Received May 16, 2017, accepted June 11, 2017, date of publication July 4, 2017, date of current version July 24, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2723039

Implementation of a Virtual Training Simulator Based on 360° Multi-View Human Action Recognition

BEOM KWON[1], JUNGHWAN KIM[1], KYOUNGOH LEE[1], YANG KOO LEE[2], SANGJOON PARK[2], AND SANGHOON LEE[1], (Senior Member, IEEE)
[1]Department of Electrical and Electronics Engineering, Yonsei University, Seoul 120-749, South Korea
[2]Electronics and Telecommunications Research Institute, Daejeon 305-700, South Korea

Corresponding author: Sanghoon Lee (slee@yonsei.ac.kr)

This work was supported in part by the ICT R&D Program of the MSIP/IITP Development of ODM Interactive Software Technology supporting Live-Virtual Soldier/Exercises under Grant R0101-15-0168 and in part by the Institute for Information and Communications Technology Promotion Grant, Korea Government, SIAT CCTV Cloud Platform, under Grant R7120-17-1007.

ABSTRACT Virtual training has received a considerable amount of research attention in recent years due to its potential for use in a variety of applications, such as virtual military training, virtual emergency evacuation, and virtual firefighting. To provide a trainee with an interactive training environment, human action recognition methods have been introduced as a major component of virtual training simulators. Wearable motion capture suit-based human action recognition has been widely used for virtual training, although it may distract the trainee. In this paper, we present a virtual training simulator based on 360° multi-view human action recognition using multiple Kinect sensors that provides an immersive environment for the trainee without the need to wear devices. To this end, the proposed simulator contains coordinate system transformation, front-view Kinect sensor tracking, multi-skeleton fusion, skeleton normalization, orientation compensation, feature extraction, and classifier modules. Virtual military training is presented as a potential application of the proposed simulator. To train and test it, a database consisting of 25 military training actions was constructed. In the test, the proposed simulator provided an excellent, natural training environment in terms of frame-by-frame classification accuracy, action-by-action classification accuracy, and observational latency.

INDEX TERMS Human action recognition, Kinect sensor, virtual training simulator.

I. INTRODUCTION

In recent years, the demand for Virtual Reality (VR) services has risen due to advancements in VR technology. The market for VR services is expanding beyond that for gaming [1], education [2], and rehabilitation services [3]. In this vein, the use of VR technology in virtual training has attracted considerable research attention. To reduce the cost of training and improve its the effectiveness, various industries, such as the military and firefighting, have attempted to develop virtual training simulators. To provide a trainee with an interactive and immersive environment, it is necessary for a virtual training simulator to recognize actions performed by the trainee. For this reason, human action recognition methods have been introduced as a major component of successful virtual training simulators [4]–[6].

Most research on virtual training simulators thus far has been based on wearable motion capture suit-based human action recognition to obtain precise information concerning human action, which is needed to synchronize the content in accordance with the action. For example, in [7], a dismounted soldier training system (DSTS) was introduced, where it is mandatory for a trainee to wear a motion capture suit consisting of a head-mounted display and several motion capture sensors. In [8], a wearable simulation interface for military training was proposed. For training, the trainee had to attach three-axis motion sensors to his/her body to track movement. Quantum3D's Expedition, one of the most well-known commercial products for virtual training simulators, also requires the trainee to wear a motion capture suit to record motion information [9]. However, this can distract the trainee from the virtual training.

Moreover, the motion capture suit provides accurate joint position data only in the early stage of the training because small positioning errors generated by the motion capture sensors accumulate over the training period. Since training time can be extensive depending on the training scenario, error accumulation may result in inaccurate pose estimation and action recognition, which can degrade the performance of the simulator.

To overcome this drawback, several studies on virtual training simulators have been conducted using a motion capture camera. Furthermore, most research using this approach has employed a Kinect sensor because of its low price and acceptable accuracy for data concerning the human skeleton [10]. For example, Wang *et al.* [11] proposed an intelligent solider combat training (ISCT) system consisting of a Kinect sensor, a projection screen, a projector, and a BB gun mounted on a motion platform. In the ISCT system, the trainee's position and posture are captured by the Kinect sensor. According to this information, the virtual enemy can disappear or appear on the projection screen during training. In [12], a virtual assembly training system was proposed consisting of a screen monitor and a Kinect sensor mounted on a screen monitor. The trainee stood in the front of the monitor and practiced virtual assembly training. During the training, the trainee's hand gestures were recorded by the Kinect sensor. In [13], a virtual snowboard-training simulator was proposed where the trainee's pose was captured using a Kinect sensor, following which a virtual coach showed an expert's pose based on the trainee's pose data. In [14], a virtual rehabilitation training system for people with restricted mobility was introduced, where a Kinect sensor was used to track the movements of the user. The user could also watch the movement of his/her character in virtual worlds. In [15], a firearms training simulator using a Kinect sensor was introduced. The Kinect sensor was used to recognize the trainee's gesture. Based on the recognized gesture, the simulator provided a gunshot target on the screen monitor.

Since the trainee in [11]–[15] did not need a wearable motion capture suit, he/she could focus on the training. However, the Kinect sensor captures human skeletal data under the assumption that the user is facing it. If the user is not, it is difficult to guarantee that it achieves an acceptable degree of accuracy on human skeletal data. Further, in [11]–[15], the Kinect sensor was in a fixed position. Therefore, the trainee had to stand facing the sensor during his/her training in [11]–[15].

To exploit this advantage of the Kinect sensor and overcome the aforementioned inconvenience, we explore the use of multiple Kinect sensors for human action recognition in virtual training. The main goal of this simulator is to provide a realistic environment while increasing the degrees of freedom of user mobility by implementing 360° multi-view human action recognition. Each Kinect sensor provides data concerning the human skeleton with acceptable accuracy in real time under the assumption that the user is facing it. Accordingly, to use this feature and capture the entire body of the

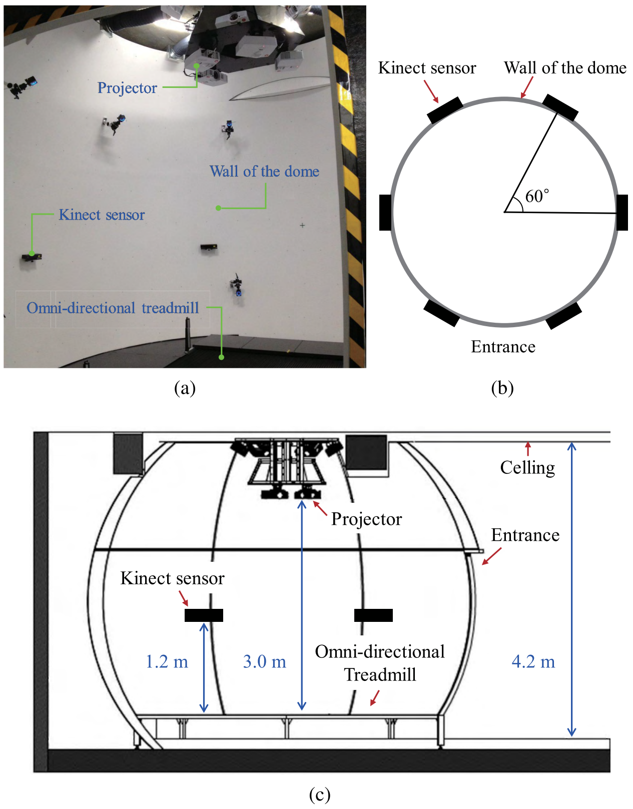


FIGURE 1. Interior (a), top view (b), and blueprint (c) of the studio.

trainee without occlusion, multiple Kinect sensors are used in the proposed simulator. Nevertheless, there are practical issues that need to be addressed: how do we track the Kinect sensors that the user is facing (front-view Kinect sensors), how do we unite multiple skeletal data obtained from multiple Kinect sensors into one skeletal data item, how do we render the skeleton data invariant to the body size and orientation of the trainee, what kind of features and classifier need to be used to recognize the trainee's action? To solve these problems, we provide an effective solution verified through a rigorous practical test.

For a realistic and immersive virtual training environment, we built the studio shown in Fig. 1. It consisted of six Kinect sensors, eight projectors, and an omni-directional treadmill in a dome-shaped structure. The six Kinect sensors were mounted on the wall of the dome in a ring to capture the entire body of the trainee. The configuration of the 360° multiple Kinect sensors enables the proposed simulator to capture more precise skeleton data for the trainee, regardless of his/her direction of motion. The skeleton data is used to recognize the trainee's actions which are then synchronized with the VR contents projected onto the wall of the dome through the projectors. The omni-directional treadmill is controlled to place the trainee at the center of the dome to maximize the degrees of freedom for user mobility. Therefore, the trainee is unconcerned about location, and can move wherever he/she wants over the omni-directional treadmill.

He/She hence undergoes more realistic and immersive virtual training.

This paper makes the following main contributions: (1) *Implementation of a practical virtual training simulator using multiple Kinect sensors.* To this end, 360° multi-view human action recognition is presented that contains coordinate system transformation, front-view Kinect sensor tracking, multi-skeleton fusion, skeleton normalization, orientation compensation, feature extraction, and classifier modules. (2) *A more realistic and immersive virtual training experience.* Compared to conventional methods, the proposed virtual training simulator is based on 360° multi-view human action recognition. Therefore, it is possible for it to capture the dynamic action of the trainee, and the recognized action is synchronized with VR contents. It enables the trainee to experience realistic and immersive virtual training. (3) *A thorough practical test to highlight the usefulness of the proposed virtual training simulator.* To achieve this, we built a database consisting of 25 military training actions. The proposed virtual training simulator provided an excellent, natural training environment in terms of frame-by-frame classification accuracy, action-by-action classification accuracy, and observational latency on the military training action database.

II. RELATED WORK

A. SENSOR-BASED HUMAN ACTION RECOGNITION

Previous studies on sensor-based human action recognition were conducted using one accelerometer [16]–[18]. In [16], the acceleration signal was obtained from an accelerometer attached to the chest of the user. From the acceleration signal, the authors found that a state of weightlessness obtained for a short while when the user ran and jumped. Based on this, they proposed a weightlessness feature to recognize walking, jumping, still (not moving), and running. In [17], classification accuracy according to the attached position of the accelerometer was analyzed for 15 actions. Furthermore, the simulation results indicated that the classification results can be varied depending on the position of the accelerometer, even for the same action. To solve this problem, Khan *et al.* [18] proposed a two-level classification method. The method first classified whether the accelerometer was attached to the upper or lower body using the calculated acceleration signal. The method then chosen one between two classifiers for the upper body and the lower body according to the classification result, and used this to recognize seven actions.

However, the single accelerometer-based methods in [16]–[18] are ineffective when required to recognize a variety of complex actions. For this reason, some studies have used multiple accelerometers [19], [20]. In [19], five accelerometers were attached to the hip, wrist, arm, ankle, and thigh of the trainee. The acceleration signals were transformed using a fast Fourier transform (FFT), and several features, such as mean value, energy, and frequency-domain entropy, were calculated using the FFT coefficients to recognize 20 actions. In [20], 30 actions were recognized by using time- and frequency-domain measures of five accelerometers, which were attached to the hip, wrist, arm, ankle, and the thigh of the trainee.

Other studies have focused on combining accelerometers with other sensors. For example, in [21], an eWatch consisting of an accelerometer, a light sensor, a temperature sensor, and a microphone was introduced. Six eWatches were placed on the wrist, belt, necklace, trouser pocket, shirt pocket, and bag of the trainee. Classification accuracy on different combinations of sensor data was calculated. In the results, the best performance was achieved when all sensor data were used for action recognition. In [22], eight wearable sensors, each consisting of an accelerometer and a gyroscope, were used to recognize 12 actions. The sensors were placed on the upper-left arm, belt, the wrists, the knees, and the ankles of the trainee. Recognition performance was evaluated for different sets of activated sensors. The best performance was recorded when all sensors were activated.

Ease of use is the most important factor in designing a virtual simulator. From this perspective, sensor-based methods above have the following problem: it is inconvenient to charge and replace the batteries of sensors during training. In particular, if some batteries are worn out, training should be stopped to charge or replace them. In such a case, the user feels inconvenient in interaction with the simulator. For this reason, we do not consider sensor-based approaches.

B. VISION-BASED HUMAN ACTION RECOGNITION

Human action recognition has been extensively investigated in computer vision in the last few decades. For example, Thurau and Hlavác [23], Yang *et al.* [24], Maji *et al.* [25], Gupta *et al.* [26], Yao *et al.* [27], and Le *et al.* [28] explored human action recognition in still images. To recognize actions of in images, the human's pose was used as a feature in [23]–[25]. However, even if the two poses are identical, they can convey different meanings according to context. For example, if a human in an image is running to kick a ball, the methods in [23]–[25] may recognize it as "running" and not "kicking." For this reason, Gupta *et al.* [26], Yao *et al.* [27], and Le *et al.* [28] considered interactions between human and object. They extracted contextual cues to model the context. Moreover, based on the context, the action of the human was recognized in [26]–[28].

Human action recognition in video clips has also been studied. To extract discriminative and robust features from video data, several handcrafted features, such as volume local binary pattern (VLBP) [29], 3D scale-invariant feature transform (3D-SIFT) [30], 3D speeded up robust features (SURF3D) [31], the histogram of optical flow (HOF) [32], and the histogram of 3D-oriented gradients (HOG3D) [33], have been proposed. The use of such handcrafted features has yielded good classification accuracy for human action recognition in video clips [34]. This has led to a large body of research on the design of handcrafted features for human action recognition.

Instead of designing features by hand, some research groups have focused on a learning methodology to automatically extract features. In this approach, several learning algorithms, such as genetic programming (GP) and convolutional neural networks (CNNs), have been widely used. For example, in [35], a GP-based feature learning method was proposed where the color and the optical flow information extracted from RGB sequences were used as input data. Each candidate feature was randomly formulated as a tree structure. For a given input data, the candidate features were evolved through selection, crossover, and mutation at every iteration along the direction of maximizing recognition accuracy. When all iterations had been completed, the best feature was selected from among the candidate features. In [36], to obtain discriminative features from RGB and depth sequences, a restricted graph-based GP (RGGP) method was proposed. Each candidate feature was represented as a three-layer tree structure: input, filtering, and feature pooling layers. For the sake of variety, various candidate features were generated by randomly assembling the filtering and pooling operators in the filtering and pooling layers, respectively. A linear support vector machine (SVM) classifier was used to recognize human action. A fitness function was formulated as the classification error rate of the SVM classifier. Each candidate feature was updated using three processes (crossover, mutation, and selection) while the RGGP ran. Finally, the feature that most significantly minimized the fitness function was selected as the best for action recognition.

CNNs have been widely used to find features learned automatically from video data. For example, in [37], a 3D CNN model for action recognition was proposed. Since, in conventional 2D CNN models, 2D convolution operations are applied only to the spatial domain, only spatial features can be extracted through the corresponding models. However, in each convolutional layer of a 3D CNN model, a 3D convolution operation is employed to extract spatial and temporal features simultaneously from video data. In [38], to address the questions of where, what, and how action was performed in video sequences, a CNN-based deep action parsing (DAP3D-Net) structure was introduced. The DAP3D-Net consisted of six convolutional layers, four pooling layers, and two fully connected layers. A Euclidean square loss layer was also connected to the second fully connected layer to find a bounding box for a human agent. A softmax loss layer was connected to the second fully connected layer to predict the probabilities of actions occuring. Two cross-entropy loss layers were connected to the first and second fully connected layers to predict the action attributes. In [39], human action recognition in depth sequences was studied. To directly learn a feature from raw depth sequences, a 3D-based deep convolutional neural network ($3D^2$CNN) was proposed. For human action recognition, Liu *et al.* [39] used skeleton joint information as well as features learned from the $3D^2$CNN. Two features were entered into two SVM classifiers, and the outputs were fused to recognize actions.

C. SKELETON-BASED HUMAN ACTION RECOGNITION

Several recent studies on human action recognition have explored approaches that utilize information concerning the user's skeletal joints. Human pose estimation has also been actively researched to obtain more precise joint information from RGB, depth, and RGB-depth data [40]–[43]. The Kinect sensor, which can simultaneously capture real-time RGB, depth, and 3D skeletal joint information, has recently received considerable attention for a wide range of computer vision applications. In particular due to its convenience in acquiring 3D skeleton data with acceptable accuracy, a large amount of research has been conducted on skeleton-based human action recognition using the Kinect sensor. This research can be categorized into two groups, where one consists of methods that use a single Kinect sensor and the other of techniques that employ multiple Kinect sensors.

Examples of the first group include [44]–[47]. In order to recognize actions from 3D user skeleton data, several features, such as joint velocity, joint angle, and angular velocity, have been used in [44]–[46]. However, in the methods proposed there, since the Kinect sensor was in a fixed position, the user had to face it while standing. In this state, the action performed by the user was recognized. Therefore, the classification accuracy can significantly vary depending on the position of the Kinect sensor. In order to overcome this problem, in [47], histograms of 3D joint locations (HOJ3D) were proposed for view-invariant human action recognition. The HOJ3D were computed using 3D joint locations provided by the Kinect sensor, and were clustered into several posture visual words. A discrete hidden Markov model (HMM) was trained using the posture visual words and used to recognize the actions of the user.

In the second group of research on skeleton-based human action recognition using the Kinect sensor, multiple Kinect sensors were utilized to recognize human actions [48]–[50]. Azis *et al.* [48] proposed a two-view human action recognition system where skeleton data obtained from two Kinect sensors were fused based on their joint tracking status. The temporal change in the fused skeleton data according to actions was modeled as a sequence. The actions of the user were then classified through a sequence matching process. In [49], skeleton data obtained from two Kinect sensors were employed to enhance action recognition performance. Based on the distance between each Kinect sensor and the centroid, some reliable sensors were selected. The reliability of the skeleton data obtained from these Kinect sensors was then calculated. Based on the overall reliability of each skeleton data item, the fused data were obtained as the weighted average of the skeleton data. Following this, the approach in [48] was applied to recognize the actions of the user. In [50], a wireless Kinect sensor network system for a VR boxing game was proposed, where two Kinect sensors were used to detect the actions of the user. Instead of fusing the skeleton data obtained from the Kinect sensors, the system in [50] selected the results of the best sensor. To do so, the

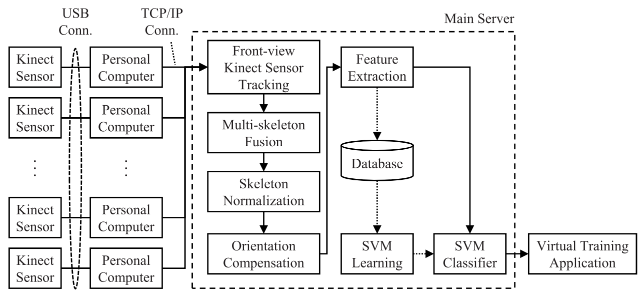


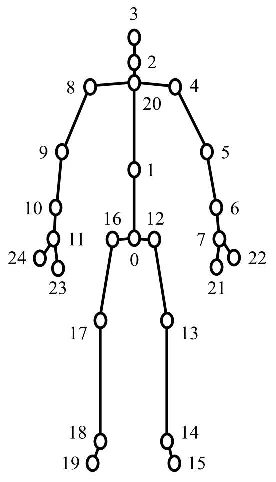
FIGURE 2. **Block diagram of the proposed simulator.**

effectiveness of each Kinect sensor was calculated based on the distance as well as orientation between it and the user. The motion information captured by the selected Kinect sensor was inputted into the VR boxing game system.

Our work belongs to skeleton-based human action recognition using multiple Kinect sensors. In our proposed training simulator, we take advantage of multiple Kinect sensors that can obtain precise user skeleton data regardless of his/her direction. In past work [48], [49], only two Kinect sensors were used to recognize the user's actions, even if the skeleton fusion methods used had been originally developed for two or more Kinect sensors. The Kinect sensors were placed in front of the user as well On the other hand, in our work, six Kinect sensor are used, and we observed that when the user did not face a Kinect sensor, the left and right sides of the skeleton provided by it were reversed. This observation is absent from [48], [49], and the skeleton fusion methods in [48] and [49] were developed without considering this issue. For this reason, it is undesirable to apply these methods to our simulator. Further, some issues are encountered in the implementation of the proposed simulator. The implementation is described in detail in the next section.

III. IMPLEMENTATION OF THE PROPOSED VIRTUAL TRAINING SIMULATOR

Fig. 2 shows a block diagram of the proposed virtual training simulator. It is composed of multiple Kinect sensors, personal computers, and a main server. Each Kinect sensor is physically connected to a personal computer via a universal serial bus (USB), and the trainee's skeleton data are transmitted to the personal computer via the USB. Moreover, data from each personal computer are transmitted to the main server through a transmission control protocol/internet protocol (TCP/IP) connection. In the main server, the Kinect sensors, which face the trainee, are detected. Once the Kinect sensors are found, skeleton data from them are united into one. To render the unified skeleton data invariant to such factors as body size and orientation of the trainee, skeleton normalization and orientation compensation algorithms are executed. Following these processes, features for human action recognition are extracted from skeleton data. Using these feature data, an SVM classifier generates the corresponding action classification result. A detailed description of each process is presented in the following subsections.



Spine Base	= 0
Spine Mid	= 1
Neck	= 2
Head	= 3
Left Shoulder	= 4
Left Elbow	= 5
Left Wrist	= 6
Left Hand	= 7
Right Shoulder	= 8
Right Elbow	= 9
Right Wrist	= 10
Right Hand	= 11
Left Hip	= 12
Left Knee	= 13
Left Ankle	= 14
Left Foot	= 15
Right Hip	= 16
Right Knee	= 17
Right Ankle	= 18
Right Foot	= 19
Spine Shoulder	= 20
Left Hand Tip	= 21
Left Thumb	= 22
Right Hand Tip	= 23
Right Thumb	= 24

FIGURE 3. **Skeleton joint information provided by the Kinect sensor.**

A. COORDINATE SYSTEM TRANSFORMATION

The Kinect sensor consists of an RGB camera, an infrared (IR) camera, and IR emitters. The RGB camera is used for color image acquisition, whereas the IR camera and the IR emitters extract depth information. The frame rate of the color and depth images is approximately 30 frames per second (fps). Developers can also acquire human skeleton data for 25 joints at 30 fps using the Kinect for Windows software development kit (SDK) offered by Microsoft. In this paper, human skeleton data consisted of the tracking state and the coordinate values of each joint.

Fig. 3 shows the skeletal joint information provided by the Kinect sensor. Human skeleton data is transmitted via the USB interface to its personal computer. The coordinate system of each Kinect sensor is transformed into the world coordinate system as follows:

$$\left[\hat{X}_i \ \hat{Y}_i \ \hat{Z}_i\right]^T = \mathbf{R}_i \left[X_i \ Y_i \ Z_i\right]^T + \mathbf{T}_i, \tag{1}$$

where $\hat{X}_i$, $\hat{Y}_i$, and $\hat{Z}_i$ are coordinates of the world coordinate system, and X_i, Y_i, and Z_i are coordinates of the coordinate system of the i^{th} Kinect sensor. The superscript $(\cdot)^T$ indicates the transpose operation. Parameter $\mathbf{R}_i$ is a 3×3-dimensional rotation matrix of the i^{th} Kinect sensor, and $\mathbf{T}_i$ is a 3×1-dimensional translation matrix of the i^{th} Kinect sensor. To obtain $\mathbf{R}_i$ and $\mathbf{T}_i$ at each Kinect sensor, a calibration procedure is initially performed using a calibration board and an IR camera. Conventionally, calibration is performed using an RGB camera and a calibration checkboard with a chessboard pattern [51]–[53]. However, the joint information provided by the Kinect sensor is represented in the IR camera's coordinate system. Moreover, the IR camera and the RGB camera are separate from each other in the Kinect sensor. Therefore, to obtain more accurate $\mathbf{R}_i$ and $\mathbf{T}_i$, we utilize an IR camera-based calibration procedure. Fig. 4 shows the calibration

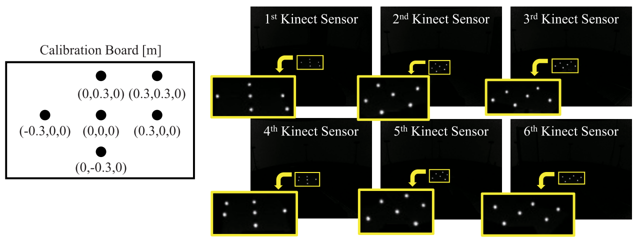


FIGURE 4. Calibration board and the infrared (IR) dots detected by each Kinect sensor.

board consisting of six IR markers and the IR dots detected by each Kinect sensor. Once the personal computer receives the IR image of the six IR dots from the corresponding Kinect sensor, each personal computer computes $\mathbf{R}_i$ and $\mathbf{T}_i$ using spatial relation information between 2D pixel coordinates of each IR dot and the 3D coordinates of the corresponding IR marker on the calibration board.

B. FRONT-VIEW KINECT SENSOR TRACKING

Once the coordinate system transformation is complete, each skeleton is aligned with respect to the origin of $(0, 0, 0)$ the IR marker on the calibration board placed at the center of the omni-directional treadmill. Fig. 5 shows the results of the visualization of the transformed skeletons when the six Kinect sensors, arranged in a ring, capture the trainee standing at the center of the ring. Although the coordinate system of each Kinect sensor is transformed into the world coordinate system, the transformed skeletons are not unified into one in this coordinate system, as shown in Fig. 5(a). Further, the Kinect sensor captures human skeleton data under the assumption that the user faces it. If not, the Kinect sensor (back-view Kinect sensor) may provide incorrect human skeleton data. As shown in Figs. 5(e)-(g), the left and right sides of the human skeleton provided by the back-view Kinect sensors are reversed. If such incorrect skeletons are entered into the SVM classifier, for example, the classification result for "throw high left" may be recognized as "throw high right." Therefore, to solve this problem, in our previous work [54], we proposed a front vector-tracing (FVT) method that ascertains front-view Kinect sensors at each frame. In the proposed simulator, the FVT method is employed to find front-view Kinect sensors.[1]

Fig. 6 shows the procedure of finding the front-view Kinect sensors through the FVT method. Let $\mathbf{v}_s^{(i)}$ be the vector from the left to the right shoulder joint of the human skeleton captured by the i^{th} Kinect sensor. Let $\mathbf{v}_p^{(i)}$ be the front-view vector of the skeleton captured by the i^{th} Kinect sensor. $\mathbf{v}_p^{(i)}$ is obtained by rotating $\mathbf{v}_s^{(i)}$ counterclockwise by 90°. The two vectors $\mathbf{v}_s^{(i)}$ and $\mathbf{v}_p^{(i)}$ are obtained per frame at each Kinect sensor.

Having chosen the front-view Kinect sensors, the fused skeleton can be obtained using skeleton data captured by them. However, since they are not identified at the outset, the fused skeleton needs to be initialized. In the proposed simulator, initial information pertaining to the skeleton is obtained by making the user face a dedicated Kinect sensor at the beginning, which is also used as the initial fused skeleton. Let $\mathbf{v}_f$ be the front-view vector of the fused skeleton. Suppose that the first Kinect sensor is the dedicated sensor. Then, $\mathbf{v}_f$ is initialized as $\mathbf{v}_p^{(1)}$. The two Kinect sensors, on either side of the dedicated Kinect sensor, are regarded as the front-view Kinect sensors.

In the next frame, the direction and orientation of the user can be changed according to his/her motion. Furthermore, the front-view Kinect sensors can be changed. Since the six sensors are uniformly aligned at 60° as shown in Fig. 1(b), three Kinect sensors are always selected as front-view sensors. Accordingly, the fused skeleton is sequentially obtained using the skeleton data captured by the three front-view Kinect sensors. In order to continue tracking the front-view Kinect sensors, the inner product of $\mathbf{v}_p^{(i)}$ and $\mathbf{v}_f$ is used to determine whether the i^{th} Kinect sensor is the front-view Kinect sensor. If the value of the inner product is positive, the i^{th} Kinect sensor becomes the front-view sensor as shown in Fig. 6(a). Conversely, if it is negative, it becomes the back-view Kinect sensor as shown in Fig. 6(b). This identification process is performed per frame for each Kinect sensor, and the front-view Kinect sensors are temporally tracjed. In the multi-skeleton fusion module, the fused skeleton is obtained by using skeleton data from the front-view Kinect sensors. The procedure of skeleton fusion is described in the next subsection.

C. MULTI-SKELETON FUSION

Let F be the set of indices of the Kinect sensors determined by the FVT method, f be the index of an element in $F, f \in F$, j be the joint index, $j \in \{0, \ldots, 24\}$, $s_{j,f}$ be the tracking state of the j^{th} joint of the f^{th} Kinect sensor, $d_{j,f}$ be the distance between the j^{th} joint of the f^{th} Kinect sensor and the f^{th} Kinect sensor, $\mathbf{P}_{j,f} = \left[x_{j,f},\ y_{j,f},\ z_{j,f}\right]$ be a vector representation of the position of the j^{th} joint of the f^{th} Kinect sensor, and $\bar{\mathbf{P}}_j = \left[\bar{x}_j,\ \bar{y}_j,\ \bar{z}_j\right]$ be a vector representation of the position of the j^{th} joint of the fused skeleton.

In the multi-skeleton fusion of the proposed simulator, $\bar{\mathbf{P}}_j$ is determined by solving the following optimization problem:

$$\min_{\bar{\mathbf{P}}_j} \sum_{f \in F} \lambda(s_{j,f}) \cdot \mu(d_{j,f}) \cdot \parallel \bar{\mathbf{P}}_j - \mathbf{P}_{j,f} \parallel \qquad (2)$$

where $\lambda(s_{j,f})$ and $\mu(d_{j,f})$ are weights that depend on $s_{j,f}$ and $d_{j,f}$, respectively.

[1]In color marker-based tracking methods [55]–[57], each marker of a different color is attached to different parts of the human body. The color markers are extracted from images and their patterns are used to compute the position and orientation of the camera. These methods can be used to find the front-view Kinect sensor. However, in our proposed simulator, the targets are projected onto the wall of the dome using the eight projectors mounted on the ceiling of the dome. To enable the trainee to experience a more realistic and immersive environment, the lights in the dome are turned off during training. The brightness of the indoor space is too low for the sensors to correctly detect markers. For this reason, we did not adopt color marker-based methods.

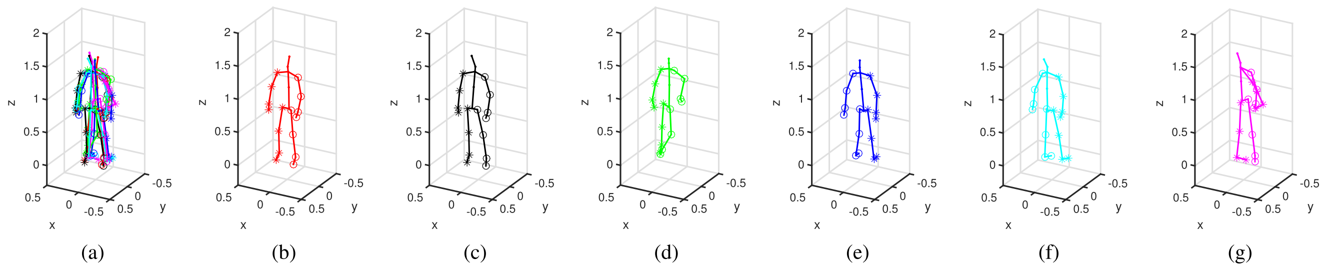
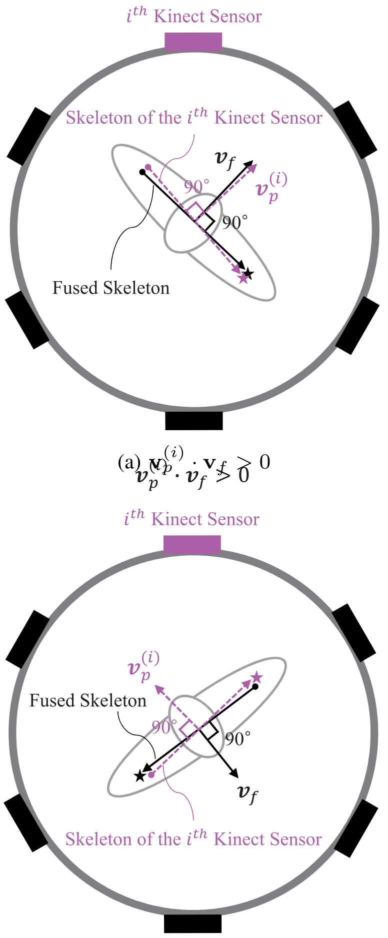


FIGURE 5. Visualization results of the transformed skeletons. The marker with the circle "o" represents the left side of the human skeleton. The marker with the asterisk "*" represents the right side of the human skeleton.



(a) $\mathbf{v}_p^{(i)} \cdot \mathbf{v}_f > 0$

$v_p^{(i)} \cdot v_f > 0$

(b) $\mathbf{v}_p^{(i)} \cdot \mathbf{v}_f < 0$

$v_p^{(i)} \cdot v_f < 0$

FIGURE 6. Procedure to find a front-view Kinect sensor through the front vector tracing (FVT) method. (a) The i^{th} Kinect sensor is a front-view Kinect sensor. (b) The i^{th} Kinect sensor is a back-view Kinect sensor. The marker "●" represents the Left Shoulder joint. The marker "★" represents the Right Shoulder joint.

The tracking state of each joint is categorized as *tracked*, *inferred*, or *not tracked*. If it is categorized as *tracked*, the joint is well tracked; if it is classified as *inferred*, the joint is occluded, and if classified as *not tracked*, the joint is completely invisible. Therefore, according to the tracking state, we define $\lambda(s_{j,f})$ as follows:

$$\lambda(s_{j,f}) = \begin{cases} 1.0 & \text{if } s_{j,f} \text{ is } \textit{tracked} \\ 0.5 & \text{if } s_{j,f} \text{ is } \textit{inferred} \\ 0.0 & \text{if } s_{j,f} \text{ is } \textit{not tracked} \end{cases} \tag{3}$$

In [58], it was shown that the accuracy of detecting the position of each joint depends on its distance from the Kinect sensor. Based on the measurement results in [58], we define $\mu(d_{j,f})$ as

$$\mu(d_{j,f}) = 1 - \left(\frac{0.4946\, e^{0.7\, d_{j,f}} - 1.1457}{5.7316 - 1.1457} \right) \tag{4}$$

where $0.4946 \;\; e^{0.7\, d_{j,f}}$ is the average error-fitting function whose function value specifies the average error in the skeleton joint according to distance $d_{j,f}$. Since an optimal sensing range of 1.2 to 3.5 m is recommended by Microsoft, $0.4946\, e^{0.7\, d_{j,f}}$ is normalized using 1.1457 (5.7316), which is the minimum (maximum) value of the error at 1.2 m (3.5 m). Then, using (3) and (4), the position of the j^{th} joint of the fused skeleton in (2) can be calculated as

$$\bar{\mathbf{P}}_j = \frac{\sum_{f \in F} \lambda(s_{j,f}) \cdot \mu(d_{j,f}) \cdot \mathbf{P}_{j,f}}{\sum_{f \in F} \lambda(s_{j,f}) \cdot \mu(d_{j,f})}. \tag{5}$$

D. SKELETON NORMALIZATION

The coordinates of the skeleton joints depend on the body sizes of the trainees. Variation in body size may cause misclassification of two motion sequences even if they represent the same motion. Therefore, the joint position data of the fused skeleton should be normalized to be invariant to the size of the body of the trainee. Thus, we use a kinematic tree consisting of 25 nodes and 24 edges. The nodes and edges represent the joints and limbs, respectively. Based on the kinematic tree, we define 24 limbs as described in Table 1 and designate the Spine Base joint ($j = 0$) as the root node of the kinematic tree. We also determine the length of each limb of the fused skeleton as the average of the limbs of 30 test subjects.

Let L_l be the length of the l^{th} limb, $l \in \{1, \ldots, 24\}$, and m_l and n_l be the indices of the starting and ending joints of the l^{th} limb, respectively, as shown Table 1. Let $\mathbf{N}_j, j = 0, \ldots, 24,$ be the position of the j^{th} joint of the normalized skeleton.

TABLE 1. Twenty-four limbs and their lengths.

Index (l)	Limb (Ending joint - Starting joint)	Length (meter)
1	Spine Mid (1) - Spine Base (0)	0.310
2	Left Hip (12) - Spine Base (0)	0.081
3	Right Hip (16) - Spine Base (0)	0.081
4	Spine Shoulder (20) - Spine Mid (1)	0.228
5	Neck (2) - Spine Shoulder (20)	0.075
6	Left Shoulder (4) - Spine Shoulder (20)	0.183
7	Right Shoulder (8) - Spine Shoulder (20)	0.183
8	Head (3) - Neck (2)	0.159
9	Left Elbow (5) - Left Shoulder (4)	0.250
10	Left Wrist (6) - Left Elbow (5)	0.236
11	Left Hand (7) - Left Wrist (6)	0.084
12	Left Hand Tip (21) - Left Hand (7)	0.086
13	Left Thumb (22) - Left Hand (7)	0.054
14	Right Elbow (9) - Right Shoulder (8)	0.250
15	Right Wrist (10) - Right Elbow (9)	0.236
16	Right Hand (11) - Right Wrist (10)	0.084
17	Right Hand Tip (23) - Right Hand (11)	0.086
18	Right Thumb (24) - Right Hand (11)	0.054
19	Left Knee (13) - Left Hip (12)	0.392
20	Left Ankle (14) - Left Knee (13)	0.380
21	Left Foot (15) - Left Ankle (14)	0.115
22	Right Knee (17) - Right Hip (16)	0.392
23	Right Ankle (18) - Right Knee (17)	0.380
24	Right Foot (19) - Right Ankle (18)	0.115

Algorithm 1 Skeleton Normalization

```
1: Input: $\bar{\mathbf{P}}_j, j = 0, \ldots, 24, L_l, m_l, n_l, l = 1, \ldots, 24$
2: Output: $\mathbf{N}_j, j = 0, \ldots, 24$
3: $\mathbf{N}_0 = \bar{\mathbf{P}}_0$
4: for $l = 1$ to 24 do
5:     $\mathbf{v}_l = \bar{\mathbf{P}}_{n_l} - \bar{\mathbf{P}}_{m_l}$
6:     $\bar{\mathbf{v}}_l = L_l \cdot \frac{\mathbf{v}_l}{\|\mathbf{v}_l\|}$
7:     $\mathbf{N}_{n_l} = \mathbf{N}_{m_l} + \bar{\mathbf{v}}_l$
8: end for
9: Terminate.
```

Based on the 24 limbs in Table 1, the joint coordinates of the fused skeleton are normalized using Algorithm 1. The normalization process starts from the root node (the Spine Base joint) and sequentially progresses in order of the limbs as listed in Table 1. Using the starting and ending positions of the joints of the limb, limb vector $\mathbf{v}_l$ is obtained. $\mathbf{v}_l$ is then normalized by the norm of its magnitude while its original direction is preserved. It is also scaled by a given length L_l in Table 1 as $\bar{\mathbf{v}}_l$. The end joint is updated using the starting joint and $\bar{\mathbf{v}}_l$. Having executed this procedure for all limbs, we can obtain the normalized skeleton. Fig. 7 shows an example of the result of skeleton normalization.

E. ORIENTATION COMPENSATION

The user's location in the world coordinate system changes dynamically according to his/her movement. To render skeleton data invariant to the absolute location of the user in the world coordinate system, we place the Spine Base joint at the origin of the world coordinate system as shown on the left side of Fig. 8. The positions of the other joints are also modified as

$$\mathbf{N}_j = \mathbf{N}_j - \mathbf{N}_0, \quad j = 1, 2, \ldots, 24. \qquad (6)$$

Algorithm 2 Orientation Compensation

```
1: Input: $\mathbf{N}_j = [x_j, y_j, z_j], j = 0, \ldots, 24$
2: Output: $\bar{\mathbf{N}}_j = [\bar{x}_j, \bar{y}_j, z_j], j = 0, \ldots, 24$
3: for $j = 1$ to 24 do
4:     $\mathbf{N}_j = \mathbf{N}_j - \mathbf{N}_0$
5: end for
6: Set $\bar{\mathbf{N}}_0$ to the origin (0, 0, 0).
7: Find the vector $\mathbf{v} = (v_x, v_y)$ from Right Hip joint to Left
   Hip joint.
8: $v_x = x_{12} - x_{16}$
9: $v_y = y_{12} - y_{16}$
10: Calculate rotation angle $\alpha$.
11: $\alpha = \arcsin\left(-v_x/\sqrt{v_x^2 + v_y^2}\right)$
12: for $j = 1$ to 24 do
13:     $\bar{x}_j = \cos(\alpha) \cdot x_j - \sin(\alpha) \cdot y_j$
14:     $\bar{y}_j = \sin(\alpha) \cdot x_j + \cos(\alpha) \cdot y_j$
15: end for
16: Terminate.
```

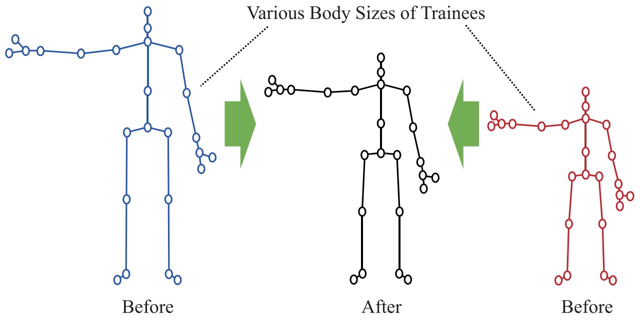


FIGURE 7. Example to illustrate the result of skeleton normalization.

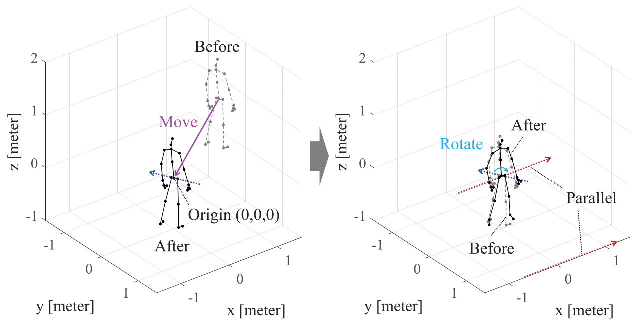


FIGURE 8. An example of orientation compensation.

Then, for view-invariant action recognition, the vector from the Right Hip joint ($j = 16$) to the Left Hip joint ($j = 12$) of the skeleton is set parallel to the x-axis of the world coordinate system as shown on the right side of Fig. 8. Algorithm 2 shows the pseudocode for orientation compensation, where $\bar{\mathbf{N}}_j$ is the position of the j^{th} joint of the skeleton after orientation compensation.

F. FEATURE EXTRACTION

Once the skeleton normalization and orientation compensation are complete, the skeleton becomes invariant to the user's body size, his/her absolute location, and viewpoint. By using this skeleton, feature extraction for human action recognition is carried out at each frame. The extracted features are as follows:

1) *3D Joint Position*: It has been reported that the 3D joint position information of the skeleton is useful for human action recognition [59]–[61]. In the proposed simulator, the 3D position information of 24 joints, excluding the Spine Base joint, is used for human action recognition. Since the Spine Base joint is placed at the origin through orientation compensation, its position is always $(0, 0, 0)$ regardless of the movement of the user. Therefore, we do not use its position information as a feature for human action recognition. The eventual feature dimensions of the *3D Joint Position* are $24 \times 3 = 72$.

2) *Unit Displacement*: The velocity of each joint has been used for human action recognition in many studies [62]–[64]. It is calculated from two consecutive records of joint positions. However, since the agility of each user is different, the joint velocities of the same motion can also vary from person to person. Moreover, since the speed of motion of the test person may be erratic, the joint velocities may not be stable features. Therefore, to achieve satisfactory performance, we define *Unit Displacement* by

$$\boldsymbol{D}_j[n] = \frac{\bar{\boldsymbol{N}}_j[n] - \bar{\boldsymbol{N}}_j[n-1]}{\|\bar{\boldsymbol{N}}_j[n], \bar{\boldsymbol{N}}_j[n-1]\|}, \quad j = 1, 2, \ldots, 24, \qquad (7)$$

where n is the frame index, and $\|\delta_1, \delta_2\|$ indicates the Euclidean distance between positions δ_1 and δ_2. Since $\bar{\boldsymbol{N}}_j[n] - \bar{\boldsymbol{N}}_j[n-1]$ is normalized by the Euclidean distance of $\|\bar{\boldsymbol{N}}_j[n], \bar{\boldsymbol{N}}_j[n-1]\|$, the *Unit Displacement* in (7) is against changes in the agility and speed of the user, and contains only the direction of movement of the joint. The the number of feature dimensions of *Unit Displacement* for 24 joints is $24 \times 3 = 72$.

3) *Joint Angle*: We define 17 *Joint Angles* as Tait-Bryan angles and use them as features for human action recognition. The components of these angles are as follows: (1) the yaw angle between the 1^{st} and 4^{th} limbs, (2) the roll and (3) yaw angles between the 6^{th} and 9^{th} limbs, (4) the roll and (5) yaw angles between the 7^{th} and 14^{th} limbs, (6) the roll and (7) yaw angles between the 9^{th} and 10^{th} limbs, (8) the roll and (9) yaw angles between the 14^{th} and 15^{th} limbs, (10) the roll and (11) yaw angles between the 2^{nd} and 19^{th} limbs, (12) the roll and (13) yaw angles between the 3^{rd} and 22^{th} limbs, (14) the roll and (15) yaw angles between the 19^{th} and 20^{th} limbs, and (16) the roll and (17) yaw angles between the 22^{th} and 23^{rd} limbs. Using the two connected limb vectors defined in Table 1, the *Joint Angles* are calculated at each frame. The number of feature dimensions of the *Joint Angle* is 17.

4) *Angular Velocity*: Let $k \in \{1, \ldots, 17\}$ be the angle index and $A_k[n]$ be the angle value of the k^{th} angle at frame n. Then, using two consecutive records of joint angles, the *Angular Velocity* of the k^{th} angle at frame n is calculated as

$$V_k[n] = \frac{A_k[n] - A_k[n-1]}{n - (n-1)}, \quad k = 1, 2, \ldots, 17. \qquad (8)$$

The number of feature dimensions of *Angular Velocity* is 17.

Although the above four features (*3D Joint Position*, *Unit Displacement*, *Joint Angle*, and *Angular Velocity*) can reflect momentary characteristics of action, this may be not sufficient for them to reflect the temporal characteristics of action. To capture these, we employ a simple moving average (SMA). For SMA calculation, each feature is temporarily stored in a buffer over B frames. Moreover, the SMA for each feature is calculated at each frame using B previously stored features as follows:

$$SMA[n] = \frac{\boldsymbol{f}[n] + \boldsymbol{f}[n-1] + \cdots + \boldsymbol{f}[n-(B-1)]}{B}, \qquad (9)$$

where $\boldsymbol{f}[n] = [\bar{\boldsymbol{N}}_1[n], \ldots, \bar{\boldsymbol{N}}_{24}[n], \boldsymbol{D}_1[n], \cdots, \boldsymbol{D}_{24}[n], A_1[n], \ldots, A_{17}[n], V_1[k], \ldots, V_{17}[k]]$ is a 1×178-dimensional concatenated feature vector at frame n. The size of the buffer B was set to 30 in the experiment.

The total number of dimensions of the features extracted at each frame is $(72 + 72 + 17 + 17) \times 2 = 356$. The extracted features are entered into the SVM classifier for training and classification.

G. SVM CLASSIFIER

To classify actions based on the momentary and temporal characteristics of action, an SVM is used as classifier in our proposed simulator. However, since it was originally designed for binary classification, it can be used only in two-class scenarios. To extend the usefulness of SVM, multi-class SVMs have been actively studied. Further, C-SVM, one of the most widely used multi-class SVMs, was used in the proposed simulator. The proposed simulator was implemented in C++ over the OpenCV library for C-SVM.

Since the SVM is a supervised learning method, manually annotated training data are required for it. The main idea underlying the SVM is to find an optimal hyperplane that separates the training data according to class labels. Since many training data items are not linearly separated, many non-linear classification methods have been studied to handle non-linearly separable training data. In non-linear classification methods, the kernel function is commonly used to map the input data of the SVM to high-dimensional feature spaces to find a hyperplane or a set of hyperplanes. In the proposed simulator, the radial basis function (RBF) was used as kernel function of the C-SVM.

Several parameters are needed for the C-SVM, including the RBF, the classification performance of which depends on the parameters. To train the C-SVM, we used OpenCV's `CvSVM::train_auto` function. This function allowed us to optimize the parameters and optimally train the C-SVM.

IV. EXPERIMENTAL RESULTS

To evaluate the performance of the proposed simulator, virtual military training was used as a representative example. To this end, the first-person shooter (FPS) game engine and contents developed by DoDaam Systems Ltd. were applied to

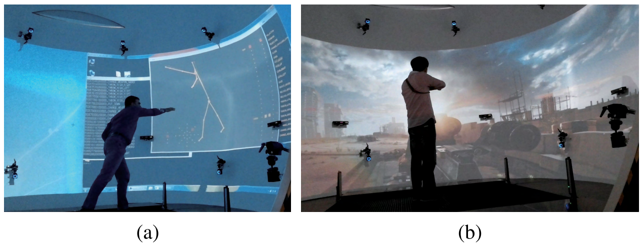


FIGURE 9. **Snapshot of the offline testing phase (a), and the practical online testing phase (b) using the proposed simulator.**

TABLE 2. **Experimental setup of hardware.**

	Main Server	Personal Computer
OS	Windows 7	Windows 8
Processor	Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40 GHz	Intel(R) Core(TM) i7-4790K CPU @ 4.00 GHz
Cores	8	4
Memory	16.0 GB	8.00 GB
GPU	Nvidia GTX 980 Ti	Intel(R) HD Graphics 4600

the proposed simulator. To ensure that the recognized action was synchronized with virtual reality contents, we performed several rigorous offline tests, as shown in Fig. 9(a). Through these tests, we ensured that the game character of the trainee was adequately controlled by using the action classification results of the SVM classifier. Moreover, the virtual game environment and virtual enemies, which were projected onto the wall of the dome using the projectors, enabled the trainee to experience more realistic and immersive virtual military training, as shown in Fig. 9(b). We used one main server and six personal computers. As shown in Fig. 2, each personal computer was physically connected to a Kinect sensors. The trainee's skeleton data obtained from each Kinect sensor were transmitted to the main server, where the proposed 360° multi-view human action recognition method was executed. A detailed description of the experimental setup of the hardware is presented in Table 2.

A. DATABASE AND ANNOTATION

To train and test the proposed simulator, the military training action datasets were captured in the studio. As shown in Fig. 1(b), six wall-mounted Kinect sensors, arranged at 60° intervals, captured the skeleton data of the subject. Each subject wore military gear and was armed with a pistol, a rifle, a grenade, and a sword, as shown in Fig. 10. In this state, each subject performed 25 military training actions: (1) *change weapon pistol*, (2) *change weapon rifle*, (3) *change weapon grenade*, (4) *change weapon sword*, (5) *throw high left*, (6) *throw high right*, (7) *throw low left*, (8) *throw low right*, (9) *reload pistol*, (10) *reload rifle*, (11) *shoot sword*, (12) *lean left*, (13) *lean right*, (14) *pick up*, (15) *put down*, (16) *open*, (17) *close*, (18) *change shoulder launch*, (19) *telescope*, (20) *gasmask*, (21) *jump*, (22) *bend*, (23) *crouch*, (24) *walk*, and (25) *run*. These training actions were performed by 30 subjects (20 serving soldiers and 10 civilians). Each subject repeated each action 20 times. The dataset

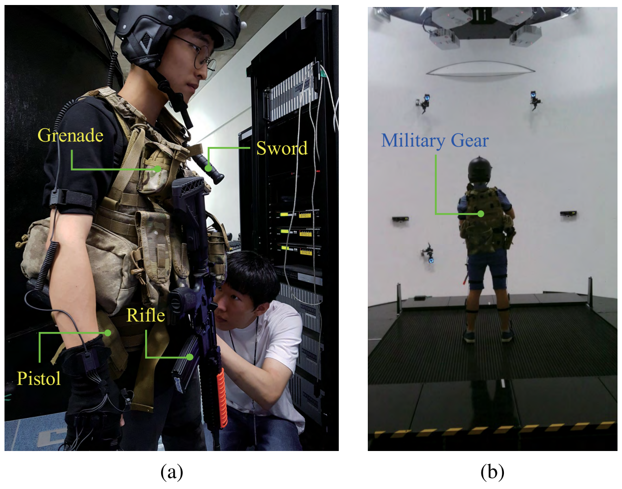


FIGURE 10. **Kinds of weapons (a) and military gear (b) used during the training and testing of the proposed simulator.**

provided a total of 15, 000 sequences. Fig. 11 shows some of the poses for the 25 military training actions from our dataset.

The action classification results are conventionally determined after watching an entire action sequence. In this case, the simulator's feedback lagged behind the trainee's action, and such latency might have caused significant degradation in interactivity between the two. In [65], the annotation paradigms were categorized into three types: sequence-level, frame-range, and action point annotation paradigms. In the sequence-level annotation paradigm, since the entire sequence was labeled with an action class label, high observational latency obtained. In the action point annotation paradigm, only the reference point (frame), which was determined based on a uniquely identified pose for action, was labeled with the action class label, and very low observational latency could be achieved. In the frame-range annotation paradigm, successive frames that centered on the reference frame were labeled with the action class label, and low observational latency was achieved.

We first used the action point annotation paradigm, but found that it was difficult to train the SVM classifier when the dataset labeled with this paradigm was used. Since the number of frames labeled with an action class label was too small, the training of the SVM classifier did not work well. As a result, the classification accuracy obtained was not acceptable. From this result, we realized that an adequate number of frames labeled with an action class label were needed to train the SVM classifier. Therefore, we employed the frame-range annotation paradigm. To classify action types in the early stages of action, we determined the action points on for the first 23 military training actions.[2] For action, we found a heuristically adequate length of frames centered on

[2]When the trainee walked (ran) during his/her training, the SVM classifier of the proposed simulator should have generated classification results as *walk* (*run*) sequentially between start and end of actions by considering game character control. For this reason, the sequence-level annotation paradigm was used in the cases of *walk* and *run*.

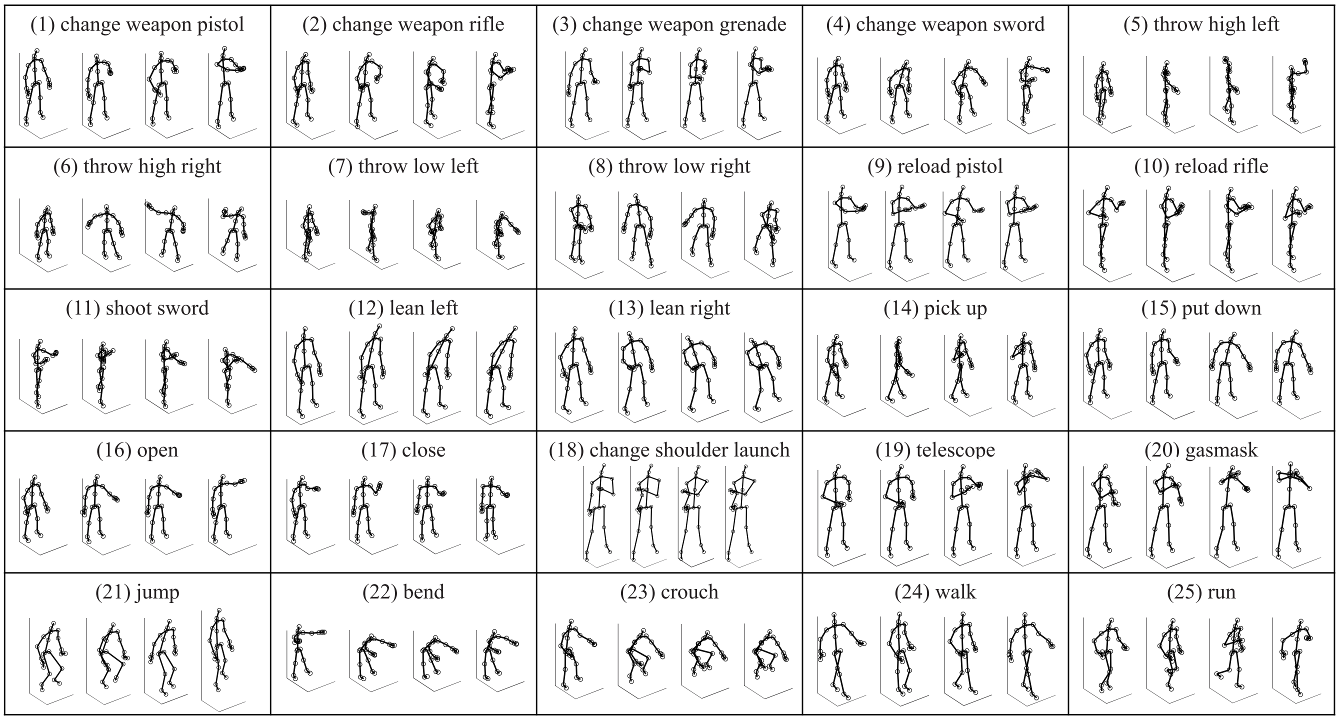


FIGURE 11. **Sample poses from 25 military training actions.**

the action point to train the SVM classifier. Based on the length of each military training action, frame-range annotation was carried out. The remaining frames were labeled with a *no action* class label (0).

B. PERFORMANCE EVALUATION

In order to evaluate the level of robustness of the proposed approach against skeleton errors, we first measured the error in the skeleton fused by the FVT method according to the number of the Kinect sensors. For accuracy of measurement, the OptiTrack motion capture system with 12 IR cameras was used to acquire ground truth data. To capture the entire body of the trainee, eight (four) IR cameras were mounted on the upper side (downside) of the wall of the dome in the ring. Moreover, 3D skeleton data obtained from the OptiTrack system was used as reference to assess the skeleton fused by the FVT method of the proposed simulator.

As shown in Fig. 12, the four poses (T-pose, crouch, rifle shooting, and pistol shooting) were captured by both OptiTrack and the proposed Kinect-based systems. The 3D skeleton model derived from the OptiTrack system was composed of a set of 20 joints. Moreover, the 3D skeleton models from the OptiTrack and the proposed Kinect sensor-based systems were slightly different from each other. For this reason, we selected one-on-one-matched joint pairs; (⟨Kinect⟩, ⟨OpitTrack⟩): (Spine Base, Hip), (Spine Mid, Spine 1), (Neck, Head), (Left Shoulder, Left Arm), (Left Elbow, Left Fore Arm), (Left Wrist, Left Hand), (Right Shoulder, Right Arm), (Right Elbow, Right Fore Arm), (Right Wrist, Right Hand), (Left Hip, Left Up Leg), (Left Knee, Left Leg), (Left Ankle, Left Foot), (Left Foot, Left Toe Base), (Right Hip, Right Up Leg), (Right Knee, Right Leg), (Right Ankle, Right Foot), (Right Foot, Right Toe Base), and (Spine Shoulder, Neck). We then calculated the average of the Euclidean distance with respect to (w.r.t.) each joint obtained by the OptiTrack system.

Table 3 shows the average error according to each pose, when the number of the Kinect sensors used for the FVT method was 1, 3, and 6, respectively. As shown in the table, the average error of the fused skeleton decreased with increase in the number of the Kinect sensors used. Further, the average error of the fused skeleton for "crouch" is greater than those of the fused skeleton for "T-pose," "rifle shooting," and "pistol shooting." This is because the pose of "crouch" contained a high level of self-occlusion, as shown in Fig. 12(b), compared with those of "T-pose," "rifle shooting," and "pistol shooting." On the other hand, the pose of "T-pose" contained a relatively low level of self-occlusion, and its average error was the least of the four poses.

In the proposed simulator, information concerning the front-view Kinect sensors was obtained by the FVT method at each frame. Further, in this state, the frame rate of the proposed simulator was measured approximately at 21 fps. If the FVT method is executed every W frames, the frame rate of the simulator can be improved. However, as a side effect, this can lead to a reduction in the accuracy of multi-skeleton fusion, which in turn leads to a reduction in the accuracy of action recognition. To verify this issue, we measured the

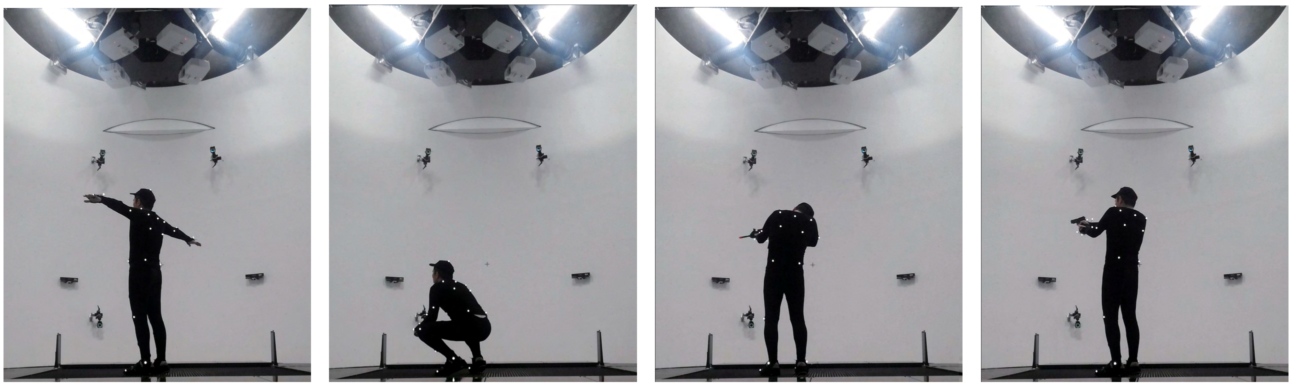


FIGURE 12. Snapshots from the four poses. Actions (from left to right): (a) T-pose, (b) crouch (c) rifle shooting, and (d) pistol shooting. Each pose was captured by the OptiTrack and the proposed Kinect sensor-based systems.

TABLE 3. Average error (in cm) per joint of the fused skeleton for four poses.

Joint	T-pose			crouch			rifle shooting			pistol shooting		
	$N=1$	$N=3$	$N=6$	$N=1$	$N=3$	$N=6$	$N=1$	$N=3$	$N=6$	$N=1$	$N=3$	$N=6$
Spine Base	5.78	4.94	4.87	8.95	8.69	5.99	6.23	6.02	5.94	6.08	5.48	5.43
Spine Mid	5.03	4.57	3.76	11.47	9.92	7.21	8.42	6.49	6.27	7.08	6.90	6.43
Neck	7.63	7.42	5.04	13.42	12.82	9.01	10.05	7.37	7.10	11.10	10.62	8.62
Left Shoulder	10.15	9.94	8.81	15.78	15.42	8.79	19.58	18.33	13.84	10.22	9.78	9.06
Left Elbow	10.63	9.61	8.59	19.69	17.22	8.99	17.93	16.00	12.62	12.14	11.76	11.33
Left Wrist	9.76	8.68	7.84	20.54	19.81	11.87	18.21	17.02	13.82	11.59	11.28	11.23
Right Shoulder	11.08	10.08	9.13	13.99	12.86	10.68	18.18	17.66	14.12	11.30	9.39	9.15
Right Elbow	12.00	9.56	8.49	15.06	13.89	9.86	17.37	16.43	12.09	12.22	11.15	10.32
Right Wrist	8.86	8.84	7.44	19.01	17.48	8.69	18.64	17.69	14.36	12.87	12.15	11.93
Left Hip	11.16	10.68	8.28	18.48	18.19	10.02	13.38	12.03	9.97	14.51	14.01	13.40
Left Knee	11.34	10.30	8.46	17.55	16.25	14.55	14.16	13.68	11.18	11.33	10.39	10.10
Left Ankle	9.06	8.93	6.84	19.96	17.26	12.74	14.56	13.40	11.72	7.34	6.54	4.97
Left Foot	10.86	10.68	8.68	20.79	20.47	15.40	11.81	11.26	9.47	6.83	6.65	4.75
Right Hip	10.62	10.28	8.15	20.25	18.65	12.19	12.84	11.75	10.11	14.67	13.12	12.24
Right Knee	11.75	10.27	8.36	16.14	15.54	15.19	14.31	13.63	10.96	11.19	10.89	10.55
Right Ankle	10.65	9.68	7.13	19.84	17.69	13.95	13.40	12.78	11.64	7.64	7.04	5.27
Right Foot	10.87	10.67	9.07	21.80	20.68	16.82	12.32	11.76	9.77	7.89	6.39	5.81
Spine Shoulder	6.89	6.45	5.89	11.13	10.46	8.26	11.13	10.46	7.26	8.48	7.85	6.87
Average	**9.67**	**8.98**	**7.49**	**16.88**	**15.74**	**11.12**	**14.03**	**12.99**	**10.68**	**10.25**	**9.52**	**8.74**

N: # of Kinect sensors used in the FVT method. In case of $N = 3$, three Kinect sensors arranged at 120° intervals were selected.

performance of the proposed simulator in terms of frame rate (fps), the average error of the fused skeleton, and the average frame-by-frame classification accuracy, defined as the sum of all correctly classified frames divided by the total number of frames in the test sequence. This evaluation was performed on our military training action database. For the cross-subject test, data for half the subjects (10 serving soldiers and five civilians) were used for training and the remaining half for testing.

Table 4 lists the performance of the proposed simulator when W was set to 1, 10, and 20, respectively. From the table, it can be seen that the frame rate of the proposed simulator could be improved when W was set to a relatively large value. However, the average error of the fused skeleton increased with the value of W. The reason for this was as follows. Before executing the FVT method, the front-view Kinect sensors could be changed according to the motion of the trainee. In such a case, it was difficult to guarantee acceptable accuracy of data relating to the fused skeleton. If incorrect skeleton data were used to train the SVM classifier, it generated incorrect classification results, as shown in Table 4. Through offline testing and practical online testing, we verified that a frame rate of 21 fps ($W = 1$) was sufficient to provide an interactive and immersive virtual training environment. Moreover, in order to synchronize the action of the game character with that of the trainee, it was important to

TABLE 4. Comparison of performance of the proposed simulator, when the FVT method performed every W frames.

	$W=1$	$W=10$	$W=20$
Frame rate (fps)	21	22.3	23.1
Error of the fused skeleton (cm)	8.31	10.59	15.78
Action-by-action classificatio accuracy (%)	93.56	90.17	84.43

Action Type	ID	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
no action	0	99	.1	.1	.1						.2					.1		.1	.1			.1		.1			
change weapon pistol	1	14	86																								
change weapon rifle	2	12		88																							
change weapon grenade	3	13			87																						
change weapon sword	4	14				86																					
throw high left	5	11					89																				
throw high right	6	10						90																			
throw low left	7	13							87																		
throw low right	8	15								85																	
reload pistol	9	3									97																
reload rifle	10	4										96															
shoot sword	11	11											89														
lean left	12	10												90													
lean right	13	10													90												
pick up	14	4														95	1										
put down	15	3															97										
open	16	13																83	4								
close	17	15																	85								
change shoulder launch	18	15																		85							
telescope	19	14																			86						
gasmask	20	6																				94					
jump	21	9																					91				
bend	22	10																						90			
crouch	23	7																							93		
walk	24	11																								89	
run	25	8																									92

FIGURE 13. Confusion matrix based on the results of the average frame-by-frame classification accuracy.

obtain precise skeleton data. For this reason, in the proposed simulator, we set the value of W to 1.

Fig. 13 shows the confusion matrix based on the results of the accuracy of frame-by-frame classification. The diagonal elements in the confusion matrix indicate the correctly classified rates for actions. The average frame-by-frame classification accuracy was approximately 93.56%. We observed that misclassification, where the *no action* class label was misclassified as another action class label, occured only in cases of *change weapon pistol*, *change weapon rifle*, *change weapon grenade*, *reload pistol*, *pick up*, *open*, *close*, *gasmask*, and *bend*. From the figure, we see that the misclassification rates of the 25 actions were concentrated in the *no action* class, and not in the other classes. Because of the well-designed features and database, few cases obtained where an action was classified as another in the experiment. This helped in the development of the post-processing module, where the action classification result sequence was converted into one-key input per action.

In conventional FPS games, the game character is controlled by input from a keyboard or a mouse. In our virtual training simulator, the game character was controlled by the action classification result. However, the SVM classifier of the proposed simulator generated the action classification result at each frame. The resulting sequence contained the same action class labels even if the given action was performed only once. If the result sequence was entered directly into the FPG game engine, the game character repeated the action several times. Therefore, one-key input per action was needed to reconcile the trainee's actions with those of the game character. To fulfill this goal, we developed a simple post-processing module that was inserted at the back of the SVM classifier module. The action classification result of the SVM classifier was entered into the post-processing module and stored in a buffer. The post-processing module found the action class label with the largest portion of the buffer. The action class label found was entered as a one-key input into the FPS game engine. As described in Sect. IV-A, the action class labels were labeled in the early stage of the given actions. Moreover, there was an interval between actions. Therefore, the case where the buffer was filled with a *no action* class obtained. Once one-key input generation was triggered, it was not triggered again until the buffer was filled with a *no action* class. Therefore, there was no more than one event firing for a single action.

If one-key input was triggered within the ground truth annotation region, it was considered a correctly classified event. If not, it was not counted. Action-by-action classification accuracy was then defined as the number of instance of correct classification divided by the total number of actions performed in the test sequences. Further, observation latency was defined as the difference between the frame where an action class label except *no action* started and that where one-key input generation was triggered. Table 5 shows the average action-by-action classification accuracy and the observational latency of the proposed simulator. We observed

TABLE 5. Average action-by-action classification accuracy and observational latency.

Action Type	Action-by-action classification accuracy (%)	Observational latency (frame)
change weapon pistol	88.7	11.1
change weapon rifle	87.8	10.8
change weapon grenade	87.9	8.3
change weapon sword	100.0	12.7
throw high left	100.0	8.8
throw high right	100.0	9.2
throw low left	100.0	7.7
throw low right	100.0	7.9
reload pistol	93.8	9.0
reload rifle	100.0	9.7
shoot sword	100.0	8.2
lean left	100.0	7.9
lean right	100.0	7.8
pick up	93.3	5.2
put down	100.0	7.9
open	88.7	6.4
close	93.3	7.5
change shoulder launch	100.0	6.9
telescope	100.0	8.9
gasmask	93.3	6.1
jump	100	7.3
bend	93.3	8.3
crouch	100	6.4
Average	**96.5**	**8.3**

that for cases of *change weapon sword*, *throw high left*, *throw high right*, *throw low left*, *throw low right*, *reload rifle*, *shoot sword*, *lean left*, *lean right*, *put down*, *change shoulder launch*, *telescope*, *jump*, and *crouch*, the average action-by-action classification accuracies were as high as 100%. This shows that the game character's actions could be perfectly controlled according to the trainee's actions.

For the other actions, performance can be explained based on the average frame-by-frame classification accuracy shown in Fig. 13. Misclassification where *no action* class label was misclassified as another action class label obtained only in these cases. We should point out that one-key input generation was triggered outside the ground truth annotation region. In the experiment, most incorrect triggers occurred at the front of the ground truth annotation region. Table 5 shows that the average observational latency was 8.4 frames for the 25 military training actions. When considering the frame rate of the proposed system at approximately 21 fps, the average latency could be converted to approximately 0.395 s. Although there were short lags between actions of the trainee and the game character, the test subjects did not experience significant inconvenience.

V. CONCLUSION

There has been a considerable amount of research interest in virtual training simulators. To provide a trainee with an interactive training environment, past research on virtual training simulators have focused on obtaining precise human action information using a wearable motion capture suit-based human action recognition approach, with less emphasis on user convenience. In this paper, we introduced a virtual training simulator based on 360° multi-view human action recognition. In our simulator, human skeleton data were captured from multiple Kinect sensors without the aid of wearable devices. Due to practical issues in the implementation of the proposed simulator, our implementation of the virtual training simulator using multiple Kinect sensors was primarily addressed in this paper. For performance evaluation, virtual military training was used. The results highlighted the effectiveness of the proposed simulator in terms of frame-by-frame classification accuracy, action-by-action classification accuracy, and observational latency.

REFERENCES

[1] D. Lee, K. Baek, J. Lee, and H. Lim, "A development of virtual reality game utilizing Kinect, oculus rift and smartphone," *Int. J. Appl. Eng. Res.*, vol. 11, no. 2, pp. 829–833, 2016.

[2] Z. Merchant, E. T. Goetz, L. Cifuentes, W. Keeney-Kennicutt, and T. J. Davis, "Effectiveness of virtual reality-based instruction on students' learning outcomes in K-12 and higher education: A meta-analysis," *Comput. Edu.*, vol. 70, pp. 29–40, Jan. 2014.

[3] D. Meldrum, A. Glennon, S. Herdman, D. Murray, and R. McConn-Walsh, "Virtual reality rehabilitation of balance: Assessment of the usability of the Nintendo Wii Fit Plus," *Disability Rehabil. Assistive Technol.*, vol. 7, no. 3, pp. 205–210, 2012.

[4] K. E. Kopecky and E. H. Winer, "MetaTracker: Unifying and abstracting 3-D motion tracking data from multiple heterogenous hardware systems," *IEEE Access*, vol. 4, pp. 189–203, 2016.

[5] H. Oh and S. Lee, "Visual presence: Viewing geometry visual information of UHD S3D entertainment," *IEEE Trans. Image Process.*, vol. 25, no. 7, pp. 3358–3371, Jul. 2016.

[6] H. Kim, S. Lee, and A. C. Bovik, "Saliency prediction on stereoscopic videos," *IEEE Trans. Image Process.*, vol. 23, no. 4, pp. 1476–1490, Apr. 2014.

[7] B. W. Knerr, "Immersive simulation training for the dismounted soldier," U.S. Army Res. Inst. Behavioral Social Sci., Tech. Rep., Feb. 2007.

[8] G. S. Taylor and J. S. Barnett, "Evaluation of wearable simulation interface for military training," *Human Factors, J. Human Factors Ergonom. Soc.*, vol. 55, no. 3, pp. 672–690, Jun. 2013.

[9] *Quantum 3D Product Literature*, Quantum3D Incorporated, San Jose, CA, USA. [Online]. Available: http://www.quantum3d.com

[10] J. Han, L. Shao, D. Xu, and J. Shotton, "Enhanced computer vision with microsoft Kinect sensor: A review," *IEEE Trans. Cybern.*, vol. 43, no. 5, pp. 1318–1334, Oct. 2013.

[11] Z.-C. Wang, C.-C. Tsai, and M.-C. Chien, "Design of an intelligent soldier combat training system," *Int. J. Autom. Smart Technol.*, vol. 2, no. 4, pp. 309–317, 2012.

[12] A. Stork *et al.*, "Enabling virtual assembly training in and beyond the automotive industry," in *Proc. IEEE 18th Int. Conf. Virtual Syst. Multimedia (VSMM)*, Sep. 2012, pp. 347–352.

[13] C. Park and J. Moon, "Using game technology to develop snowboard training simulator," in *Proc. Int. Conf. Human-Comput. Interact.*, Jul. 2013, pp. 723–726.

[14] F. Cassola, L. Morgado, F. de Carvalho, H. Paredes, B. Fonseca, and P. Martins, "Online-Gym: A 3D virtual gymnasium using Kinect interaction," *Procedia Technol.*, vol. 13, pp. 130–138, 2014.

[15] D. Bogatinov, P. Lameski, V. Trajkovik, and K. M. Trendova, "Firearms training simulator based on low cost motion tracking sensor," *Multimedia Tools Appl.*, vol. 76, no. 1, pp. 1403–1418, 2017.

[16] Z. He, Z. Liu, L. Jin, L.-X. Zhen, and J.-C. Huang, "Weightlessness feature—A novel feature for single tri-axial accelerometer based activity recognition," in *Proc. IEEE 19th Int. Conf. Pattern Recognit. (ICPR)*, Dec. 2008, pp. 1–4.

[17] A. M. Khan, Y.-K. Lee, S. Y. Lee, and T.-S. Kim, "A triaxial accelerometer-based physical-activity recognition via augmented-signal features and a hierarchical recognizer," *IEEE Trans. Inf. Technol. Biomed.*, vol. 14, no. 5, pp. 1166–1172, Sep. 2010.

[18] A. M. Khan, Y.-K. Lee, S. Lee, and T.-S. Kim, "Accelerometer's position independent physical activity recognition system for long-term activity monitoring in the elderly," *Med. Biol. Eng. Comput.*, vol. 48, no. 12, pp. 1271–1279, Dec. 2010.

[19] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," in *Proc. Int. Conf. Pervasive Comput.*, Apr. 2004, pp. 1–17.

[20] E. M. Tapia *et al.*, "Real-time recognition of physical activities and their intensities using wireless accelerometers and a heart rate monitor," in *Proc. 11th IEEE Int. Symp. Wearable Comput.*, Boston, MA, USA, Oct. 2007, pp. 37–40.

[21] U. Maurer, A. Smailagic, D. P. Siewiorek, and M. Deisher, "Activity recognition and monitoring using multiple sensors on different body positions," in *Proc. IEEE Int. Workshop Wearable Implant. Body Sensor Netw. (BSN)*, Apr. 2006, pp. 1–4.

[22] A. Y. Yang, S. Iyengar, P. Kuryloski, and R. Jafari, "Distributed segmentation and classification of human actions using a wearable motion sensor network," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2008, pp. 1–8.

[23] C. Thurau and V. Hlaváč, "Pose primitive based human action recognition in videos or still images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2008, pp. 1–8.

[24] W. Yang, Y. Wang, and G. Mori, "Recognizing human actions from still images with latent poses," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 2030–2037.

[25] S. Maji, L. Bourdev, and J. Malik, "Action recognition from a distributed representation of pose and appearance," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2011, pp. 3177–3184.

[26] A. Gupta, A. Kembhavi, and L. S. Davis, "Observing human-object interactions: Using spatial and functional compatibility for recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 10, pp. 1775–1789, Oct. 2009.

[27] B. Yao, X. Jiang, A. Khosla, A. L. Lin, L. Guibas, and L. Fei-Fei, "Human action recognition by learning bases of action attributes and parts," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Nov. 2011, pp. 1331–1338.

[28] D. T. Le, R. Bernardi, and J. Uijlings, "Exploiting language models to recognize unseen actions," in *Proc. 3rd ACM Conf. Int. Conf. Multimedia Retr.*, Apr. 2013, pp. 231–238.

[29] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, Jul. 2002.

[30] P. Scovanner, S. Ali, and M. Shah, "A 3-dimensional sift descriptor and its application to action recognition," in *Proc. ACM Multimedia*, Sep. 2007, pp. 357–360.

[31] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Comput. Vis. Image Understand.*, vol. 110, no. 3, pp. 346–359, 2008.

[32] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Anchorage, AK, USA, Jun. 2008, pp. 1–8.

[33] A. Klaser, M. Marszalek, and C. Schmid, "A spatio-temporal descriptor based on 3D-gradients," in *Proc. 19th Brit. Mach. Vis. Conf. (BMVS)*, Sep. 2008, pp. 275–284.

[34] H. Wang, D. Oneata, J. Verbeek, and C. Schmid, "A robust and efficient video representation for action recognition," *Int. J. Comput. Vis.*, vol. 119, no. 3, pp. 219–238, Sep. 2016.

[35] L. Liu, L. Shao, X. Li, and K. Lu, "Learning spatio-temporal representations for action recognition: A genetic programming approach," *IEEE Trans. Cybern.*, vol. 46, no. 1, pp. 158–170, Jan. 2016.

[36] L. Liu and L. Shao, "Learning discriminative representations from RGB-D video data," in *Proc. 23th Int. Joint Conf. Artif. Intell. (IJCAI)*, vol. 4. Aug. 2013, pp. 1493–1500.

[37] S. Ji, W. Xu, M. Yang, and K. Yu, "3D convolutional neural networks for human action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 221–231, Jan. 2013.

[38] L. Liu, Y. Zhou, and L. Shao. (2016). "DAP3D-Net: Where, what and how actions occur in videos?" [Online]. Available: https://arxiv.org/abs/1602.03346

[39] Z. Liu, C. Zhang, and Y. Tian, "3d-based deep convolutional neural network for action recognition with depth sequences," *Image Vis. Comput.*, vol. 55, pp. 93–100, Nov. 2016.

[40] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik, "Human pose estimation with iterative error feedback," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 4733–4742.

[41] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Oct. 2016, pp. 483–499.

[42] X. Zhou *et al.*, "Sparseness meets deepness: 3D human pose estimation from monocular video," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 4966–4975.

[43] F. Zhou and F. De la Torre, "Spatio-temporal matching for human pose estimation in video," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 8, pp. 1492–1504, Aug. 2016.

[44] V. Bloom, D. Makris, and V. Argyriou, "G3D: A gaming action dataset and real time action recognition evaluation framework," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops*, Providence, RI, USA, Jun. 2012, pp. 7–12.

[45] G. T. Papadopoulos, A. Axenopoulos, and P. Daras, "Real-time skeleton-tracking-based human action recognition using Kinect data," in *Proc. Int. Conf. Multimedia Modeling*, Jan. 2014, pp. 473–483.

[46] V. Bloom, V. Argyriou, and D. Makris, "Hierarchical transfer learning for online recognition of compound actions," *Comput. Vis. Image Understand.*, vol. 144, pp. 62–72, Mar. 2016.

[47] L. Xia, C.-C. Chen, and J. K. Aggarwal, "View invariant human action recognition using histograms of 3D joints," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2012, pp. 20–27.

[48] N. A. Azis, H.-J. Choi, and Y. Iraqi, "Substitutive skeleton fusion for human action recognition," in *Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, Feb. 2015, pp. 170–177.

[49] N. A. Azis, Y. S. Jeong, H. J. Choi, and Y. Iraqi, "Weighted averaging fusion for multi-view skeletal data and its application in action recognition," *IET Comput. Vis.*, vol. 10, no. 2, pp. 134–142, 2016.

[50] M. Li, W. Song, L. Song, K. Huang, Y. Xi, and K. Cho, "A wireless Kinect sensor network system for virtual reality applications," in *Proc. Int. Conf. Comput. Sci. Appl.*, Dec. 2016, pp. 61–65.

[51] Z. Zhang, "Microsoft Kinect sensor and its effect," *IEEE Multimedia*, vol. 19, no. 2, pp. 4–10, Feb. 2012.

[52] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot Syst. (IROS)*, Vilamoura, Portugal, Oct. 2012, pp. 573–580.

[53] C. Raposo, J. P. Barreto, and U. Nunes, "Fast and accurate calibration of a Kinect sensor," in *Proc. IEEE Int. Conf. 3D Vision-3DV*, Jun. 2013, pp. 342–349.

[54] J. Kim, I. Lee, J. Kim, and S. Lee, "Implementation of an omnidirectional human motion capture system using multiple Kinect sensors," *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, vol. E98-A, no. 9, pp. 2004–2008, 2015.

[55] S. Yonemoto, A. Matsumoto, D. Arita, and R.-I. Taniguchi, "A real-time motion capture system with multiple camera fusion," in *Proc. IEEE Int. Conf. Image Anal. Process.*, Sep. 1999, pp. 600–605.

[56] R. Y. Wang, "Practical color-based motion capture," Massachusetts Inst. Technol., Cambridge, MA, USA, Tech. Rep., 2011.

[57] J. Liu, S. Chen, H. Sun, Y. Qin, and X. Wang, "Real time tracking method by using color markers," in *Proc. IEEE Int. Conf. Virtual Reality Vis. (ICVRV)*, Sep. 2013, pp. 106–111.

[58] M. A. Livingston, J. Sebastian, Z. Ai, and J. W. Decker, "Performance measurements for the Microsoft Kinect skeleton," in *Proc. IEEE Virtual Reality Workshops*, Mar. 2012, pp. 119–120.

[59] J. Gu, X. Ding, S. Wang, and Y. Wu, "Action and gait recognition from recovered 3-D human joints," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 40, no. 4, pp. 1021–1033, Aug. 2010.

[60] J. Wang, Z. Liu, Y. Wu, and J. Yuan, "Mining actionlet ensemble for action recognition with depth cameras," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2012, pp. 1290–1297.

[61] W. Zhu *et al.* (2016). "Co-occurrence feature learning for skeleton based action recognition using regularized deep LSTM networks." [Online]. Available: https://arxiv.org/abs/1603.07772

[62] A. Ramey, V. González-Pacheco, and M. A. Salichs, "Integration of a low-cost RGB-D sensor in a social robot for gesture recognition," in *Proc. ACM/IEEE Int. Conf. Human-Robot Interact.*, 2011, pp. 229–230.

[63] M. Zanfir, M. Leordeanu, and C. Sminchisescu, "The moving pose: An efficient 3D kinematics descriptor for low-latency action recognition and detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 2752–2759.

[64] W. Ding, K. Liu, F. Cheng, and J. Zhang, "STFC: Spatio-temporal feature chain for skeleton-based human action recognition," *J. Vis. Commun. Image Represent.*, vol. 26, pp. 329–337, Jan. 2015.

[65] S. Nowozin and J. Shotton, "Action points: A representation for low-latency online human action recognition," Microsoft Res., Cambridge, U.K., Tech. Rep. MSR-TR-2012-68, 2012.

BEOM KWON was born in South Korea in 1989. He received the B.S. degree in electrical and electronic engineering from Soongsil University, Seoul, South Korea, in 2012. He is currently pursuing the M.S. and Ph.D. degrees with the Multidimensional Insight Laboratory, Yonsei University. His research interests are in the area of wireless communication networks, computer vision, and machine learning.

JUNGHWAN KIM was born in South Korea, in 1990. He received the B.S. degree in electrical and electronic engineering from Yonsei University, Seoul, South Korea, in 2013. He is currently with Jinhak Inc., Seoul, where he is involved in natural language processing. His research interests are in the area of computer vision and natural language processing.

KYOUNGOH LEE was born in South Korea in 1990. He received the B.S. degree in electronic engineering from Soongsil University, Seoul, South Korea, in 2014. He is currently pursuing the M.S. and Ph.D. degrees with the Multidimensional Insight Laboratory, Yonsei University. His research interests are in the area of software defined network, computer vision, and machine learning.

YANG KOO LEE received the B.S. degree in computer and information engineering from Cheongju University, Cheongju, South Korea, in 2002, and the M.S. and Ph.D. degrees in computer science from Chungbuk National University, Cheongju, in 2004 and 2010, respectively. He was a Research Student with the University of Aizu, Fukushima, Japan, in 2009. From 2010 to 2011, he was a Post-Doctoral Fellow with Chungbuk National University. Since 2011, he has been with the Electronics and Telecommunications Research Institute, Daejeon, South Korea, where he is currently a Senior Researcher. His main research interests include location-based services, sensor networks, data mining, and human–computer interaction technology for virtual reality application.

SANGJOON PARK received the B.S. and M.S. degrees in electronics engineering from Kyung-Pook National University in 1988 and 1990, respectively, and the Ph.D. degree from the Computer Science Department, North Carolina State University, in 2006. He was a Senior Researcher with Agency for Defense Development from 1990 to 2001. He has been an Adjunct Professor with the University of Science and Technology since 2011. He is currently a Project Leader with the Electronics and Telecommunications Research Institute. His current research interests are in indoor localization, human action recognition, augmented reality, and virtual reality system.

SANGHOON LEE (M'05–SM'12) received the B.S. degree in electrical engineering from Yonsei University in 1989, the M.S. degree in electrical engineering from the Korea Advanced Institute of Science and Technology in 1991, and the Ph.D. degree in electrical engineering from The University of Texas at Austin in 2000. From 1991 to 1996, he was with Korea Telecom. From 1999 to 2002, he was with Lucent Technologies, where he was involved in 3G wireless and multimedia networks. In 2003, he joined the faculty of the Department of Electrical and Electronics Engineering, Yonsei University, Seoul, South Korea, where he is a Full Professor. His current research interests include image/video quality assessment, computer vision, graphics, cloud computing, and multimedia communications and wireless networks. He currently serves as a member of the Technical Committee of the IEEE Multimedia Signal Processing (2016–) and the IEEE IVMSP Technical Committee (2014–) and the APSIPA IVM TC Vice Chair (2016–). Prior to that, he was the Technical Program Co-Chair at the International Conference on Information Networking 2014, and at the Global 3-D Forum 2012 and 2013, and was the General Chair of the 2013 IEEE IVMSP Workshop. He received the 2015 Yonsei Academic Award from Yonsei University, the 2012 Special Service Award from the IEEE Broadcast Technology Society, and the 2013 Special Service Award from the IEEE Signal Processing Society. He was an Associate Editor of the IEEE TRANSACTIONS ON IMAGE PROCESSING (2010–2014). He has been an Associate Editor of the IEEE SIGNAL PROCESSING LETTERS (2014–) and the *Journal of Electronic Imaging* (2015–) and the Chair of the IEEE P3333.1 Quality Assessment Working Group (2011–). He also served as a special issue Guest Editor of the IEEE TRANSACTIONS ON IMAGE PROCESSING in 2013, and as an Editor of the *Journal of Communications and Networks* (2009–2015).

• • •