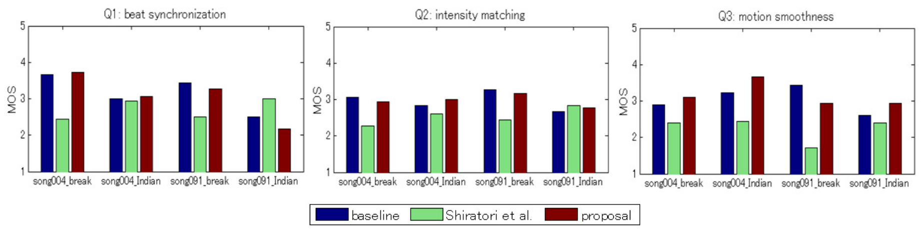

Fig. 5 Mean opinion scores (MOSs) of three different methods including the baseline method (global dynamic programming[3]), Shiratori's method[2], and the proposed method.

the above PC, we get $t = DNP(0.0030L - 0.0003)$ by the linear regression.

User Study: As the target of our application is general consumers, a total of 18 non-expert observers (ages in their 20s and 30s) are selected to evaluate the results in our user study. The participants have evaluated the above 4 samples by assigning a score for 1 to 5 for the 3 questions about beat synchronization, intensity matching, and motion smoothness, respectively, where "strongly disagree", "somewhat disagree", "neutral", "somewhat agree", and "strongly agree", are denoted by the numbers 1 to 5 in that order. By randomly displaying the results, participants do not know which sample is generated by which method.

Fig. 5 shows the mean opinion scores (MOSs) from all the participants for all tests, where the length of the music segment is set as 5 seconds by preliminary experiments. The MOSs in both the baseline method and the proposed method are generally not worse than those in Shiratori's method[2]. At the same time, a t-test is performed between the proposed method and the baseline, which shows that there is no significant difference between the proposed method and the baseline method in all the four contents for all the three questions.

5. Conclusions

This paper has proposed a novel scheme for motion synchronization with streaming music using a motion capture database. As far as we know, this is the first such system for streaming music using both beat and intensity features. Specially designed for this purpose, a graph-based representation called *meta motion graph* is constructed on the database beforehand, where not only are the necessary features embedded but also the synchronization capacity is calculated to express the ability to synchronize with music. Therefore, our system can search a best path for each segment, considering not only the synchronization of the current segment but also the constraints of the previous and future segments.

6. Acknowledgment

The MoCap data used in this project were obtained from mocap.cs.cmu.edu. The database was created with funding from NSF EIA–0196217. The music data were obtained from staff.aist.go.jp/m.goto/RWC-MDB/[11]. The CG model was provided by (c) ISAO witchcraft. All the participants in our user studies are greatly appreciated.

References

1) T.H. Kim, S.I. Park, and S.Y. Shin, "Rhythmic-motion synthesis based on motion-beat analysis," *ACM Trans. on Graphics*, vol. 22, no. 3, pp. 392–401, Jul 2003.
2) T.Shiratori, A.Nakazawa, and K.Ikeuchi, "Dancing-to-music character animation," *Computer Graphics Forum*, vol. 25, no. 3, pp. 449–458, Jul 2006.
3) J.Xu, K.Takagi, and R.Kawada, "Automatic composition of motion capture animation for music synchronization," in *Proc. of EUROGRAPHICS short paper*, 2010, pp. 85–88.
4) D.A. Forsyth, O.Arikan, L.Ikemoto, and J.F. O'Brien, "Computational studies of human motion: Part 1, tracking and motion synthesis," *Foundation and Trends in Computer Graphics and Vision*, vol. 1, no. 2, pp. 77–254, 2006.
5) K. Onuma, C. Faloutsos, and J.K. Hodgins, "FMDistance: A fast and effective distance function for motion capture data," in *Proc. of EUROGRAPHICS 2008 Short Papers*, 2008.
6) L.Kovar, M.Gleicherl, and F.Pighin, "Motion graphs," *ACM Trans. on Graphics*, vol. 21, no. 3, pp. 473–482, Jul 2002.
7) CMU Motion Capture Database, *http://mocap.cs.cmu.edu/*, 2003.
8) J.Xu, K.Takagi, and A.Yoneyama, "Beat induction from motion capture data using short-term principal component analysis," *The Journal of The Institute of Image Information and Television Engineers*, vol. 64, no. 4, pp. 577–583, April 2010.
9) J.Wang and B.Bodenheimer, "An evaluation of a cost metric for selecting transitions between motion segments," in *SCA '03*, 2003, pp. 232–238.
10) S.Dixon, "Automatic extraction of tempo and beat from expressive performances," *Journal of New Music Research*, vol. 30, no. 1, pp. 39–58, Mar 2001.
11) M.Goto, H.Hashiguchi, T.Nishimura, and R.Oka, "Rwc music database: Popular, classical, and jazz music databases," in *ISMIR '02*, 2002, pp. 287–288.