Sensors and Materials, Vol. 34, No. 5 (2022) 1971–1990
MYU Tokyo

S & M 2947

Home Fitness and Rehabilitation Support System Implemented by Combining Deep Images and Machine Learning Using Unity Game Engine

Neng-Sheng Pai, Pin-Xiang Chen, Pi-Yun Chen,* and Zi-Wen Wang

Department of Electrical Engineering, National Chin-Yi University of Technology,
Taichung 41170, Taiwan (ROC)

(Received November 20, 2021; accepted March 14, 2022)

Keywords: depth image, posture recognition, skeleton tracking, Kinect v2, augmented reality (AR), AdaBoost, Unity

In this study, we aim to develop a game support system that allows users to work out and rehabilitate at home alone. The system first reads the user's depth image through the Kinect v2 sensing interface, converts it into the user's skeleton, then continues to track the human body and creates the required database through machine learning after recording the dynamic changes in the skeleton. This information is then applied to a game platform designed with the Unity game engine. Finally, the game screen is connected to smart glasses via Bluetooth, allowing users to experience the game in augmented reality (AR). The database is constructed via the adaptive enhanced AdaBoost algorithm used in machine learning, and the architecture of the Unity game platform is edited in C#. The support system of the home fitness and rehabilitation game is completed after being combined with Kinect v2. There are two modes in the game platform, fitness and rehabilitation, with five movements in the fitness mode and 10 movements in the rehabilitation mode. Both modes have three sub-modes: independent training, coach demonstration, and mini-games. We demonstrated through tests that the system can allow users to easily and comfortably rehabilitate and work out at home as if there were a coach guiding them. Therefore, in addition to effectively improving the accuracy of movements, the system can also help avoid injuries or accidents caused by inaccurate movements.

1. Introduction

The outbreak of the Covid-19 pandemic in recent years has caused many changes in people's habits and reduced their willingness to go out, especially during waves of infections, causing many places, such as gyms and rehabilitation centers, to temporarily close. This has made it much more difficult to perform fitness activities and rehabilitation, which require a constant frequency of activity for a long time, resulting in people requiring fitness activities or rehabilitation to seek alternative solutions.

*Corresponding author: e-mail: chenby@ncut.edu.tw
https://doi.org/10.18494/SAM3734

ISSN 0914-4935
https://myukk.org/

The solution proposed in this paper is a support system based on the Kinect v2 sensing interface that allows users to rehabilitate or work out directly at home with the Unity game engine. Kinect v2, a depth image sensor developed by Microsoft,[1] was initially applied in the interface device of the Xbox One console so that players could directly control movements with their postures, gestures, and voice without holding a control device. In related studies, Park *et al.* proposed a depth-image-based body segmentation method[2] to improve human movement recognition, and their proposed method improved the accuracy by 15% compared with other recognition systems for custom movements such as falling or kicking. Saidin and Shukor developed a Kinect-based fall detection system,[3] which was designed to detect falls and alert caregivers by calculating the distance between each joint and the floor using a movable device installed underneath Kinect to make it easier to follow the subject's movements. The Unity game engine used in their study was developed by Unity Technologies.[4] This game engine is used to create 2D and 3D games and animations with diverse development environments, has a visual and detailed property editor, and can export real-time game previews to multiple platforms. In related studies, the virtual collaborative experimental platform created by Xia *et al.*[5] established a virtual experimental environment with the Unity platform and by simulating experiment modules with corresponding modules, and collaborative synchronization was achieved by enabling the server to adopt permission control and real-time synchronization. Mattingly *et al.*[6] constructed a 3D robot with Maya software, and used Kinect to control the robot to simulate movements and observe its simulated movements. Wu[7] developed a game by combining the Unity game engine with virtual reality (VR). Wu wrote the game with the animators in Unity, combined its scripts, and created a first-person shooting game by connecting the system with VR devices.

The main goal of this paper is to establish a game support system that enables users to work out and rehabilitate alone at home, allowing them to conduct training by themselves and be guided by a virtual trainer without leaving their home. This paper is divided into two parts. We first discuss the recognition stage, i.e., recognizing human postures using depth images and machine learning, and then establishing a database. The second part is the application, i.e., applying the database to the Unity game. The human skeleton frame constructed from the input depth image can be compared with or follow the posture required by the game to create a game that supports both rehabilitation and fitness. Finally, the user can play the game while wearing a VR device to enhance the freedom of movement and no longer be constrained by the computer screen.

2. System and Hardware Architecture

In this paper, the human skeleton is traced and recognized by using depth images with adaptive boosting (AdaBoost). A flow chart of the system is shown in Fig. 1. The system is divided into (a) the recognition stage and (b) the application stage. In stage (a), depth images are acquired by the photographic depth sensor of Kinect v2. The depth images and skeleton data are applied to the AdaBoost algorithm used in machine learning in the form of video to train the machine learning and form the database, and then the training results are tested to see whether

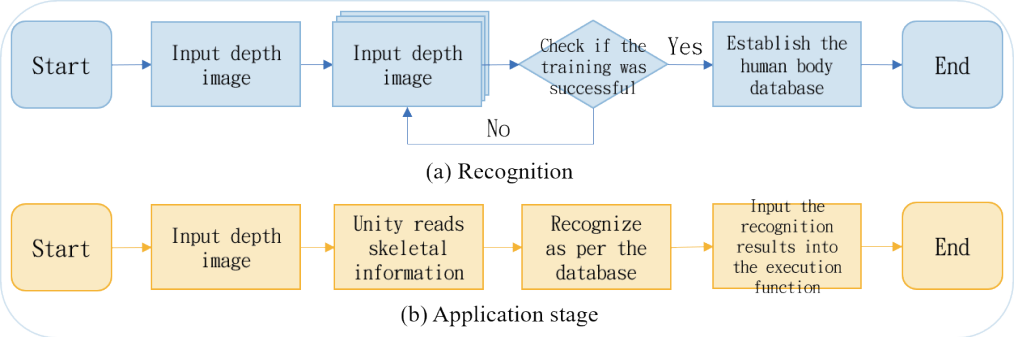


Fig. 1. (Color online) System flow chart.

they meet the criteria. If they meet the criteria, the database required for the game can be obtained, otherwise retraining is needed. In stage (b), the depth images are input to Unity, from which the current human skeleton information can be read, and the trained posture database is input to the game, allowing the game to recognize the user's current posture and give appropriate feedback.

Figure 2 shows the architecture of this hardware. The depth images are obtained by the photographic depth sensor of Kinect v2, and the human skeleton information is recognized by the software interface provided. All the information is then compiled to create the human posture database of the system and is stored in the computer. In addition, based on the Unity game interface, the game software for rehabilitation and fitness can be transmitted to the VR device for augmented reality (AR) display through Bluetooth.

3. Human Posture Recognition

As the image input terminal, Kinect v2 recognizes the image of the human body after receiving the depth image, then marks the joints of the human body and trains the human body posture dataset using the AdaBoost algorithm.

3.1 Depth image capture

The distance between the target and the sensor can be obtained by depth image capture. The time of flight (ToF) is adopted in this study to capture the depth, as shown in Fig. 3.

The ToF measurement method applied in Kinect v2 controls gate switches and generates light pulses with a high-speed driver circuit, projecting the pulsed light in a very short time such that the pulsed light hits the object in the experiment, where it is reflected. The reflected pulses of light are captured by a camera and are converted into a depth image by calculation. The data required is the time (t_D) from the projection to the reflection of the pulsed light and the distance (D) between the object and the depth lens. The following equation is satisfied, where c is the speed of light (approximately 3×10^8 m/s).

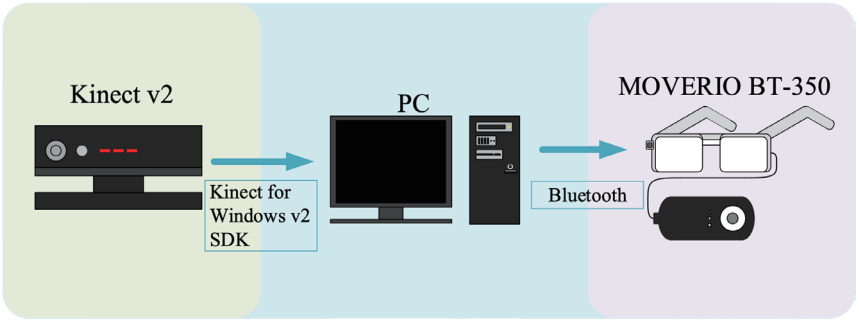


Fig. 2. (Color online) Hardware architecture.

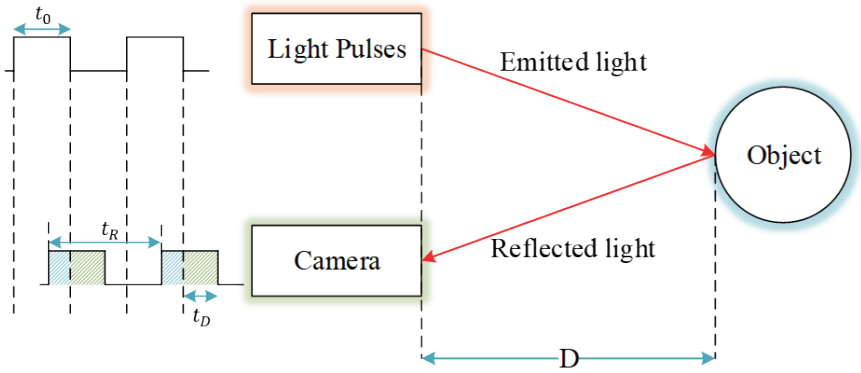


Fig. 3. (Color online) ToF.[8]

$$D = \frac{1}{2} c t_D \tag{1}$$

In this study, to confirm that the sensed distances are correct, the measured distances in Fig. 4 are compared with those output by the sensor in Fig. 5.

Table 1 shows that the distances obtained by depth sensing via the ToF method are nearly the same as the measured distances, thus confirming the accuracy of the depth measurement.

3.2 Skeleton tracing

The human skeleton is traced in three stages as shown in Fig. 6. The depth image of the human body is first recognized by the depth extraction method, then the recognized image is applied to classify and read different parts of the human body through the random forest algorithm. Finally, the joint points in the human skeleton are obtained through the mean shift algorithm.

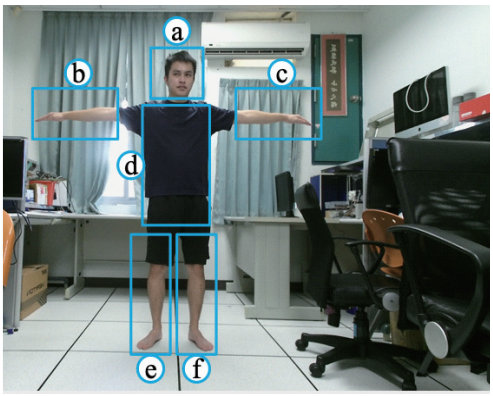


Fig. 4. (Color online) Measured distances.

Fig. 5. (Color online) Sensed distances.

Table 1
Comparison of distances.

	(a) Head	(b) Right hand	(c) Left hand	(d) Body	(e) Right foot	(f) Left foot
Measured distance (mm)	2403	2532	2530	2104	2250	2250
Sensed distance (mm)	2400	2530	2530	2100	2250	2250

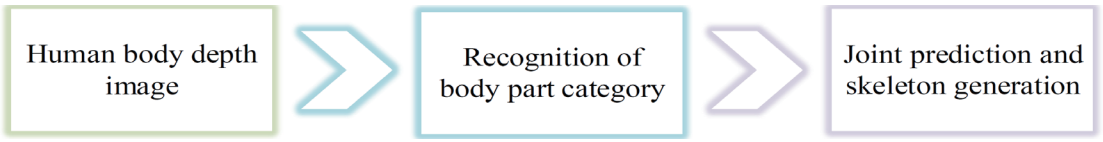


Fig. 6. (Color online) Skeleton recognition process.

3.2.1 Depth image of the human body

To reduce the computational burden and complexity of the subsequent processing, the unnecessary background images are removed by depth feature extraction, leaving only the desired human body masks. The calculation process is expressed as[9]

$$f_\theta(I,x)=d_I\left[x+\frac{u}{d_I(x)}\right]-d_I\left[x+\frac{v}{d_I(x)}\right]. \quad (2)$$

In Eq. (2), the depth of pixel x of image I is $d_I(x)$ and the offset vector is $\theta = (u,v)$, as shown in Fig. 7, where X denotes pixel x, i.e., the pixel being classified as either the human body or the

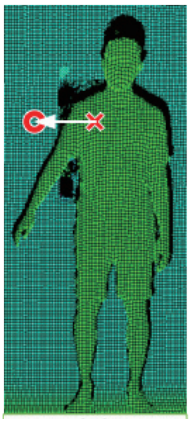


Fig. 7. (Color online) Schematic diagram of the depth feature extraction method.

background, and O indicates the position of the offset pixel. The images are classified according to the difference between the two depths.

When the offset and classified pixels lie in the background and the human body, respectively, as in Fig. 7, the depth of the pixel in the background will be greater than that of the human body, and when the difference between the depths is too large, the pixel with the greater depth will be considered as the background and removed to distinguish between the background and the human body.

3.2.2 Recognition of body part category

In body part recognition, the above depth feature extraction method temporarily marks the neighboring areas with similar depths as the same part, and then classifies the human body blocks by the random forest algorithm. This is a prediction model that combines T decision trees, where each split node consists of a feature f_θ and a threshold τ that are used to classify pixel x in image I. Each branch of the tree is derived by comparison with a threshold τ. The terminal node is reached at the tth tree, with the learning distribution $P(c|I,x)$ of the body part label c stored, and the final classification results are obtained by averaging and unifying all distributions, as follows:[10]

$$P(c \mid I,x) = \frac{1}{T}\sum_{t=1}^{T} P_t(c \mid I,x). \tag{3}$$

The samples are also trained using decision trees. Different training samples are adopted for each decision tree, and the samples may be classified by body type or posture. Each pixel in the depth image of the human body to be measured is recognized by the decision tree to obtain the probability that it is a specific part of the human body, and then the probability determined by each decision tree is calculated synthetically to complete the classification of the part of the human body.

3.2.3 Joint prediction and skeleton generation

The final stage is to search for the positions of the joint points of the classified human body, for which the mean shift clustering algorithm is used in the Kinect operation,[11] as shown in Fig. 8.

The mean shift clustering algorithm selects a point in the dataset, draws a circle with this point as the center, finds the mean of the vectors from this point to all the points in the circle, merges the vector mean with the center of the circle to derive the new center of the circle, and iterates these steps until it converges to a stable point. The position of this point is the final selected position, which is the position of one of the joint points recognized in the human skeleton. The results are shown in Fig. 9.

Kinect v2 can acquire 25 joint points, namely, the head, lower jaw, collar, spine, pelvic center, and the left and right joints of the shoulder, elbow, wrist, palm, fingertips, thumb, hip, knee, ankle, and foot. The joints are presented in 3D images. The system can sense a total of six human skeletons at the same time, and the position and direction of each joint can be obtained.

3.3 Construction of human posture database

In this study, posture recognition is applied to construct a human posture classification database by machine vision using AdaBoost. The architecture of AdaBoost is shown in Fig. 10. Each classifier of a given training set is assigned an equal weight. If a sample has been correctly classified in the training, then the weight of the sample is decreased when constructing the training set of the next classifier. In contrast, when a sample is not accurately classified, then the weight of the sample is increased and the updated weight of the sample set is input to the next classifier to be trained. The iteration is continued for a set number of times, then the trained weak classifiers are combined into a strong classifier. When weak classifiers are combined, the weights of weak classifiers with a low error rate are increased and the weights of those with a large error rate are decreased, thus increasing the accuracy of the final strong classifier.

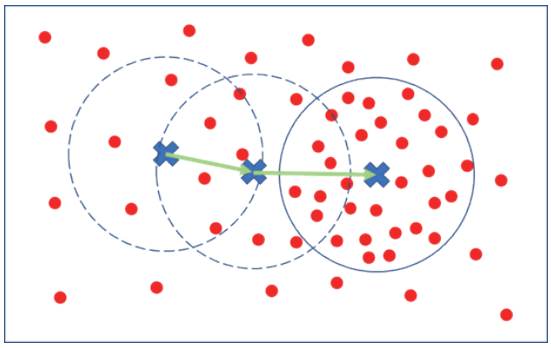


Fig. 8. (Color online) Mean shift algorithm.

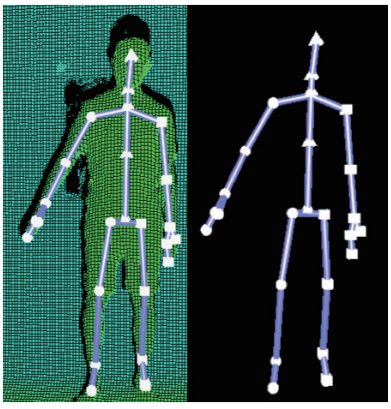


Fig. 9. (Color online) Human skeleton frame.

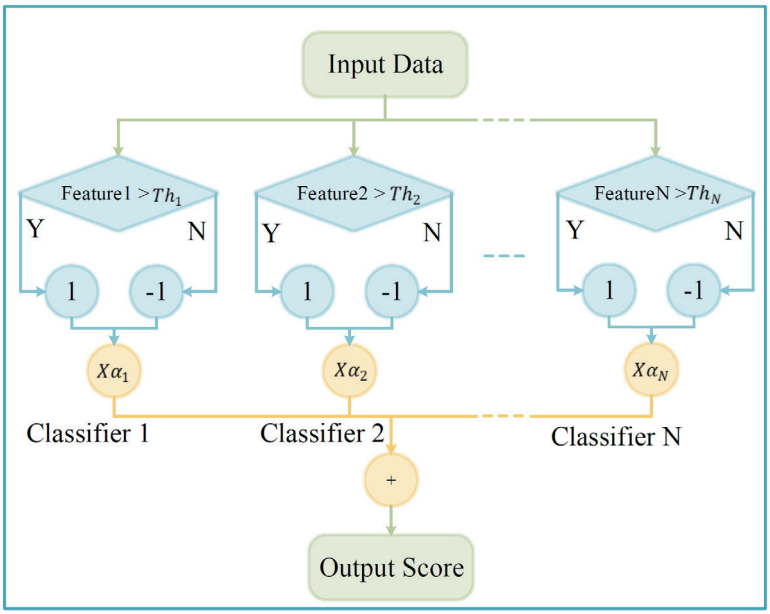


Fig. 10. (Color online) AdaBoost architecture.[12]

The first step of the AdaBoost algorithm is to input various parameter messages, including accuracy conditions, weak classifier parameters, and the number of weak classifiers. Then the training set samples, expressed as Eq. (4), are input.[13]

$$S = \{(x_1, y_1),\ (x_2, y_2) \ldots (x_n, y_n)\} \tag{4}$$

Here, x_n is a feature vector of feature space X and $y_n \in \{-1, +1\}$ is a marker. When $y_n = +1$, x_n is a positive sample, and when $y_n = -1$, x_n is a negative sample. n is the number of weak classifiers.

The weight distribution D of the training data is initialized by setting all the weights to $1/n$:

$$D_1(i) = \frac{1}{n},\ i = 1, 2, \ldots, n\ . \tag{5}$$

The second step is to perform several iterations to train the weak classifiers h_t based on the weight distribution, as follows:

$$h_t : R^d \to \{-1, +1\}, \tag{6}$$

where t is the iteration number, $t = 1, 2, \ldots, T$, and T is the maximum number of iterations.

The classification error of each weak classifier is then calculated. The sum of the misclassified sample weights is indicated by ε_t when the classification error satisfies the property $h_t(x_i) \neq y_i$:

$$\varepsilon_t = \sum_{i:h_t(x_i)\neq y_i} D_t(i). \tag{7}$$

The weight factor α_t of weak classifier t is then computed as

$$\alpha_t = 0.5\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right). \tag{8}$$

The weight distribution $D_{t+1}(i)$ of the training data is updated for the next iteration. When the sample is correctly classified, the weight is decreased, and if the sample is incorrectly classified, the weight is increased to improve the correctness of the next iteration, expressed as

$$D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{z_t}, \tag{9}$$

where Z_t is the denominator set by normalization so that the sum of the updated weights is 1, expressed as

$$Z_t = \sum_{i=1}^{n} D_t(i)e^{-\alpha_t y_i h_t(x_i)}. \tag{10}$$

Finally, after T iterations, the output strong classifier is a linear combination of the selected weak classifiers, as shown in Eq. (11). Each weak classifier is assigned a property according to the obtained weight, then the assigned weak classifiers are combined to obtain the high-resolution strong classifier ($H(x)$) .

$$H(x) = sign\left[\sum_{t=1}^{T} \alpha_t h_t(x)\right] \tag{11}$$

In the training set, the postures to be recognized are marked as positive samples, while the postures not to be recognized are marked as negative samples. The iteration process is stopped when the accuracy of the training set reaches 95%, and the parameters are adjusted and recalculated using Eq. (6) if the expected conditions are not reached. The human posture recognition database used in this study is output if the expected accuracy is reached.

The error rates of the final movement database are shown in Tables 2 and 3, which respectively indicate the error rates of the movements in the rehabilitation mode and the fitness mode under coaching, presented as confusion matrices.

Table 2
(Color online) Confusion matrix of rehabilitation movement recognition.

Confusion matrix		Actual movements (number of times)										
		Both hands upward	Left hand upward	Right hand upward	Left leg side raises	Right leg side raises	Left leg rear raises	Right leg rear raises	Left leg front raises	Right leg front raises	Right leg front raises	Right leg straight raises
Predicted movements	Both hands upward	50	0	0	0	0	0	0	0	0	0	0
	Left hand upward	0	50	0	0	0	0	0	0	0	0	0
	Right hand upward	0	0	50	0	0	0	0	0	0	0	0
	Left leg side raises	0	0	0	46	0	2	0	2	0	0	0
	Right leg side raises	0	0	0	0	46	0	3	0	3	0	0
	Left leg rear raises	0	0	0	2	0	48	0	2	0	0	0
	Right leg rear raises	0	0	0	0	2	0	47	0	0	0	0
	Left leg front raises	0	0	0	2	0	0	0	46	0	2	0
	Right leg front raises	0	0	0	0	2	0	0	0	47	0	1
	Left leg straight raises	0	0	0	0	0	0	0	0	0	48	0
	Right leg straight raises	0	0	0	0	0	0	0	0	0	0	49
Error rate (%)		0	0	0	8	8	4	6	8	6	4	2
Total error rate (%)		4.18										

Table 3
(Color online) Confusion matrix of fitness movement recognition.

Confusion matrix		Actual movements (number of times)				
		Deep squat	Left lateral lunge	Right lateral lunge	Left lunge	Right lunge
Predicted movements	Deep squat	48	3	2	0	0
	Left lateral lunge	1	47	0	1	0
	Right lateral lunge	1	0	48	0	1
	Left lunge	0	1	0	49	0
	Right lunge	0	0	1	0	49
Error rate (%)		4	8	6	2	2
Total error rate (%)		4.4				

It can be seen from the tables that the accuracies of the rehabilitation and fitness movement databases obtained in this study are 95.82 and 95.6%, respectively. Both values satisfy the stopping condition; thus, the databases can be applied to the Unity game created in this study.

4. Design of Fitness and Rehabilitation Support Game in Unity

The construction of the game using the Unity game engine is introduced from the outside to the inside, as shown in Fig. 11. The project, in simple terms, is the body of the game. By creating a project, multiple scenes are combined, where each scene is a level of the game. The scenes are sorted and edited by scripts in the project to control the order and manner of each level.

The scene is usually a game level or an option page, and can be switched through scripts based on user instructions or changes to the state in the game. The main scenes in this study are divided into two main categories (fitness and rehabilitation), each with three sub-categories of scenes (independent training, coach demonstration, and mini-game) with their own option pages.

The animator orders the recorded animations and writes the script to the trigger. When triggered, characters or objects act in accordance with the recorded movements. Therefore, the animations are recorded first, with their characters made, and then the movement sequence and connection are arranged in the animator. Finally, the script is written for a successful trigger.

4.1 Scripts

The script is a game component that controls the behavior and state of game objects in an additional way, which can make the maintenance and construction of scripts easier. The same script can be attached to different game objects at the same time, and different scripts can be attached to the same object at the same time. The scripts mainly used in this study (Kinect

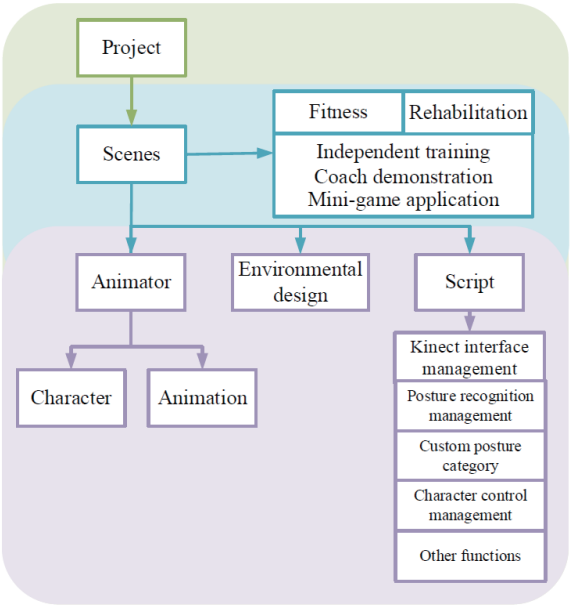


Fig. 11. (Color online) Unity game architecture.

interface management, custom posture categorization, posture recognition management, character control management, and other functions) are described in the following. We introduce their functions and main construction strategies in Fig. 12.

(1) Kinect interface management

This script enables Unity to connect with Kinect. It reads the skeleton information obtained by Kinect and the custom posture categories.

(2) Custom posture category

There are two custom posture categories. One is the category in the posture database trained as described in Sect. 3, which is input to the written script and read for determination. The other is the category that is defined directly when the joint is at the set position by writing the desired joint position manually.

(3) Posture recognition management

Upon posture recognition, the accuracy of the posture is determined. When the set accuracy is met, the posture is recognized as the current posture, and when the recognition is completed, different feedback is obtained with the levels set in the game, so that the player can understand the correct posture of the movement or the part that needs to be improved.

(4) Character control management

The skeleton information sensed by Kinect can not only be used to identify the current posture, but also be applied to the designed characters so that they can make the same movements as the users. When the posture required by the conditions is identified, the characters can make the movements set in the animator. When they are used in different characters and appear in the scene at the same time, coach and user embodiment can be achieved.

(5) Scripts for other functions

In addition to the above scripts, all the functions in Unity require corresponding scripts to operate, such as level switching, changing the user interface (UI), setting a scene, or attaching physical phenomena such as a rigid body and gravitational force to the game objects. The diversity of game settings can be increased with the application of scripts.

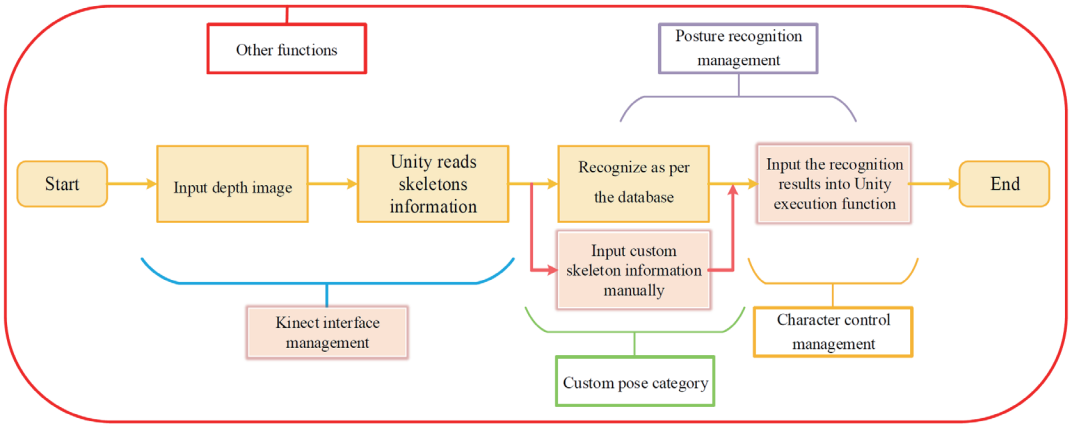


Fig. 12. (Color online) Application scope of scripts.

5. Game Design and Results

In the experiments in this study, the human skeleton information is obtained by the skeleton tracking system of Kinect v2, and the database is made by the AdaBoost algorithm, which is then input to the game system designed in this study. The following introduces the completed home rehabilitation and fitness game support system.

5.1 Level setting

After opening the game, the user first enters the game menu and selects the desired level through the options interface. The user then chooses the desired activity. Figure 13 shows the composition of the levels.

The user can select the desired level by tapping the buttons on the screen or return to the previous level via the "Back" button. To quit the game, the user can tap the "Quit" button on the main menu to close the game program, as shown in Figs. 14 and 15.

5.2 Coaching

The game screen of the coaching mode is shown in Fig. 16. There are two character modules on the screen, one of which acts as the user's avatar and the other acts as the coach to guide the user. The coach demonstrates first while the user follows the coach and tries to make the correct movements. Each UI and the process in the game are described as follows.

(1) Movement display UI

If the user's current movement is recognized as a posture in the database, its name is displayed here.

(2) Movement reminder UI

This UI displays the name of the next movement to be performed and the coach performs the same movement.

(3) Character embodiment

The right character embodies the user, who reproduces the current posture of the user captured by Kinect, while the left female character embodies the coach, who demonstrates the correct movements in order, as shown in Fig. 17.

(4) Timing UI

The timing UI records the time the user takes to complete all the movements. Timing starts from the moment the user enters the level and ends when the user has performed all the specified movements. This time is shown on the screen, which allows the user to compare it with the previous time taken for the user to complete the game to ascertain whether the performance of the user has improved.

(5) Task bar UI

The task bar shows the proportion of the number of movements the user has completed relative to the total number of movements in the level.

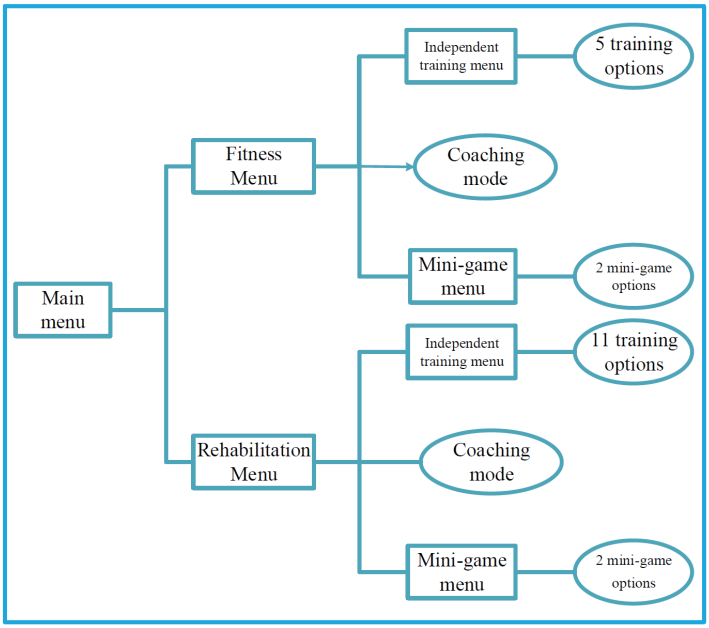


Fig. 13. (Color online) Composition of levels.

Fig. 14. (Color online) Main menu window.

Fig. 15. (Color online) Fitness menu window.

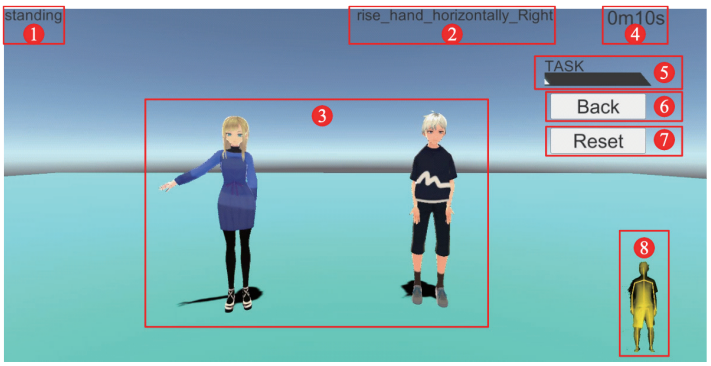
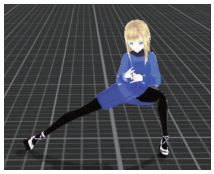
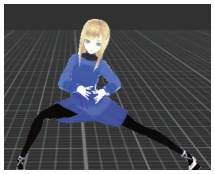
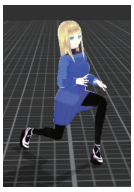
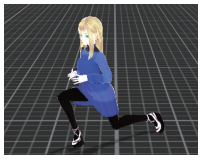
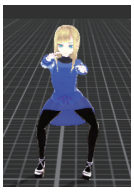
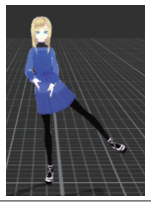
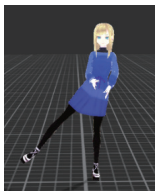
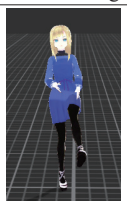
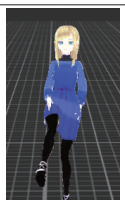
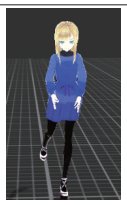
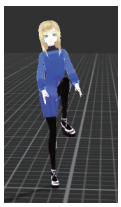
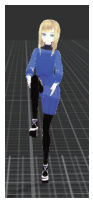
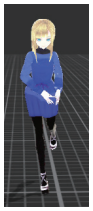
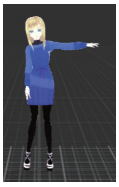
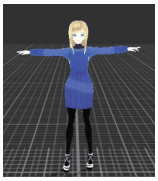
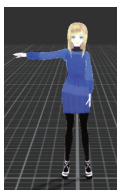


Fig. 16. (Color online) Interface of coaching mode.

Fitness movements		
		
Left lateral lunge		Right lateral lunge
		
Left lunge	Right lunge	Deep squat

(a)

Rehabilitation movements		
		
Left leg side raises		Right leg side raises
		
Left leg front raises	Right leg front raises	Right leg rear raises
		
Left leg straight raises	Right leg straight raises	Left leg rear raises
		
Left hand upward	Both hands upward	Right hand upward

(b)

Fig. 17. (Color online) (a) Fitness movements and (b) rehabilitation movements of coach demonstration.

(6) Back button

Tapping this button stops all functions in the coaching mode and returns to the previous level menu.

(7) Reset button

Tapping this button resets the level to the initial state, and the movement reminder UI and the coach return to the first movement and start over again. The task bar UI and timing UI are also reset.

(8) Depth image of human body

The current skeleton information captured by Kinect is displayed on the screen, from which the user can learn the current sensing status.

As an example, Table 4 shows the reference completion time for each movement in the fitness mode. A time close to 10 s is optimal, but the time should not be less than 10 s. We employed five testers for training, and the time was recorded to determine the progress of each user.

Tables 5–9 show the times of the movements completed by the five testers. Each tester was trained five times with five repeated movements.

It can be observed from Tables 5–9 that although each tester took a different amount of time to perform all the movements, less time was required for each movement as the amount of training increased, demonstrating that this system can effectively improve the efficiency of movement of the user.

5.3 Independent training

The user selects the movements to be trained from the level and enters the independent training mode. The game screen is shown in Fig. 18, with the character in the screen embodying the user, whose current posture is displayed. The game screen shows the completeness of the user's current movement and the points that need improvement. The UIs in Fig. 18 are as follows.

(1) Movement reminder UI

The reminder UI at the top of the screen reminds the user of the imperfect parts of the movements, with the necessary improvement expressed in words. The lower circle indicates the completion status of the movement by the proportion of the green part. The circle is gray in the case of a completely incorrect posture.

(2) Movement display UI

This UI displays the name of the user's current movement.

(3) Character embodiment

The character embodies the user, whose current posture captured by Kinect is reproduced.

(4) Task bar UI

The task bar shows the proportion of the number of correct movements the user has completed.

(5) Back button

Tapping this button stops all functions in the independent training mode and returns to the previous level menu.

Table 4
Reference times for movements.

Name of movement	Time (s)
Deep squat	10
Left lunge	10
Right lunge	10
Left lateral lunge	10
Right lateral lunge	10
Total time	50

Table 5
Time of movements completed by tester 1.

Number of times	Deep squat	Left lunge	Right lunge	Left lateral lunge	Right lateral lunge	Total time (s)
1	15	14	16	17	16	78
2	15	16	14	14	14	73
3	12	13	13	12	12	62
4	11	11	12	11	12	57
5	10	10	11	10	11	53

Table 6
Time of movements completed by tester 2.

Number of times	Deep squat	Left lunge	Right lunge	Left lateral lunge	Right lateral lunge	Total time (s)
1	16	15	16	18	15	80
2	17	15	14	15	15	76
3	14	14	13	15	13	69
4	12	13	13	13	12	63
5	11	11	12	12	11	57

Table 7
Time of movements completed by tester 3.

Number of times	Deep squat	Left lunge	Right lunge	Left lateral lunge	Right lateral lunge	Total time (s)
1	18	17	17	19	20	91
2	17	17	18	18	18	88
3	17	15	15	18	17	82
4	15	15	15	16	17	78
5	13	13	13v	15	15	69

Table 8
Time of movements completed by tester 4.

Number of times	Deep squat	Left lunge	Right lunge	Left lateral lunge	Right lateral lunge	Total time (s)
1	19	20	19	20	20	98
2	18	18	18	20	19	93
3	18	17	17	18	18	88
4	17	17	16	17	17	84
5	15	17	15	16	15	78

Table 9
Time of movements completed by tester 5.

Number of times	Deep squat	Left lunge	Right lunge	Left lateral lunge	Right lateral lunge	Total time (s)
1	17	18	17	16	15	83
2	16	17	17	15	16	81
3	14	15	16	13	14	72
4	13	14	14	12	12	65
5	12	12	12	11	11	58

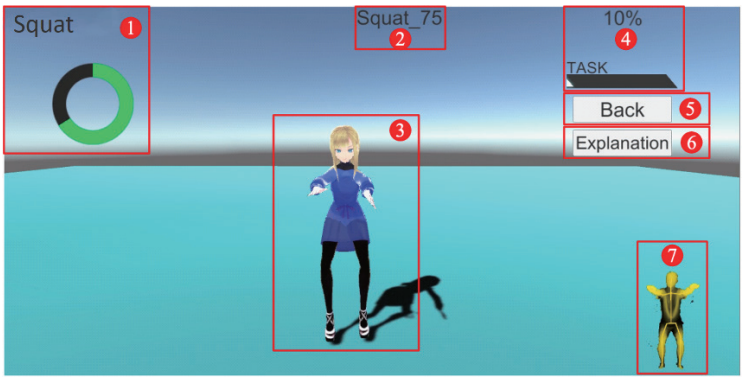


Fig. 18. (Color online) Independent training mode.

(6) Explanation button

Tapping this button opens the movement details page, where users can learn about points needing attention regarding the movement.

(7) Depth image of human body

The current skeleton information captured by Kinect is displayed on the screen, from which the user can learn the current sensing status.

5.4 Mini-game application

The mini-game applies the posture recognition method used in the previously described games into simple games, as shown in Fig. 19. The user controls the yellow ball to move left and

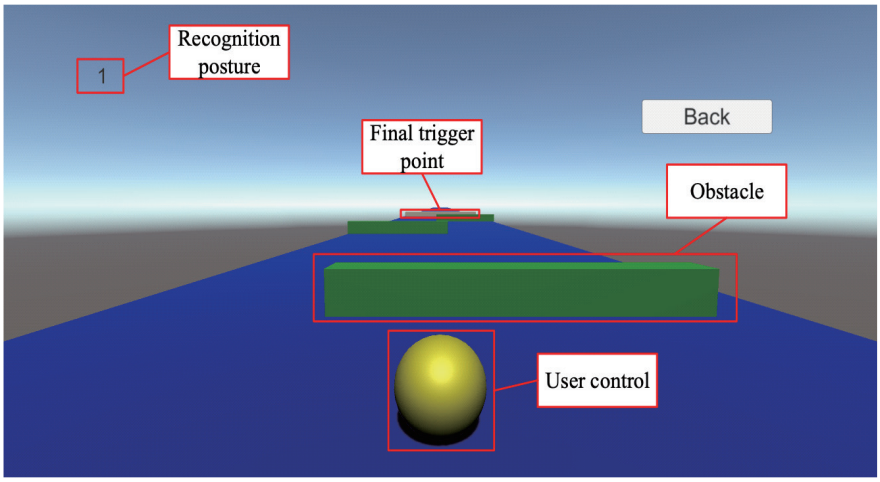


Fig. 19. (Color online) Mini-game application mode.

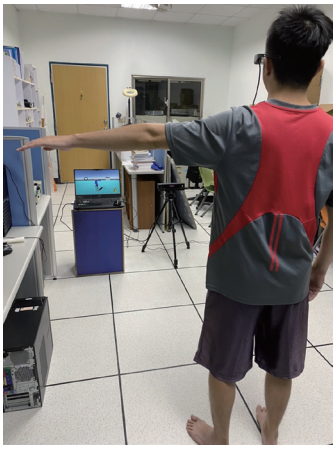


Fig. 20. (Color online) Actual setup with AR device.

right with the posture selected in the level (the movements in this mode are divided into left and right types) while avoiding green obstacles or falling off the platform. When the yellow ball hits the gray final trigger point, the user has completed the game successfully.

5.5 Application combined with AR device

Finally, the game screen is transmitted to the AR device via the Bluetooth module, so that the screen can follow the user's movements and the user can move without being constrained by the position of the computer screen. The actual setup is shown in Fig. 20.

6. Conclusion

We have developed a home fitness and rehabilitation support system that allows users to properly rehab or work out alone at home, improve the accuracy and enjoyment of training, and thereby reduce the likelihood of injury and motivate them to regularly carry out training. The system is divided into two main parts: the recognition stage and the application stage. In the recognition stage, as the image input terminal, Kinect v2 recognizes the image of the human body after receiving the depth image, marks the joints of the human body, and then trains the human body posture dataset using the AdaBoost algorithm. In the application stage, the trained posture dataset is input to the Unity game, so that the game can recognize the accuracy of the user's movements. There are two main modes of the game. In the coaching mode, movements are demonstrated by the coach through an animator with remade data input, so that users can follow the movements of the coach. In the independent training mode, the user can be trained after selecting the desired movements. The user's current posture, the accuracy rate of the movement, and the improvement are displayed on the game screen as feedback so that the user can understand the correct movement posture. Therefore, in addition to effectively improving the accuracy of movements, the system can also help avoid injuries or accidents caused by inaccurate movements.

Acknowledgments

This work was supported by the Ministry of Science and Technology, Taiwan, under contract number: MOST 110-2221-E-167- 034-.

References

1 Microsoft, Kinect for Windows: https://developer.microsoft.com/zh-tw/windows/kinect/ (accessed July 2021).
2 S. Park, U. Park, and D. Kim: Proc. 2018 Int. Conf. Electronics, Information, and Communication (ICEIC, 2018) 483–485.
3 N. A. Saidin and S. A. A. Shukor: Proc. 2020 8th IEEE Conf. Systems, Process & Control (ICSPC, 2020) 220–224.
4 Unity Technologies, Unity: https://unity.com/ (accessed July 2021).
5 X. L. Xia, D. Honghai, L. Jianfeng, and Z. Hao: Proc. 2018 IEEE Int. Conf. Safety Produce Informatization (IICSPI, December 2018) 546–550.
6 W. A. Mattingly, D.-J. Chang, R. Paris, N. Smith, J. Blevins, and M. Ouyang: Proc. 2012 17th Int. Conf. Computer Games (CGAMES, October 2012) 56–59.
7 J. Wu: Proc. 2020 Int. Conf. Computer Vision, Image and Deep Learning (CVIDL, November 2020) 592–595.
8 S. Gokturk, H. Yalcin, and C. Bamji: Proc. Conf. Computer Vision and Pattern Recognition (CVPR 2004, July 2004).
9 J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake: Proc. Conf. Computer Vision and Pattern Recognition (CVPR 2011, August 2011) 1297–1304.
10 Medium, Random Forest Simple Explanation: https://williamkoehrsen.medium.com/random-forest-simple-explanation-377895a60d2d (accessed July 2021).
11 Y. Chen, P. Hu, and W. Wang: Proc. 2018 11th Int. Congr. Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI 2018, October 2018) 1–5.
12 K. Kim and H. I. Choi: Proc. 2017 4th Int. Conf. Computer Applications and Information Processing Technology (CAIPT, August 2017) 1–5.
13 E. W. Trejo and P. Yuan: Proc. 2018 3rd Int. Conf. Advanced Robotics and Mechatronics (ICARM, July 2018) 606–611.