IEICE TRANS. INF. & SYST., VOL.E95–D, NO.6 JUNE 2012

PAPER

Efficient Generation of Dancing Animation Synchronizing with Music Based on Meta Motion Graphs

Jianfeng XU[†a)], ***Nonmember***, **Koichi TAKAGI**[††b)], ***Member***, ***and*** **Shigeyuki SAKAZAWA**[†c)], ***Senior Member***

SUMMARY This paper presents a system for automatic generation of dancing animation that is synchronized with a piece of music by re-using motion capture data. Basically, the dancing motion is synthesized according to the rhythm and intensity features of music. For this purpose, we propose a novel meta motion graph structure to embed the necessary features including both rhythm and intensity, which is constructed on the motion capture database beforehand. In this paper, we consider two scenarios for non-streaming music and streaming music, where global search and local search are required respectively. In the case of the former, once a piece of music is input, the efficient dynamic programming algorithm can be employed to globally search a best path in the meta motion graph, where an objective function is properly designed by measuring the quality of beat synchronization, intensity matching, and motion smoothness. In the case of the latter, the input music is stored in a buffer in a streaming mode, then an efficient search method is presented for a certain amount of music data (called a segment) in the buffer with the same objective function, resulting in a segment-based search approach. For streaming applications, we define an additional property in the above meta motion graph to deal with the unpredictable future music, which guarantees that there is some motion to match the unknown remaining music. A user study with totally 60 subjects demonstrates that our system outperforms the stat-of-the-art techniques in both scenarios. Furthermore, our system improves the synthesis speed greatly (maximal speedup is more than 500 times), which is essential for mobile applications. We have implemented our system on commercially available smart phones and confirmed that it works well on these mobile phones.

key words: *motion graph, music synchronization, motion synthesis, optimization, mobile applications*

1. Introduction

To enrich the experience of music, automatic generation of dancing animation is attracting ever greater attention, where the dancing motion is concatenated to synchronize with the music by re-using motion capture data [1]–[3]. Basically, these techniques are inspired by the phenomenon of people wanting to dance to music. Regarding people's dancing to music, one of the features common to both dance and music is the rhythmic structure [4], [5], 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 [3] by Laban theory [6], which is related to the spatial domain. As pointed out by Shiratori et al. [3], 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, we restate the following 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 [3], which may be expressed as the kinetic energy [7].

In this paper, we also employ the above rhythm and intensity features as a means to obtain music synchronization. Comparing to the conventional methods, we improve both the quality of generated animation and the speed of motion synthesis with the goal of mobile applications. The basic idea is to prepare a special data structure beforehand, considering the potential of improving the quality of dancing animation and reducing the computation of motion synthesis. For this purpose, we propose a novel meta motion graph structure to embed the necessary features including both beat and intensity, which is extended from the concept of so-called *motion graphs* [8]–[10]. Therefore, motion synthesis is cast as a searching problem for a path on the graph that minimizes an objective function, where dynamic programming is effective to greatly reduce the computational cost. At the same time, a proper objective function is important in our application. Two straightforward requirements are that (1) the beat and (2) intensity features in the generated motion should be synchronized with their corresponding features in music. Moreover, as the third requirement, (3) the motion should be synthesized in high quality, where we focus on the motion smoothness in this paper. The above three requirements are the main objective in our system, which will be described in Sect. 3.2.

In this paper, we consider two scenarios for streaming music and non-streaming music, where local search and global search are required respectively. In the case of non-streaming music, the global optimization is achieved by the above dynamic programming algorithm. In the case of streaming music, the local search is mandatory. Basically, our system synchronizes the motions with music segment by segment, matching both the beat and intensity features.

Manuscript received July 19, 2011.
Manuscript revised January 19, 2012.

[†]The authors are with KDDI R&D Laboratories, Inc., Fujimino-shi, 356–8502 Japan.
[††]The author is with KDDI Corporation, Tokyo, 102–8460 Japan.
a) E-mail: ji-xu@kddilabs.jp
b) E-mail: ko-takagi@kddi.com
c) E-mail: sakazawa@kddilabs.jp

DOI: 10.1587/transinf.E95.D.1646

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. In such a segment-based searching approach, it is essential to connect to the motions in the previous segment smoothly and guarantee the ability to synchronize with the unknown remaining segments. Our system is carefully designed to deal with these constraints as shown in Sect. 4.2.

The main contributions of this paper are as follows:

- An efficient system is developed for both non-streaming applications and streaming applications that is implemented on mobile phones. A user study with 60 subjects demonstrates that our system outperforms the state-of-the-art techniques.
- A novel graph-based representation is proposed specially for music synchronization. Conventional motion graphs are unsuitable for our task because they only focus on the kinematics of motion without the necessary features for music synchronization. In this paper, a graph-based representation with meta data for the above purpose is presented.
- A proper objective function is designed for the three requirements of beat synchronization, intensity synchronization, and motion quality, which is inspired by psychologists [11], [12] and professional dancers. With the objective function, global searching and segment-based searching approaches are employed for non-streaming applications and streaming applications respectively.

The remainder of this paper is organized as follows. In Sect. 2, we briefly survey the related techniques on music synchronization. In Sect. 3, we describe the proposed system of motion synthesis for non-streaming music in detail, which serves as a basis of the next section. In Sect. 4, we describe the extension for streaming music. In Sect. 5, we report and discuss our experimental results. Finally, in Sect. 6, we present the conclusions of this paper.

2. Related Work and Issues to be Solved

While a vast literature is available on the subject of synchronizing video with music [13], [14], we focus on the techniques based on reusing motion capture data [1]–[3] in this section. In addition, there are several pioneer researches from audio/music field and robot field. A famous example is Cindy [15], developed by Goto et al. Basically, Cindy is a CG agent that dances to music, where six mappings between specific motions and specific drum-sounds are defined in advance. Autonomous dancing robots have been developed recently [16], [17], where those techniques focus on the audio processing, especially the beat induction in real time. For motions, their papers have not synthesized the motions for the music. Basically, a sequence of motions is arranged for the entire music beforehand. It works on relatively simple motions such as "simple raises, simple steps, arm swings, and arm and foot taps" in [16] and step motions in [17]. In our paper, we focus on motion synthesis while directly using the existing method to detect music beats. Our technique can automatically obtain the "best" combination of motions according to the synchronization of both beat and intensity feature and the motion's smoothness even on a much more complex motion database such as break dance. In this section, we discuss related work from the viewpoint of music synchronization systems, motion graphs techniques, and searching strategies in motion graphs.

2.1 Music Synchronization Systems

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[†]. Similarly, Alankus et al. [2] also use beat features to search a best path on their transition graph for music synchronization. Shiratori et al. [3] 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. From the viewpoint of features, Shiratori's system is close to the system proposed in this paper. Aiming at mobile applications, our system improves both the quality of generated animation and the speed of motion synthesis.

2.2 Motion Graphs

The basic idea for the above music synchronization systems [1]–[3] is re-use of motion capture data, where graph-based representations are demonstrated to be a very powerful tool. Many related studies have been reported, as briefly summarized in [18]. In the game industry, these graphs, called *move trees*, are originally created manually [19]. Several automatic methods are independently proposed to construct motion graphs in SIGGRAPH 2002 [8]–[10]. Basically, a motion graph is a directed graph structure of possible connections between motions, turning the set of initial clips from a list of sequences into a connected graph of movements.

Since then, many variations of motion graphs have been reported. Gleicher et al. [20] present *snap-together graphs* where common poses are used as hubs in the graph. Safonova and Hodgins [21] create an *interpolated motion graph* by combining the standard motion graph and interpolation techniques. Zhao et al. [22] further use interpolation of motion segments of the same contact to add additional data to the database in their *well-connected motion graph*. Besides interpolation, continuous constrained optimization is introduced in the *optimization-based graph* by

[†]Due to absence of intensity constraints, it depends on external constraints to synthesize motions such as user's mouse movements as demonstrated in [1].

Ren et al. [23]. Beaudoin et al. [24] cluster similar motions to get an understandable graph structure referred to as a *motion-motif graph*. In summary, these techniques focus on the kinematics of motion to obtain higher quality graphs.

However, kinematics on its own cannot realize our task of music synchronization due to the absence of such necessary features as motion beat and intensity. Moreover, in conventional motion graphs, there may not be any control over the structure of the graph, leading to the rhythmic structure of motion being destroyed [1] and the issue of difficult searching [20]. In this paper, a *meta motion graph* is constructed to obtain a structured graph-based representation that keeps the rhythmic structure of motion and embeds both beat and intensity features. Furthermore, we derive a necessary property (called *synchronization capacity*) of the nodes for the streaming application, which expresses the ability to synchronize with music.

2.3 Searching Strategies in Motion Graphs

Through motion graphs, motion synthesis is cast as a searching problem for a path that minimizes an objective function [25]. The searching strategy highly depends on the application purpose. Given user constraints, Kovar et al. [8] improve the depth-first search to obtain graph walks using a branch-and-bound strategy and incremental search. Lee et al. [10] use the greedy best-first search approach and traverse only a fixed number of frames to maintain a constant rate of motion. Arikan et al. [9] develop a hierarchical, randomized search strategy. They present another method [26] that uses a dynamic programming approach and coarse-to-fine refinement to search for the motion sequence. Dynamic programming, the complexity of which is linear to the path length, is also adopted by Lee and Lee [27].

However, the original dynamic programming algorithm is unsuitable for streaming applications, where local search is a must [25]. Local search, as used in [8], evaluates the properties of only a certain number of nodes ahead when choosing what node to transit to. This might lead to the *horizon problem* [25] due to the unpredictable future data. In this paper, we calculate the synchronization capacity in our meta motion graphs to alleviate this problem.

3. Proposed System of Motion Synthesis for Non-streaming Music

In this section, we describe our system for synchronizing dancing motion with non-streaming music by global searching approach, where the entire piece of music is available for motion synthesis. Note that the techniques in this section also serves as a basis of streaming music, where local search is mandatory.

As shown in Fig. 1, our system is composed of two parts. In the first part, which is performed beforehand, we re-organize the motion capture database into a graph structure by the motion beat and intensity. In the second part, a human motion is generated from the graph to synchronize

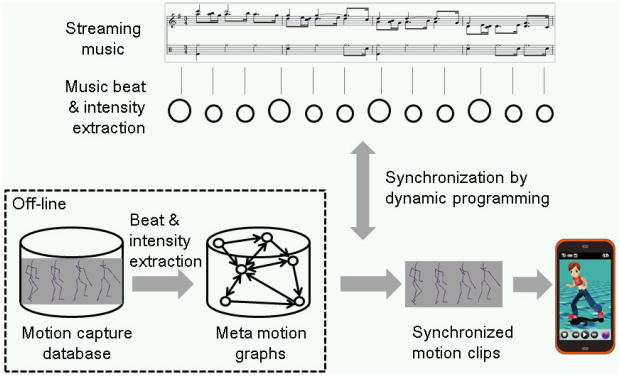

Fig. 1 System framework of generating a motion that is synchronized with the input music. Meta motion graphs are constructed beforehand and a best path is globally searched in the graph to generate a synchronized motion.

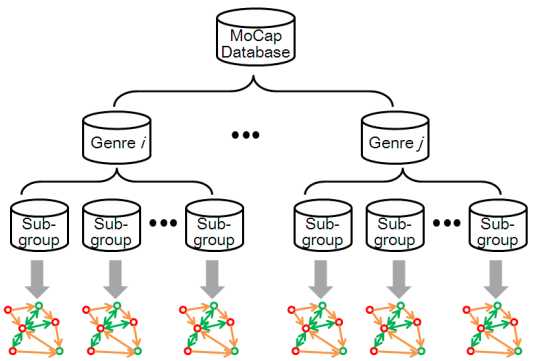

Fig. 2 Construction of meta motion graphs in sub-groups according to motion genres and motion tempos.

with the input music and then a CG character is rigged for rendering.

3.1 Construction of Meta Motion Graphs

As described in Sect. 2.2, it is necessary to embed the features used in music synchronization in the proposed meta motion graphs. As shown in Fig. 2, first, we separate the entire database into sub-groups by the motion genres from the annotations in the database (see [28]), and motion tempos which are calculated from motion beat extraction. 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.

Extraction of motion beat & intensity: 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, first proposed in our previous work [29], is adopted to detect the frames at beat instants (called *beat frames*) in each motion. We observe that major movement variance clearly reveals motion beats, which may be extracted by PCA due to the spatial correlation in motion. However, conventional global PCA does not work for complex motion because PCA is a linear

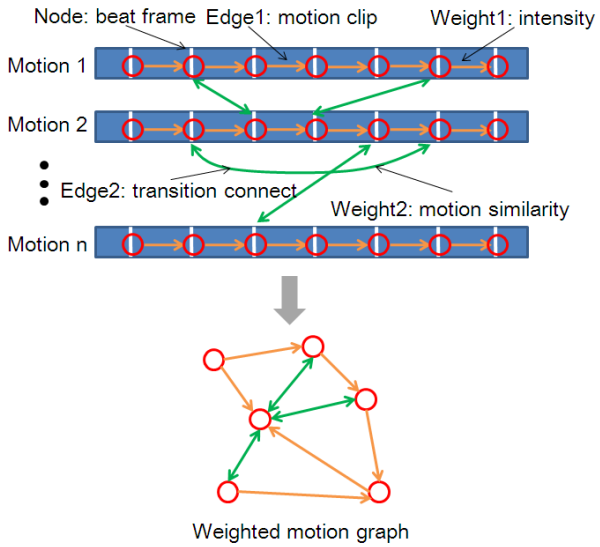

Fig. 3 Construction of a meta motion graph in a sub-group, where only beat frames are checked for transition possibility and edge weights are assigned for music synchronization.

model while a complex motion is highly non-linear. The basic idea in short-term PCA is similar to piece-wise linear approximation to a non-linear problem based on the fact that motion data are almost linear in the short term due to the strong temporal coherence. Short-term PCA performs PCA in a sliding window in joint position space and regards the peaks and nadirs of the coordinates in the first principal component as candidates for motion beats, which are further refined by frequency analysis as [1]. For the detailed procedure, please refer to [29]. After beat extraction, the motion tempo is regarded as the number of beats in a minute. The motion intensity can be calculated as the total kinetic energy between two neighboring beats to express the level of excitement of motions [7].

Motion graph construction: As shown in Fig. 3, a meta motion graph MMG 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., a measure of motion intensity $W^1(E)$ for E^1 and a measure of pose similarity $W^2(E)$ 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 Eq. (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 Eq. (2). Note that we let the bi-directional edge take no time to jump from one frame to another frame in order to preserve the rhythm structure when transiting by bi-directional edges among motions.

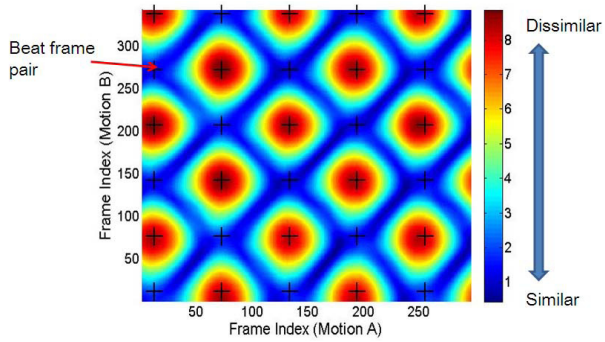

Fig. 4 Distances between frame pairs in two walking motions. Crosses denote the beat frame pairs, which locate the peaks and nadirs regularly, inferring that the corresponding poses in different cycles are properly found.

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

$$w^2(F_B^i, F_B^j) = \max(rd, THR) \quad (2)$$

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

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 [30]†, THS is a threshold that controls the number of bi-directional edges, THR is a threshold for penalty of selecting a bi-directional edge (set as 2.0 in our implementation), M is the joint number, $w(m)$ denotes the joint weight, 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. By the meta motion graphs, we obtain a structured graph-based representation instead of an unstructured one in the standard motion graphs [29], where the rhythmic structure is maintained.

Although only the beat frames are compared, most of the meaningful connections are preserved. For example, in two walking motions as shown in Fig. 4, the frame distances express well the periodicity of motions. Even if considering only the beat frame pairs (see crosses in Fig. 4), the corresponding poses (e.g. hitting ground) in different cycles are connected as desired. Moreover, the rhythmic structure is maintained in all the connections. Namely, any path from the connections will keep the original periodicity. Therefore, the greatest benefit of meta motion graphs is the ability to obtain a structured graph-based representation instead of an unstructured one in the standard motion graphs. This allows the rhythmic structure to be maintained and reduces the computational cost in both the construction and time-limited searching stages. This characteristic makes music synchronization possible.

†Although only pose information is used in Eq. (3), velocity information is essentially included in Eq. (1). Note that Eq. (3) is just an example to calculate the the motion/pose similarity, which works rather well in our system. More advanced definitions and their comparison are available in [31].

3.2 Motion Synthesis by Global Searching

Given a piece of music, we firstly extract music beats using existing techniques such as [32] and music intensity which may be the sound pressure level between two beat instants [33]. Then, we select a meta motion graph according to the music genre and music tempo. Lastly, a best path is searched in the graph by dynamic programming beat by beat.

Objective function design: Now we need 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 (The ratio of new frame rate and old frame rate is defined as *modification strength ms*. It is known that a reasonable tolerance *ms* (about 15%) gives little influence to motion quality [3]). 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 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 for beat synchronization. Note that although temporal modification of frame rate itself is regarded as no additional cost, the change of modification strength *ms* may cause motion artifacts. Obviously, these changes only happen at bi-directional edges, where the motion inconsistence exists simultaneously. Therefore, a penalty should be paid for selecting a bi-directional edge to avoid any deterioration of the motion quality, see the term $s \cdot w^2(F_B^i, F_B^j)$ in Eq. (4). Meanwhile, if a bi-directional edge is selected, its neighboring 24 frames are blended by SLERP function.

As a result, the objective 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:

$$eCost(e(F_B^i, F_B^j), k) = \begin{cases} \| \overline{w^1}(F_B^i, F_B^j) - \overline{I}(k) \| & \text{if } e(F_B^i, F_B^j) \in E^1 \\ s w^2(F_B^i, F_B^j) \| \overline{w^1}(F_B^j, F_B^{j+1}) - \overline{I}(k) \| & \text{if } e(F_B^i, F_B^j) \in E^2 \\ \infty & \text{others} \end{cases} \quad (4)$$

$$s = \max(ms, 1/ms) \quad (5)$$

where $\overline{w^1}$ denotes the normalized edge weight in the set of W^1, $\overline{I}(k)$ denotes the normalized music intensity from the k-th beat to $(k+1)$-th beat or the k-th beat interval, 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 Eq. (2)), and *ms* denotes the ratio of new frame rate and old frame rate. Note that standard scores are calculated for the normalization. In addition, since the bi-directional edge itself takes no time, it is only used with the following mono-directional edge to match music beat by beat.

Global searching with dynamic programming: Then, assuming we have K beat intervals in the music, our objective function is $\min \sum_{k=1}^{K} eCost(e(F_B^i, F_B^j), k)$, which can be optimized by dynamic programming as shown in Eqs. (6)–(8). At least one initial node is required in dynamic programming. We use the first beat frames of all the motions in the graph as multiple initial nodes $InitS$.

$$P(F_B^v, 0) = \begin{cases} 0 & if \quad F_B^v \in InitS \\ \infty & others \end{cases} \quad (6)$$

$$P(F_B^v, k) = \min_{F_B^i \in V} \{P(F_B^i, k-1) + eCost(e(F_B^i, F_B^v), k)\} \quad (7)$$

$$TP(K) = \min_{F_B^v \in V} \{P(F_B^v, K)\} \quad (8)$$

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, and $TP(K)$ is the total cost of a best path for the entire music.

Equation (7) shows that the current best path with the last node of F_B^v is searched based on the previous best path whose last node is any node F_B^i. In the standard dynamic programming, the objective function is the edge weight that is constant. In our task, the cost function is dynamic, which is updated with the music intensity for the k-th beat interval as Eq. (4). Suppose the average indegree 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. In the worst case of full graph, the complexity is $O(N^2K)$. However, the constructed graphs are rather sparse because only beat frames are permitted to connect together.

4. Proposed System of Motion Synthesis for Streaming Music

In this section, we describe our system for synchronizing human motion with streaming music by local searching approach. In such an application, the basic constraint is that only part of the music data are available for our system at a time instant to synthesize a dancing animation on-the-fly, which causes the so called *horizon problem* [25], and there is no chance to further modify any of the animation that has been rendered.

Our system uses a double-buffering scheme as shown in Fig. 5 (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

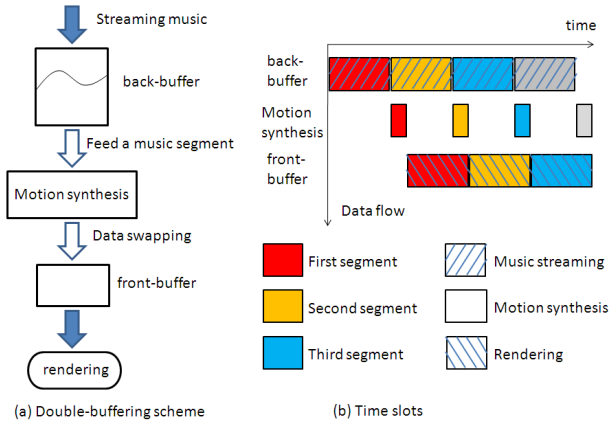

Fig. 5 Double-buffering scheme (a) and its time slots (b) of our prototype system for synchronizing human motion with streaming music.

Fig. 5 (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 dancing animation in real time (In this paper, the term of "real time" means that once the music gets started, the music and its affiliated graphics will be continuously played without any interruption.). 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.

4.1 Extended Meta Motion Graphs for Streaming Application

As pointed out by Forsyth et al. [25], 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 procedure of calculating synchronization capacity is based on the following four 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 (see Algorithm 2), 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 Fact (2) or (4) iteratively.

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

Although the calculation of synchronization capacity seems a kind of generalization of so called "pruning the dead ends and sinks", it is not enough to just prune the dead ends and sinks as Kovar et al. did [8] in our application. In many dancing genres, the entire motion is composed of several kinds of basic movements. Basically, the connections among different kinds of basic movements are rather weak, e.g. only mono-directional edges. By pruning the dead ends and sinks, those weak connections are removed, isolating the basic movements. In our extended meta motion graph, we preserve all the nodes to avoid the isolation and removal of those basic movements while knowing their synchronization capacity. Moreover, by keeping all the nodes, it benefits the first and last segments synthesizing better motions, where music intensity in the intro and ending sections is observed to be usually lower than others.

4.2 Motion Synthesis by Segment-Based Searching

As described before, we search a best path in the extended meta motion graph $EMMG$ 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 [32] and the music intensity, which is similar to non-streaming music. Then, a best path for the current segment is searched in the selected graph beat by beat using a method similar to non-streaming music in Sect. 3.2. However, 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. Therefore, Eq. (8) is modified as Eq. (9).

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

where $P(F_B^v, k)$ denotes the cost of a best path for the first k beat intervals in the segment with the last node of F_B^v, $TP(K)$ denotes 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. Note that the unit of optimization

is a segment instead of the entire piece of music as Sect. 3.2. Therefore, our method is a local searching method from the viewpoint of the entire piece of music. However, the computational complexity is the same as that in non-streaming music.

5. Experimental Results and Discussions

In this section, we analyze the performance of our system for both non-streaming music and streaming music with the comparison of Shiratori's system, which uses the same features as ours and thus is regarded as the comparison target. For implementation details of Shiratori's system [3], the length of music segments is set as 10 s and the threshold for candidate motion segments is set as 1.0 according to our preliminary investigation. In Sect. 5.1, the computational cost is measured on a common PC. In Sect. 5.2, the motion quality is evaluated and discussed including the beat synchronization, intensity matching, and motion smoothness.

5.1 Computational Cost

As described in Sect. 3.2, the complexity is linear to the length of music piece in our method while it is exponential in Shiratori's method. In this section, we analyze the computational cost on a PC with a Core2® Duo 2.2 GHz CPU and a 4 GB RAM memory. Two pieces of music are tested from RWC music database [34] including a slow-tempo song (Song004) and a fast-tempo song (Song091). The motions are obtained from CMU MoCap database [28] including the Break and Indian dances. Table 1 shows the computing time of motion synthesis by the global search and Shiratori's method respectively. It takes less than 10 seconds to generate a 60 s animation in our method for the Indian dance. In the worst case, it takes 80 seconds to generate a 60 s animation for Song091 and break dance. The average speedup is 169.5 times for 30 s contents and 219.2 times for 60 s contents, which shows our method is much faster than Shiratori's method [3]. When the music doubles in length, computing time of our method is averagely 2.05 times and that of Shiratori et al. [3] is averagely 2.61 times, inferring that the computational cost increases faster in [3] than ours. Moreover, the computational cost in our method is predictable given the music length and motion genre (see the analysis in Sect. 3.2) while the time varies much in [3] due to the number of candidate motion segments depends on the music and the motion database.

On the other hand, we test the computing time for different segment lengths in the local search, which is performed on a PC with a Core2® Quad 2.83 GHz CPU and a 3.25 GB RAM memory. As shown in Fig. 6, it takes less than 1.0 seconds on average to synthesize a 5 s-segment motion in our method in the worst case, 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.

Table 1 Comparison of computational cost (seconds) between Shiratori's method and our method (global search). The time of the off-line process is excluded.

		Song004		Song091	
		30 s	60 s	30 s	60 s
Break	Shiratori's	735	2103	6039	15027
	global search	27	56	40	80
Indian	Shiratori's	284	757	2025	4888
	global search	3	7	5	9

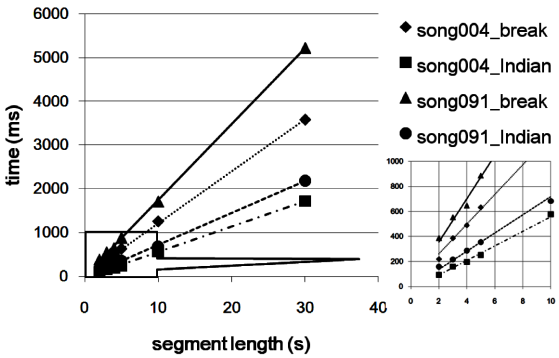

Fig. 6 Comparison of average computational cost (milliseconds) for a segment. Segment length includes 2 s, 3 s, 4 s, 5 s, 10 s, and 30 s (entire length).

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

Benefiting from the high efficiency of our algorithm, we have successfully implemented our system on mobile phones with Android(TM) OS and confirmed that it works well on au(TM)'s IS03 (Qualcomm Snapdragon QSD8650 1 GHz and 512 MB RAM), IS06 (Qualcomm Snapdragon QSD8650 1 GHz and 401 MB RAM) and Google(TM)'s Nexus One (Qualcomm Snapdragon QSD8250 1 GHz and 512 MB RAM).

5.2 Evaluation of Motion Quality

Twelve pieces of music are tested from the above RWC music database [34]. The dancing motions are the same as those in Sect. 5.1. 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. As the target of our application is general consumers, a total of 60 non-expert observers (41 of 60 being females) are asked to evaluate the results in order to know the evaluation results from real users, who are equally separated into the following two groups†. In the first group, we compare the global search-

†For the number of the respondents, because we cannot find the direct reference for that, we refer to those in video quality evaluation [35]. According to the requirement in video quality evaluation [35], at least 13 observers are usually required, which is obeyed in our user study.

Table 2 Summary of T-tests in two groups.

		Q1			Q2			Q3		
		better	same	worse	better	same	worse	better	same	worse
Group 1	global search vs. Shiratori	5	6	1	10	2	0	12	0	0
	local search vs. Shiratori	6	5	1	6	6	0	10	2	0
Group 2	10 s vs. 5 s	1	11	0	0	12	0	1	11	0
	10 s vs. 2 s	1	11	0	1	11	0	4	8	0

ing method and segment-based searching method with Shiratori's method [3], where the beat and intensity features are common to all the methods. In the second group, we focus on the segment-based searching method, testing different lengths of the music segment, namely, 2 seconds, 5 seconds, and 10 seconds, as these can be regarded as a reasonable range in real applications.

In each group, the 30 participants have evaluated the 36 samples by assigning a score for 1 to 5 for the following 3 questions, where "strongly disagree", "somewhat disagree", "neutral", "somewhat agree", and "strongly agree", are denoted by the numbers 1 to 5 in that order. Since the respondents are laypersons, some anchoring videos are provided to tell them the meaning of questions in order to avoid misunderstanding. Although the three questions correspond to the three requirements in our objective function (see Sect. 3.2), they are perceptively important in our application, thus regarded as reasonable in our opinion. Researches [11], [12] strongly suggest the innate beat perception of human being. Intensity perception is confirmed in our interview with professional dancers. And the smoothness is required to avoid the motion artifacts from motion synthesis. By randomly displaying the results, participants do not know which sample is generated by which method. To ensure a better understanding of the questions and the evaluation criteria, some learning materials are displayed to participants as quality anchors before the tests.

- Q1 (beat synchronization): The motion beats are synchronized with those of the music.
- Q2 (intensity matching): The strength of the motion is matched with that of the music.
- Q3 (motion smoothness): The motion is smooth.

Figures 7 and 8 show the mean opinion scores (MOSs, average scores) [36]) from all the participants for all the 12 samples. At the same time, a t-test is performed as shown in Table 2, where the confidence interval is set as 95%. In Group 1, the global search and local search are compared to Shiratori's method respectively. As can be seen, the motion quality by both global search and local search is generally better or competitive to Shiratori's method. Especially, the motion is more smooth in our method. In Group 2, the segment length of 5 s and 2 s are compared to that of 10 s. Basically, there is no significant difference between 10 s segment and 5 s segment in all the three questions. According to our algorithm, there should be no difference for the beat synchronization (Q1) no matter how long the segment is[†]. However, the probability of falling the local minimum should increase when the segment becomes short, influencing the quality of intensity matching and motion smoothness. For the intensity matching, this effect has not been observed in our user study. However, we do observe that the motion smoothness becomes worse in the case of 2 s. Note that one can change the effect by modifying the weights of bi-directional edges. Considering the trade-off with the initial delay, it is recommended that the segment length be set at about 5 seconds.

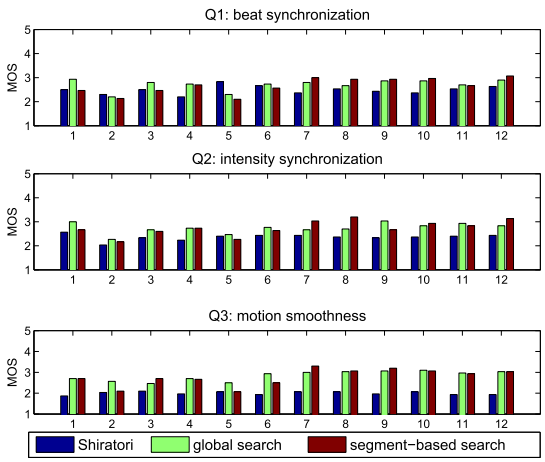

Fig. 7 Mean opinion scores (MOSs) of 12 pieces of music from three different methods including Shiratori's method, global search, and segment-based search (Group 1).

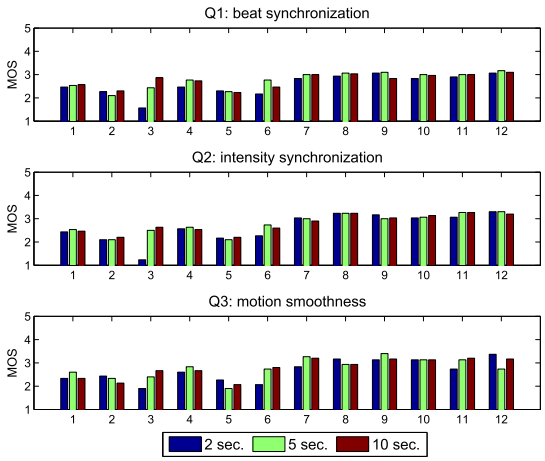

Fig. 8 Mean opinion scores (MOSs) of 12 pieces of music from segment-based search with different segment lengths (Group 2).

[†]Here we assume the detected beats in both music and motion are perfectly accurate. However, the error from detection algorithms is unavoidable although we have especially improved the motion beat detection algorithm [29].

Table 3 Average of mean opinion scores (MOSs) of 12 samples.

method	Q1	Q2	Q3
Shiratori	2.49	2.36	2.00
global search	2.71	2.74	2.84
local search	2.67	2.74	2.78
2 sec.	2.57	2.63	2.66
5 sec.	2.77	2.79	2.78
10 sec.	2.76	2.78	2.79

Although the proposed methods achieve better performance than the conventional one, the average scores from all the methods (See the details in Table 3) are below 3.0, which implies that other factors affect the evaluation. Basically, the professional dancers consider not only the low level features such as beat instants and intensity of the music but also the high level features of music, improvisational performance, and their own style when they dance to the music. It is very challenging to automatically generate so highly complex art form, where we have much future work to do. In addition, there are some factors that are beyond of the scope of this paper. For example, it is known that the CG character's influence is perceptible in animations [37]. In our use study, a very simple model is used with just 3,776 polygons, worsening evaluation scores.

6. Conclusions and Future Work

This paper has proposed a novel scheme for motion synchronization with streaming/non-streaming music using a motion capture database. Specially designed for this purpose, a graph-based representation called *meta motion graph* is constructed on the database beforehand, where necessary features, including both beat and intensity, are embedded. Then, our system can search a best path for music synchronization by global searching and local searching approaches for non-streaming music and streaming music respectively. Furthermore, we have implemented our system on commercially available smart phones and confirmed that it works well on these mobile phones.

Basically, our system can only deal with those songs within a range of about 15% changes in tempo. We do not consider the situation where the tempo is dramatically changed in the case of both streaming and non-streaming music. However, it is possible to deal with variable-tempo music in our framework, which may be our future work. Since we have the technique to detect music beats in variable-tempo music in real time such as Murata et al. [17], a straightforward extension for both streaming and non-streaming music is to use a two-layer graph structure for different motion graphs that cover a wide range of tempos, where the motion graph can be switched freely according to the music tempo. When generating the motions in the case of non-streaming music, we can still use global search method. In the case of streaming music, once the tempo change is detected in music, a delay will happen before transiting to a proper motion graph.

In addition, according to professional dancers, not all the beat instants are known to be of the same importance in terms of human perception. In the future, we plan to estimate the importance of beat frames in both music and motion to make a better match with human perception. Furthermore, it would be preferable to synthesize dancing motion with a finer granularity unit instead of a segment level in a streaming application. For instance, only a short slice of music is increasingly added. However, in such a case, it is necessary to determine the current motion not only by the refreshed music but also by the previous data. Logically, the determination unit is longer than the refreshed unit for better performance, where the Markov model may be used. This is also a part of our future work.

It should be mentioned that our system is based on low level features. A future research direction is toward realistic and well-organized choreographing as done by a skilled choreographer, which requires necessary high level features and rules. Although it is straightforward to adopt additional features in the set of edge weights, it is rather difficult to obtain the high level features unless manual operation is performed due to the so called semantic gap.

Acknowledgments

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/ [34]. 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. Graphics, vol.22, no.3, pp.392–401, July 2003.
[2] G. Alankus, A.A. Bayazit, and O.B. Bayazit, "Automated motion synthesis for virtual choreography," Computer Animation and Virtual Worlds, vol.16, no.3-4, pp.259–271, 2005.
[3] T. Shiratori, A. Nakazawa, and K. Ikeuchi, "Dancing-to-music character animation," Computer Graphics Forum, vol.25, no.3, pp.449–458, July 2006.
[4] E.W. Large, "On synchronizing movements to music," Human Movement Science, vol.19, no.4, pp.527–566, Oct. 2000.
[5] T. Shiratori, A. Nakazawa, and K. Ikeuchi, "Detecting dance motion structure through music analysis," Proc. Sixth IEEE International Conference on Automatic Face and Gesture Recognition, pp.857–862, May 2004.
[6] A.H. Guest, Labonotation: the System of Analyzing and Recording Movement, Taylor and Francis, 1987.
[7] K. Onuma, C. Faloutsos, and J.K. Hodgins, "FMDistance: A fast and effective distance function for motion capture data," Proc. EUROGRAPHICS 2008 Short Papers, 2008.
[8] L. Kovar, M. Gleicher, and F. Pighin, "Motion graphs," ACM Trans. Graphics, vol.21, no.3, pp.473–482, July 2002.
[9] O. Arikan and D.A. Forsyth, "Interactive motion generation from examples," ACM Trans. Graphics, vol.21, no.3, pp.483–490, 2002.
[10] J. Lee, J. Chai, P.S.A. Reitsma, J.K. Hodgins, and N.S. Pollard, "Interactive control of avatars animated with human motion data," ACM Trans. Graphics, vol.21, no.3, pp.491–500, July 2002.
[11] J. Phillips-Silver and L.J. Trainor, "Feeling the beat: Movement

influences infant rhythm perception," Science, vol.308, no.5727, p.1430, June 2005.
[12] I. Winkler, G.P. Haden, O. Ladinig, I. Sziller, and H. Honing, "Newborn infants detect the beat in music," Proc. National Academy of Sciences USA (PNAS), vol.106, no.7, pp.2468–2471, 2009.
[13] X.S. Hua, L. Lu, and H.J. Zhang, "Automatic music video generation based on temporal pattern analysis," MULTIMEDIA '04: Proc. 12th annual ACM international conference on Multimedia, pp.472–475, 2004.
[14] J. Wang, C. Xu, E. Chng, L. Duan, K. Wan, and Q. Tian, "Automatic generation of personalized music sports video," MULTIMEDIA '05: Proc. 13th annual ACM international conference on Multimedia, pp.735–744, New York, NY, USA, 2005.
[15] M. Goto and Y. Muraoka, "A virtual dancer "cindy" — interactive performance of a music-controlled cg dancer," Proc. Lifelike Computer Characters '96, p.65, 1996.
[16] D. Grunberg, R. Ellenberg, Y. Kim, and P. Oh, "Creating an autonomous dancing robot," Proc. 2009 International Conference on Hybrid Information Technology, pp.221–227, 2009.
[17] K. Murata, K. Nakadai, K. Yoshii, R. Takeda, T. Torii, H.G. Okuno, Y. Hasegawa, and H. Tsujino, "A robot uses its own microphone to synchronize its steps to musical beats while scatting and singing," 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp.2459–2464, 2008.
[18] P.S.A. Reitsma and N.S. Pollard, "Evaluating motion graphs for character animation," ACM Trans. Graphics, vol.26, no.4, article 18, 2007.
[19] M. Mizuguchi, J. Buchanan, and T. Calvert, "Data driven motion transitions for interactive games," Proc. EUROGRAPHICS 2001 short papers, 2001.
[20] M. Gleicher, H.J. Shin, L. Kovar, and A. Jepsen, "Snap-together motion: assembling run-time animations," I3D '03: Proc. 2003 symposium on Interactive 3D graphics, pp.181–188, 2003.
[21] A. Safonova and J.K. Hodgins, "Construction and optimal search of interpolated motion graphs," ACM Trans. Graphics, vol.26, no.3, article 106, July 2007.
[22] L. Zhao and A. Safonova, "Achieving good connectivity in motion graphs," Graphical Models, vol.71, no.4, pp.139–152, 2009.
[23] C. Ren, L. Zhao, and A. Safonova, "Human motion synthesis with optimization-based graphs," Computer Graphics Forum, vol.29, no.2, pp.545–554, 2010.
[24] P. Beaudoin, S. Coros, M. van de Panne, and P. Poulin, "Motion-motif graphs," Proc. 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp.117–126, 2008.
[25] 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.
[26] O. Arikan, D.A. Forsyth, and J.F. O'Brien, "Motion synthesis from annotations," ACM Trans. Graphics, vol.22, no.3, pp.402–408, 2003.
[27] J. Lee and K.H. Lee, "Precomputing avatar behavior from human motion data," Proc. 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp.79–87, 2004.
[28] CMU Motion Capture Database, http://mocap.cs.cmu.edu/, 2003.
[29] 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.
[30] J. Wang and B. Bodenheimer, "An evaluation of a cost metric for selecting transitions between motion segments," SCA '03: Proc. ACMSIGGRAPH/Eurographics Symposium on Computer Animation, pp.232–238, 2003.
[31] B.J.H. van Basten and A. Egges, "Evaluating distance metrics for animation blending," Proceedings of the 4th International Conference on Foundations of Digital Games, pp.199–206, 2009.
[32] S. Dixon, "Automatic extraction of tempo and beat from expressive performances," J. New Music Research, vol.30, no.1, pp.39–58, March 2001.
[33] ITU-R, Algorithms to measure audio programme loudness and true-peak audio level, 2006.
[34] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "Rwc music database: Popular, classical, and jazz music databases," ISMIR '02: Proc. 3rd International Conference on Music Information Retrieval, pp.287–288, 2002.
[35] H.E. Khattabi, A. Tamtaoui, and D. Aboutajdine, "Video quality assessment measure with a neural network," Int. J. Comput. Inf. Eng., vol.4, no.3, pp.167–171, 2010.
[36] S. Winkler, Digital video quality: vision models and metrics, John Wiley & Sons, 2005.
[37] J. Hodgins, J. O'Brien, and J. Tumblin, "Perception of human motion with different geometric models," IEEE Trans. Vis. Comput. Graph., vol.4, no.4, pp.307–316, 1998.

Jianfeng Xu received the B.S. (with honor) and the M.S. degrees from Tsinghua University, China, in 2001 and 2004 respectively and the Ph.D. degree from the University of Tokyo, Japan, in 2007. He has been working at KDDI R&D Laboratories, Inc. since 2007 and now is a research engineer in Media and HTML5 Application Laboratory. His research interests include generation, analysis, and re-use of motion capture data.

Koichi Takagi received his M.E. degree from the Tokyo Institute of Technology in 1998. He has been working at Kokusai Denshin Denwa since 1998 and now is a manager in Technology Strategy Department at KDDI Corporation.

Shigeyuki Sakazawa received the B.E., M.E., and Ph.D degrees from Kobe University, Japan, all in electrical engineering, in 1990, 1992, and 2005 respectively. He joined Kokusai Denshin Denwa (KDD) Co. Ltd. in 1992. Since then he has been with its R&D Division, and now he is a senior manager of Media and HTML5 Application Laboratory in KDDI R&D Laboratories, Inc., Saitama, Japan. His current research interests include video coding, video communication system, image recognition and CG video generation. He received the best paper award from IEICE in 2003.