Dance Experience System Using Multiple Kinects

Seongmin Baek and Myunggyu Kim

***Abstract*—We present a new method for comparing users' motions captured in real time by multiple Kinect sensors with an expert's movements stored in a DB so as to help users experience and learn how to dance. Recently, lots of experience games where users copy some motions to score have been developed. However, it is difficult to collect users' joint data or to clearly compare movements with one sensor because of blocked body parts and unsuccessful tracking. Therefore, such games cannot be applicable to learning beyond simple in-game experiences. As an alternative, this study proposes a method for using multiple Kinect sensors to combine skeletal data and thus to retrieve motions. Also, we propose a method for directly comparing postures between characters of different body sizes. The direct comparison of postures ensures accuracy as no motion is adapted in the process. The proposed dance experience learning system is suitable for learning how to dance as it is easy to implement, features real-time posture comparison and displays the results of important body parts compared.**

***Index Terms*—Experience, motion capture, multiple kinects, posture comparison.**

I. Introduction

Marker-less motion capture system has long been investigated in the field of computer vision, with most studies focusing on using multiple cameras to capture users' movements or appearances. Lately, with the advent of Microsoft Kinect, inexpensive real-time capture systems have drawn attention, followed by extensive research efforts. Kinect provides RGB images and depth data at the same time and is applied to experience games drawing on a single sensor for extracting users' postures in real time.

However, when a single sensor is used, a player must stand facing the sensor whilst all joints must be visible to the sensor so that joint data can be captured properly. Still, when some body parts are blocked, e.g. the player turning sideways, depth data cannot be entered, leading to unsuccessful posture estimation. Due to such a drawback, experience games based on a single sensor rely on only the body parts visible to the sensor and measure their scores.

To address the issue of some body parts being blocked, more than two sensors need be used so as to minimize the self-occluded body parts. Recently, lots of methods for using many Kinect sensors have been proposed. Auvinet *et al.* [1] proposed a method for generating 3D body shapes based on visual hulls. Berger *et al.* [2] proposed a method for using four Kinects to extract silhouettes and capture motions. Zhang *et al.* [3] applied particle filtering and partition sampling techniques to track postures. These proposed methods draw on not skeletal data but the optimization process with silhouette or template matching to estimate postures. Williamson *et al.* [4] proposed a soldier training system using multiple Kinects capable of capturing motions even when users rotate 360 degrees. Asteriadis *et al.* [5] proposed a method for applying energy functions to estimate joint positions.

The proposed system here helps to experience and learn dance motions by capturing users' movements in real time, comparing them with those of experts' and displaying the results. Multiple Kinects are used to capture motions. Here, Kinect sensors need be calibrated and synchronized. As Kinects cannot tell users' front-facing from back-facing, joint position data can be mixed with left and right sides being reversed, ending up in some awkward motions (see Fig. 6).

Comparing the captured joint data with the experts' postures when the body sizes of characters differ is another issue that need be addressed. Motion retargeting has been proposed as a method for adapting motions between characters of different body sizes. However, as the motions are altered, they can diverge from actual postures of users. Therefore, a method is required for comparing postures between characters of different sizes without altering motions.

Section II of this paper describes overview of dance experience system. Section III deals with calibration between multiple Kinect sensors and data synchronization. Section IV elucidates how to combine data captured by multiple Kinect sensors. Section V elaborates on how to compare postures between characters of different body sizes, followed by results and conclusions.

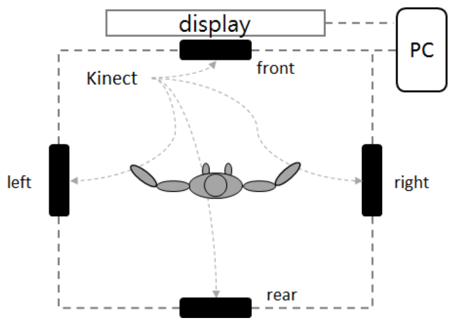


Fig. 1. System setup with four Kinects.

II. System Overview

Currently, Kinect SDK 1.8 enables 4 Kinects to be connected to a computer (Windows® 7) to easily set up a

Manuscript received October 8, 2014; revised December 12, 2014. This research was supported by the Ministry of Science, ICT & Future Planning (MSIP) and Institute for Information & communications Technology Promotion (IITP) in the digital content source technology development program 2013. [Project Title: Development of Performance Analysis and Generation Technology for Experiencing, Learning and Creating Traditional and Popular Dances, Project No. 10044309].

The authors are with Visual Contents Research Department, Electronics and Telecommunications Research Institute, Daejeon, Korea (e-mail: {baeksm, mgkim}@etri.re.kr).

system. As in Fig. 1, a user's motion is captured in all directions (360 degrees) by installing Kinects in the front and rear and on the left and right. To match the coordinates of sensors, calibration is carried out initially based on the user's joint data. As the sequential skeletal data from multiple Kinects lead to time gaps, a synchronization module is used to match the time. A main sensor is chosen based on the joint status tracked by each sensor. Then, an initial reference model is generated based on the joint data captured by the main sensor. Finally, a 3D posture is retrieved based on the reference model by searching and combining joint data captured by other sensors. The captured user posture is compared in real time with that of an expert, with the results of important parts compared being displayed to the user. The entire process is presented in Fig. 2.

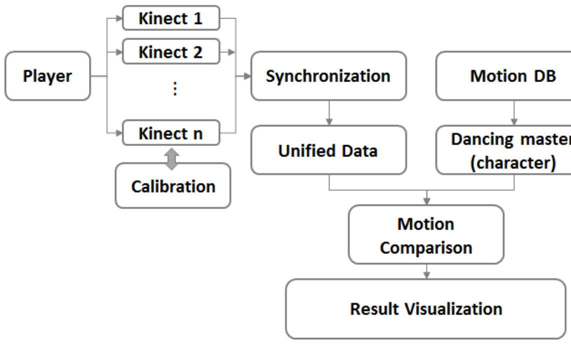


Fig. 2. System process.

III. Calibration and Synchronization

Each of the multiple Kinects has a coordinate system. Thus, those different coordinate systems need be unified. Here, the unified coordinate system need be set up based on the front sensor, not the absolute coordinate calculation. Then, the other sensors' coordinate systems need be converted into a reference coordinate system by calculating the transformation matrix. For calibration, a large planar rectangle may be used [1], [6]. Here, however, the skeletal data are directly used for easiness and convenience to take advantage of Kinects.

The user stands facing a direction at an angle of about 45 degrees so that all body joints (20 joints) can be captured by two Kinects. To have the joints well tracked, the user should stand with arms and legs outspread. Among the 20 joints captured by each Kinect, the direction can be calculated using the triangle formed by 3 joint positions, i.e. Hips, left hip and right hip (see Fig. 3).

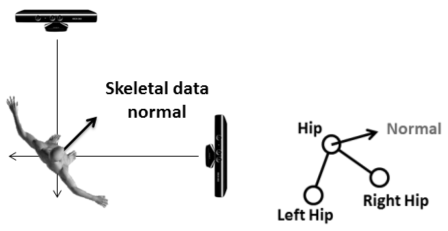


Fig. 3. Calibration process and normal vector for rotation.

The direction calculated in each sensor should be congruent with that of the reference coordinate system. Therefore, the angle difference between the two normal vectors is used to calculate the rotation matrix. The rotation matrix used for matching the normal vectors is same as the one for matching the coordinate system of a sensor against the reference coordinate system. The rotation matrix is applied to the 20 joint positions and matched against the reference coordinate system. Then, the difference in joint positions between the two sensors is calculated to yield the translation vector. The calibration process yields the transformation matrix consisting of the rotation matrix and translation vector.

The aforementioned process is repeated for each pair of two sensors. As the front camera's coordinate system is set up as the reference, both right and left cameras need be matched against the front camera's coordinate system, respectively, to calculate the transformation matrix. The rear camera's coordinate system is converted into that of the right camera and matched against the front camera's coordinate system by applying the right-front transformation matrix. Fig. 4 shows the calibration process and results. As is well known, when more than two Kinect sensors are used, interference issues may arise. Still, no significant interference occurs in the process of capturing the skeletal data in this trial.

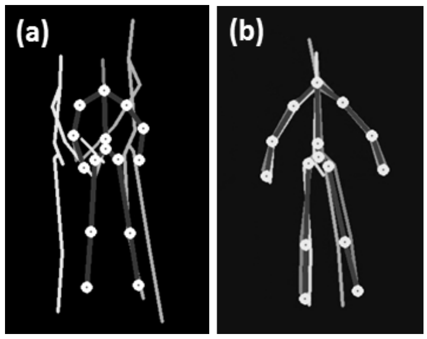
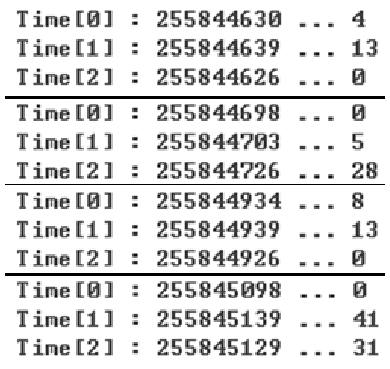
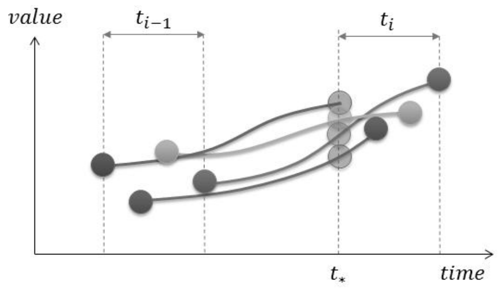


Fig. 4. Result of calibration (a) before, (b) after.



```
Time[0] : 255844630 ... 4
Time[1] : 255844639 ... 13
Time[2] : 255844626 ... 0
Time[0] : 255844698 ... 0
Time[1] : 255844703 ... 5
Time[2] : 255844726 ... 28
Time[0] : 255844934 ... 8
Time[1] : 255844939 ... 13
Time[2] : 255844926 ... 0
Time[0] : 255845098 ... 0
Time[1] : 255845139 ... 41
Time[2] : 255845129 ... 31
```

(a). Time of capture in three Kinect and delay time.



(b). Interpolation using spline curve.
Fig. 5. Synchronization method.

The multiple Kinect sensors connected to a PC capture data in sequence. Therefore, a delay occurs by dozens of milliseconds while each sensor captures motions (Fig. 5a). When joints move fast, the joint positions captured by each sensor may vary. Kinect SDK does not support synchronization for multiple Kinects but can recognize the time of capture in each sensor by milliseconds. Therefore, we propose a synchronization method via data interpolation based on the earliest point of capture time by recording the time upon each Kinect capturing data. To calculate new joint values, a spline interpolation is applied to the current point of time input from each sensor (t_i) and the joint position captured at a previous point of time (t_{i-1}) in line with the synchronization time (t_*) (Fig. 5b). Here, the Ferguson curve is used for the spline interpolation (1). Joint data can be calculated more accurately with calibration and synchronization.

$$\begin{gathered}P(t_*)=\left(1-3t_*^2+2t_*^3\right)P(0)+\left(3t_*^2-2t_*^3\right)P(1)+\\ \left(t_*-2t_*^2+t_*^3\right)P'(0)+\left(-t_*^2+t_*^3\right)P'(1)\\ \left(t_{i-1}<t_*\le t_i\right)\end{gathered} \tag{1}$$

IV. Data Merging

The data coming from multiple sensors are merged into the skeletal data as follows. When Kinect SDK successfully tracks joint positions, it returns TRACKED, whereas it returns NOT_TRACKED when it fails to track the joints. The number of joints tracked by each sensor is counted. Then, the sensor that successfully tracks the largest number of joints is set as the main sensor. As it is easier for Kinect sensors to track joint motions when the player stands facing them, the sensor placed to face the player is highly likely to become the main sensor.

It is, however, hard to select the main sensor when only the upper half of the body rotates and thus faces a different direction from the one that the lower half of the body faces. Thus, this study divides the body data into five parts and chooses a main sensor for each part (Fig. 6). After a reference model is generated based on the joint values tracked by each main sensor, the joint values tracked by the other sensors are weighted and merged. As the joints can be tracked more accurately when the player stands facing the sensors, the weights are determined in line with the direction the player faces. Initially, the player stands facing the front sensor, which corresponds to the Z axis in the reference coordinate system. When the user rotates, the vector of the root joint direction is calculated to estimate the direction to which the user is rotating in each frame.

Kinect sensors track joints, assuming the user always stands facing them, without detecting whether the user is actually facing forward or backward. For example, when the player stands facing the front sensor, the rear sensor thinks of the player as looking backward. As each sensor assumes the player stands facing it, the right-hand joint position tracked by the front sensor appears as the left-hand joint position in the rear sensor (see Fig. 7). Problems arise if the inconsistent data are merged. To solve this problem, before the data are merged, the ranges of joint positions are computed and thus only the data of joints placed in approximate positions are merged. In searching the range of joint positions, the joints positioned closer to those in the reference model generated from the main sensor constitute the data of identical joints. This applies only to the joints in the arms and legs where it is necessary to tell apart from left to right, excluding the joints in the torso. As in Fig. 8, based on the reference model, the distances to the joints on the left and right are compared with each other to select the one at the shorter distance. In case a joint cannot be tracked in the reference model, its position is estimated with reference to its parent's or previous position.

Joint position errors from Kinect sensors and other errors resulting from data merging may cause noise. To reduce the noise of joint positions input in real time, motion filtering based on the Kalman filter used by Shin *et al.* [7] is applied here.

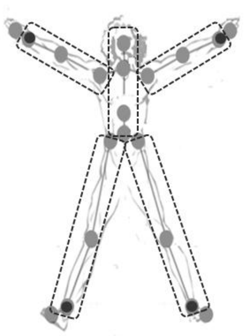


Fig. 6. Skeletal data: five parts (torso, arms, and legs).

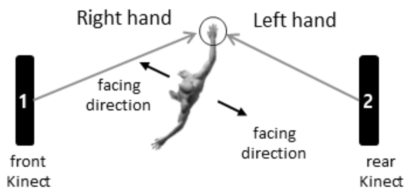
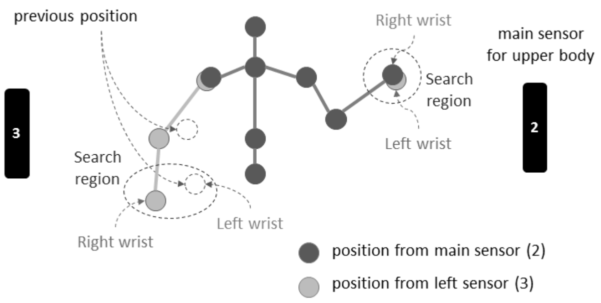


Fig. 7. Problem of facing forward or backward.


previous position
Right wrist
main sensor for upper body
Search region
3
2
Left wrist
Search region
Left wrist
Right wrist
● position from main sensor (2)
● position from left sensor (3)
Fig. 8. The method of searching the range of joint positions.

V. Motion Comparison

If the articulated figures have identical structure and same segment lengths, postures are easy to compare between characters such as the comparison of joint positions and velocities. The dance learning system compares postures between an expert and a user, whose joint structures are identical but whose body lengths are different. Thus, there is no point in directly comparing joint positions and velocities between the two characters. Accordingly, an alternative is needed to compare postures between characters of different body sizes.

As a rule, motion retargeting is applied to characters of different sizes in character animation. Gleicher [8] used a space-time constraint solver to apply motions to characters of different body lengths. This method, however, is not suitable for a real-time system. Tak *et al*. [9] proposed an online retargeting method based on inverse kinematics. This method is suitable for a real-time system and reduces noise. In dancing, however, overall postures are considered more important than the positions of end-effectors. Computer puppetry [7] takes concept of important aspects of the motion to modify motions by applying inverse kinematics when the positions of end-effectors are important.

Still, when retargeting is applied to the comparison of postures, motions need be modified in line with body sizes, ending up in different motions from the original ones. Hence, the present study proposes a method for directly comparing postures between characters of different body sizes. To compare postures, the posture vectors of important body parts need be defined [10]. There are five posture vectors for the torso, each arm and each leg (see Fig. 9). The vectors link each root joint to the neck joint, the shoulder joint to the wrist joint and the hip joint to the ankle joint.

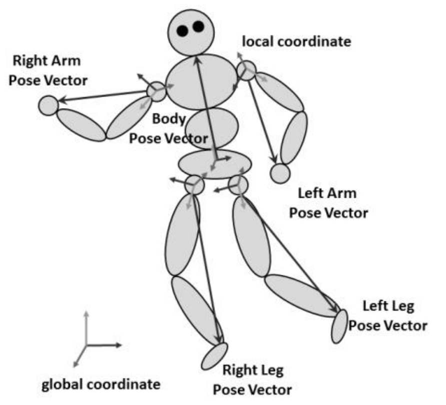


Fig. 9. Posture vector representation of the skeleton.

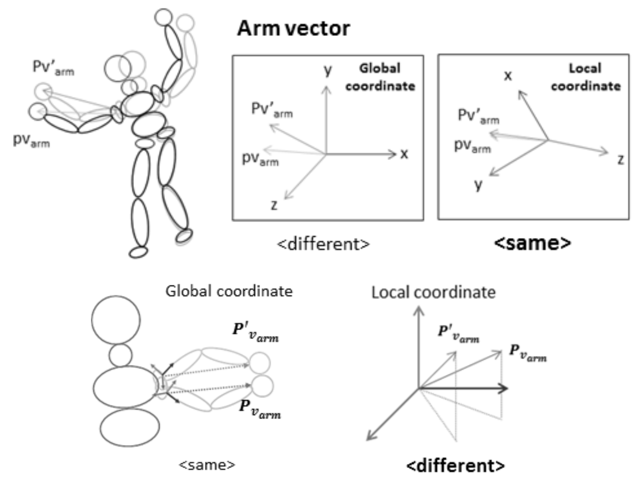
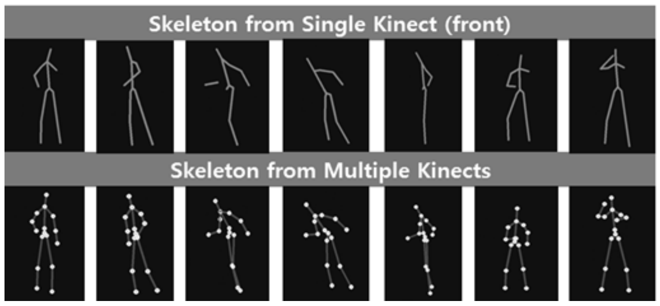


Fig. 10. Comparison of global with local coordinate for dance pose.

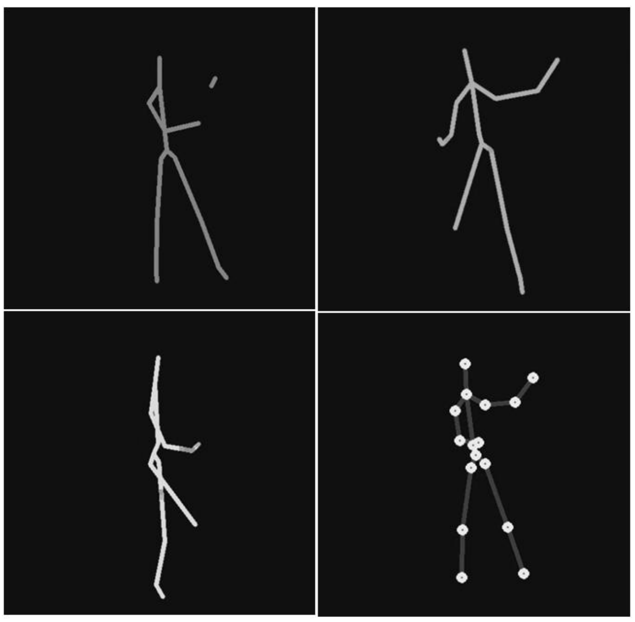
When body sizes differ, comparing joint positions is meaningless. Likewise, when body ratios differ, there is no point in comparing joint angles because postures vary even at congruent angles. Yet, when posture vectors are used for comparison, the problems arising from using body sizes and ratios can be solved. The posture vectors are calculated based on a local, not global, coordinate system. Unlike in a global coordinate system, incorrect parent joints will not affect the child ones in a local coordinate system. For example, in case the spine joint tilts a bit further to the left in a posture, all joints in the upper half of the body will vary in the global coordinate system, whereas the arm postures are found to be correct with only the waist part being different in the local coordinate system (Fig. 10a). This is equivalent to correcting the mistakes only in actual dance instructions. Also, the local coordinate system can detect variations by rotations. For example, when the shoulder joints rotate a bit further with the wrists being positioned in exactly the same way, the local coordinate system can determine that the varied direction of the posture vector has led to the incorrect shoulder posture (see Fig. 10b).

To compare the posture vectors, the directions and lengths of vectors are calculated, so as to determine whether a posture is similar or incorrect based on the variations in directions and lengths. Similar vector directions and lengths lead to similar postures. As the body sizes and ratios differ between characters being compared, the length of the posture vector is estimated to be a relative length to the body size. For example, the arm posture vector is divided by the whole length of the arm to get the vector length.

The proposed method of comparing posture vectors based on the local coordinate system is appropriate for motions where postures are important as in dancing, is easy to implement and is capable of comparing postures regardless of body sizes.


Skeleton from Single Kinect (front)
Skeleton from Multiple Kinects

(a). Result from a single front Kinect (top) and Multiple Kinects (bottom).



(b). Front (top-left), right (top-right), left (bottom-left) Kinect data, and unified data from multiple Kinects (bottom-right).
Fig. 11. Visualization result from a single Kinect and multiple Kinects.

VI. Results

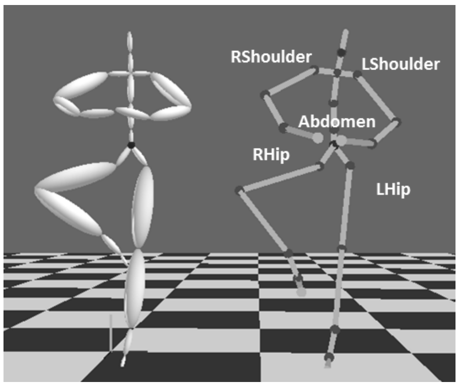
Fig. 11 shows the results from a single Kinect and multiple Kinects, respectively. As seen in the Fig. 11, a single Kinect often fails to track joint positions because some body parts are blocked, and thus cannot represent those blocked parts. In contrast, multiple Kinects show a much higher performance in capturing users' motions.

Table I summarizes the percentage of joints (e.g. elbow, wrist, knee, and ankle) tracked by each Kinect compared with that of joints tracked by the multiple Kinects while capturing users' motions (3,000 frames).

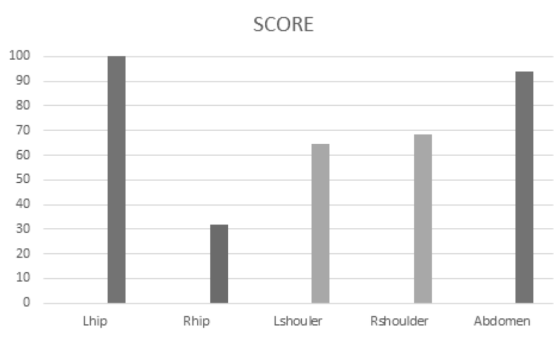
Fig. 12 illustrates two postures and the result of postures compared. The posture vectors of the torso, arms and legs were compared. Then, a graph is plotted of the degree of similarity in postures (see Fig. 12b). As it is easy to see incorrect parts from the graph, the proposed system is helpful for users to learn how to dance.

TABLE I: Example of The Number of Joints Tracked by Kinect

	Joints	Front	Right	Left	**Unified**
UPPEr	L Elbow	82.0%	74.6%	83.4%	**99.2%**
	L Wrist	85.7%	74.8%	88.4%	**99.1%**
	R Elbow	81.4%	83.8%	68.3%	**99.1%**
	R Wrist	85.8%	83.7%	70.6%	**98.8%**
Lower	L Knee	94.7%	87.8%	86.5%	**99.7%**
	L Ankle	84.3%	78.9%	88.2%	**99.4%**
	R Knee	93.6%	83.6%	88.3%	**99.9%**
	R Ankle	80.5%	67.9%	86.7%	**99.1%**
Average		**86.0%**	**70.0%**	**82.6%**	**99.3%**



(a) Differently sized characters that have taken similar poses.



(b) Score corresponding to the pose.

Fig. 12. The result of comparison between poses.

In this paper, we propose a method for helping users to experience and learn how to dance. Multiple Kinect sensors are used to capture users' motions in real-time and thus to collect joint data whilst minimizing any body parts blocked.

The proposed method is distinct from the conventional dance games using a single sensor. As dance games compare just a few joints facing sensors, they are not suitable for learning how to dance. Also, in comparing dance movements, the proposed system can compare joint posture vectors, not just a few joint positions, enabling a direct and seamless comparison of postures between characters of different body sizes. The proposed method is suitable for a real-time dance movement learning system, and extensively applicable to other sports movements as well as dancing.

References

[1] E. Auvinet, J. Meunier, and F. Multon, "Multiple depth cameras calibration and body volume reconstruction for gait analysis," in *Proc. the 2012 11th International Conference on, Information Science, Signal Processing and their Applications (ISSPA)*, pp. 478-483, 2012.

[2] K. Berger, K. Ruhl, Y. Shcroeder, C. Bruemmer, and A. Scholz, "Markerless motion capture using multiple color-depth sensors," *Vision, Modeling, and Visualization(VMV)*, 2011, pp. 317-324.

[3] L. Zhang, J. Sturm, D. Cremers, and D. Lee, "Real-time human motion tracking using multiple depth cameras," in *Proc. the International Conference on Intelligent Robot Syystems (IROS)*, 2012, pp. 2389-2395.

[4] B. Williamson, J. L. Viola, T. Roberts, and P. Garrity, "Multi-kinect tracking for dismounted soldier training," in *Proc. the Interservice/Industry Training, Simulation and Education Conference (I/ITSEC)*, Orlando, Florida, USA, December 2012, pp. 3-5.

[5] S. Asteriadis, A. Chatzitofis, D. Zarpalas, D. S. Alexiadis, and P. Daras, "Estimating human motion from multiple Kinect sensors," in *Proc. the 6th International Conference on Computer Vision/Computer Graphics Collaboration Techniques and Applications*, ACM, 2013, pp. 3-8.

[6] D. L. A. Cruz, "Calibration of a multi-kinect system," M.S. thesis, Tampere University of Technology, Finland, 2012.

[7] H. J. Shin, J. Lee, S. Y. Shin, and M. Gleicher, "Computer puppetry: An importance-based approach," *ACM Transactions on Graphics (TOG)*, vol. 20, no. 2, pp. 67-94, April 2001.

[8] M. Gleicher, "Retargetting motion to new characters," in *Proc. the 25th Annual Conference on Computer graphics and Interactive techniques*, July 1998, pp. 33-42.

[9] K. Choi and H. Ko, "On-line motion retargeting," in *Proc. Seventh Pacific Conference on, IEEE* the *Computer Graphics and Applications*, 1999, pp. 32-42.

[10] S. Baek and M. Kim, "Dance motion generation with pose constraints," *Advances in Sport Science and Computer Science*, pp. 281-289, 2013.

Seongmin Baek was born in 1973 in Korea. He received his MS in computer science (virtual reality) from Pohang University of Science and Technology (POSTECH), Korea in 2001.

He is working as a senior member of engineering staff at Visual Contents Research Department of ETRI with interest in Digital Contents, Animation, and Physics.

Myunggyu Kim was born in 1967 in Korea. He received his Ph.D. from University of Maryland at College Park (Department of Physics). He is working as a principal research scientist at Visual Contents Research Department of ETRI. And his research interest include network, simulation, and physics.