Multimed Tools Appl (2017) 76:12281–12314
DOI 10.1007/s11042-016-3641-6

Designing an exergaming system for exercise bikes using kinect sensors and Google Earth

Shih-Yu Huang[1,2] · Jen-Perng Yu[3] · Yuan-Kai Wang[4] · Jia-Wei Liu[1]

Received: 23 October 2015 / Revised: 30 March 2016 / Accepted: 26 May 2016 /
Published online: 2 June 2016

Abstract This paper proposes an Exergaming system for exercise bikes. With the assistance of a Kinect device and the proposed body-movement-detection algorithm, exercise bike users are required to perform correct neck and shoulder movements to control the airplane trajectory in Google Earth. They can take a flying tour in the virtual reality provided by Google Earth while riding an exercise bike. According to the experimental results, 95 % of the users in the experiment considered the proposed Exergaming system to be very entertaining; more than 85 % of the users affirmed that the assigned neck and shoulder movements effectively help stretch the muscles in these body parts; the detection rate of the proposed body-movement algorithm was over 90 %. Therefore, the proposed Exergaming system is a good assisting system for exercise bikes.

Keywords Exergaming · Kinect · Google Earth · Exercise bike

1 Introduction

People nowadays spend much of their time sitting in front of a computer. They suffer from neck and shoulder pain or stiffness because they stay in a certain position for too long. Regular exercise can cure their pain. Some people walk or jog outdoors. Others ride exercise bikes indoors. Exercise bikes are a popular kind of fitness equipment. Their main purpose it to train cardio-respiratory capacity; they also provide a good workout for the legs, buttock, and gluteal

✉ Shih-Yu Huang
syhuang@mail.mcu.edu.tw

1 Department of Computer Science and Information Engineering, Ming Chuan University, Taipei, Taiwan

2 Taoyuan: 5 De Ming Rd., Gui Shan District, Taoyuan County 333, Taiwan

3 Department of Information Management, Ming Chuan University, Taipei, Taiwan

4 Department of Electrical Engineering, Fu Jen University, Taipei, Taiwan

muscles. However, exercise bikes have two drawbacks. First, exercise bikes are not capable of providing a whole-body workout. For example, the muscles in the neck and shoulder do not exercise much in the process. Second, riding an exercise bike is monotonous. Rider users usually watch TV or listen to music while riding.

Exergaming is a new technique [19]. Its purpose is to allow users to exercise and play a game at the same time so as to obtain the workout effect in a pleasurable environment. Google Earth [22] is a virtual globe software developed by Google Inc. This software places satellite images, aerial photographs, and GIS in a three-dimensional model. Google Earth street view shows 360° panoramic street photographs according to users' demand. This software also contains a 3-D building module. Google Earth is a well-known application that allows users to explore the world in a very entertaining way. The Kinect [8] sensor is a somatosensory system launched by Microsoft. It is a sensor constituted of a RGB camera, a depth sensor, and multiple microphones. Kinect is able to obtain RGB images, depth value, and human skeleton information. The two new techniques discussed above were combined in this paper to develop an Exergaming system for exercise bikes. The Kinect device was used to detect users' body movements when riding an exercise bike; the corresponding flight control was activated in Google Earth. As a result, the users could take a virtual flying tour in Google Earth's virtual scenarios when riding an exercise bike. The result made their dreams of flying in the sky come true. However, the users were required to perform correct neck and shoulder movements in order to control the airplane trajectory or the flying mode in Google Earth. With the proposed system, the limit of exercise bikes, i.e., only being able to train the lower part of the body, was improved. Exercise bikes become a more comprehensive workout tool with the assistance of the proposed system.

Three problems have to be carefully considered to achieve the above goal. The first one is how to efficiently detect the body movements from Kinect. The second one is how to precisely detect the shoulder and neck movements. The third one is how to design some aerobatics in Google Earth to increase the motivation of users. The proposed Exergaming system for exercise bikes consists of three main functional modules to solve the above problems. The first one is the aircraft motion detection module in Google Earth which obtains human skeleton information through Kinect. The information is used to define multiple aircraft motion events, including the airplane's speed control, turn control, and climb control. Controlling this module allows the users to take a virtual flying tour in Google Earth's virtual world. The second on is the neck and shoulder movement detection module which uses the Kinect somatosensory device to obtain the users' human skeleton information and face postures when they are performing neck and shoulder gestures. The third one is the aerobatics module. It includes some specially-designed aerobatics modes such as vertically spinning the airplane, circling-around certain landmarks (Taipei 101, for instance), or travelling above the River Thames in England to increase the motivation of users to perform neck and shoulder movements. These movements were based on Pilates moves.

The structure of this paper is described as follows. Section 2 introduces related work applying the Kinect technique to Exergaming for rehabilitation. Section 3 describes the proposed system and techniques. Section 4 discusses the experimental results. Section 5 offers the conclusion and suggestions for future work.

2 Related work applying the Kinect's technique to exergaming for rehabilitation

Traditionally, evaluating the effectiveness of a patient's rehabilitation treatment requires observation and diagnosis. Most evaluation methods require at least one physical therapist or occupational therapist to execute a one-on-one therapeutic process. This kind of process takes a great deal of time and patience. Exercise is a form of treatment in physical medicine and rehabilitation for patients who suffer from musculoskeletal disorders. These patients are usually treated with clinical therapeutics or in-home therapeutics. Possible problems they encounter involve the lack of continual rehabilitation behavior and incorrect rehabilitation movements. However, therapeutic Exergaming can solve the above problems.

Exergaming combines exercise with games. Therapeutic Exergaming includes exercise therapy (rehabilitation), additionally. The gaming aspect attracts patients and increases their wiliness to undergo rehabilitation treatment and do more physical activity. This kind of system is also called a virtual-reality rehabilitation system. More and more studies show that virtual-reality rehabilitation systems have a positive effect in the field of physical medicine and rehabilitation.

One work [10] mentioned that sensors and computing could be applied to the human-motion-capture technique in rehabilitation engineering. This technique could record human exercise trajectory; it offers high precision and highly reliable data. Furthermore, analysis of captured motions leads to better clinical trial results and behavior evaluation. Motion-capture sensory devices can be categorized into two types: the contact type and non-contact type. Contact sensors must be placed on the human body. These sensors have good precision, but they may make the wearer feel restrained. Moreover, it takes too long to install and uninstall this kind of sensor. The latter type of sensor captures the positions of human joints using a computer vision technique. In the remainder of this section, common motion-capture sensors using computer-vision-based techniques are discussed.

Computer-vision-based motion-capture sensors capture the features of movements through RGB image processing. A previous study [20] used a digital-image-capturing-based rehabilitation system and executed clinical trial assessment. The above study developed a low-cost vision-based movement-tracking system. The system used the concept of three-dimensional space. Two vertical-view-point cameras were installed in the system, and a VR game was developed on the basis of a network. This game enabled the repetition of rehabilitation exercises. A dual-camera motion-capture system was adopted in another work [4]. The rehabilitation system they developed used a webcam and a thermal sensor to recognize hand gestures. The above work evaluated the color-image-fragmentation method and the movement-capture method, and compared the obtained results. The infrared (IR) mark-based detecting systems on the market use infrared cameras. This kind of systems can obtain precise infrared reflection marks, or IR-3D-posation LED marks.

Depth image sensors are a newly developed technique which offers an alternative to RGB image sensors. A depth-image sensor can capture the distance between a camera and the surface of the object, and therefore obtains the depth information. As a result, dividing an image into separate parts and removing the image background becomes much easier and more accurate. Human skeleton tracking also becomes possible with depth information.

In two papers [7, 18], the author mentioned that the detecting accuracy of common image techniques is negatively influenced by factors such as clothes, hair, and skin color. Although a depth camera helps reduce this kind of negative influence, major body changes and clothing shapes still produce interference. However, a depth image sensor is still superior to traditional sensors. Since human body postures vary greatly, a large training dataset of human behaviors, including movements like dance, kicks, walking, and so on, are required. Other works [3, 12] have also described a new way of tracking body postures. The key technique is to use the movement characteristics of certain parts of the body, to separate and reassemble the body with pixels, and find the 3-D joint points by simply comparing the depth pixels.

Kinect is a sensory system which combines a RGB camera, a depth sensor, and multiple microphones. Kinect can capture RGB images, depth images, and skeleton information. In two studies [13, 14], the author proposed a Kinect-based rehabilitation system to help patients exercise. The users of the system played "Tai-chi" at home. The rehabilitation system could evaluate whether the patient's Tai-chi movements were correct or whether their movements had the rehabilitation effect. The skeleton tracking of Kinect verified the patients' arm gestures as they performed them. The system compared the patients' gestures with the pre-recorded gestures of the coach and conducted evaluation. The experimental result showed that the patients' were a lot more willing to use this rehabilitation system. Using the Kinect system to practice Tai-chi at home was a great help for rehabilitation. One study [15] has provided an attempted to evaluate Kinect's performance and discussed Kinect's tracking accuracy, especially the solidness and reliability of Kinect. The author set standards for body training and Kinect's precision. As demonstrated by the experimental result, Kinect is a good choice. Another work [23] used Kinect to test a mark-based solid photo shooting system by the traditional method. The above study indicated Kinect's calibration could generate better precision with the use of a 3-D movement capturing system.

The system proposed in several works [1, 6, 11] was constituted of virtual reality and some designed rehabilitation movements. The system used Kinect to detect the patient's movement and perform face recognition and speech recognition (SR). The system was used on stroke patients' rehabilitation with some Kinect training. Compared to expensive traditional rehabilitation systems, Kinect is superior in cost and in clinical practice. Another work [25] has designed a virtual-reality rehabilitation system with high fidelity. The system used an assistant occupational therapist which was constructed by Kinect's somatosensory devices and adopted an unscented-Kalman-filter-based human body posture tracking algorithm to predict the posture of the upper arms. The virtual animation watched by the users reacted fluently to their real postures. The system is very effective for upper arm rehabilitation practice in daily life.

In addition to the Kinect sensor, Sony's PlayStation and Nintendo's Wii are two other often employed Exergaming sensors. One work [21] has attempted to compare the specifications between Sony's PlayStation, Nintendo's Wii, and Microsoft Xbox 360's Kinect by using different Exergaming interfaces in various rehabilitation projects. Therapeutic exercise and exercise capture were required during the course. They tried various rehabilitation games and invited 15 female patients who suffered from fibromyalgia syndrome for testing. The results revealed that Kinect had good effects.

The related works to retrieve and recognize object's pose and gesture from 3D depth data can be classified into shape model and kinematic model [2]. The shape model represents human body with abundant information including shape and appearance (such as flesh and skin) by volumetric or surface-based approaches. Challenges of pose deformations, multi-view and occlusion are solved by robust matching algorithms such as Hausdorff distance learning [5] and multi-modal graph learning [27]. Action retrieval by 3D depth data becomes more complicated because of the dynamics of gestures, and temporal similarity should be measured by robust algorithms such as dynamic time warping [26]. The kinematic model is a more economic and efficient method than the shape model [17]. It represents human poses with skeletons that consist of only segments of linked joints. Robust matching and retrieval of human gestures are then achieved by Bayesian networks [24] and finite state machines [9]. For pragmatic applications such as the exergaming in this paper, instant response with accurate matching is critical and therefore the kinematic model with a finite state machine approach is greatly favored.

3 The proposed system and its techniques

Figure 1 demonstrates the structure of the proposed system. The user sat on the exercise bicycle. The Kinect sensor was placed 1.5 m away in front of the bicycle. The screen showed the image of an airplane flying in Google Earth's virtual reality. Information about movement detection and flying mode control were also projected on the screen for users to interact with. The airplane's direction, speed, and perspective were completely controlled by the user's body tilt, arm gestures, and paddle movements. Moreover, many flying scenarios were designed in Google Earth. Each special neck or shoulder movement was able to help users stretch the muscles and activate a corresponding scenario simultaneously. The movements mentioned above were all detected using skeleton information and face postures that had been captured by the Kinect sensor. The details are discussed as follows.

A. The design of the aircraft motion detection module in Google Earth

The design in the proposed system could control six aircraft motions in Google Earth scenarios: climb, descend, left banked turn, right banked turn, advance, and retreat. Six corresponding body movements were trunk extension, trunk flexion, trunk left-side bend, trunk right-side bend, pedals pushed-down, and pedals pulled-up. Figure 2 demonstrates the flying scenarios of an airplane's right banked turn, left banked turn, climb, and descent. In

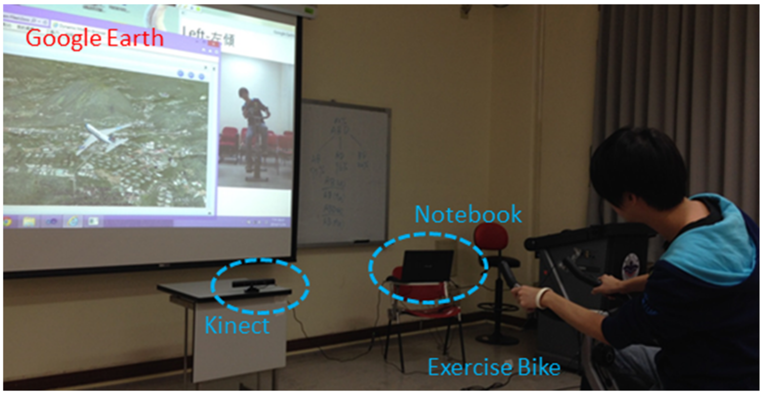


Fig. 1 The structure of the proposed system

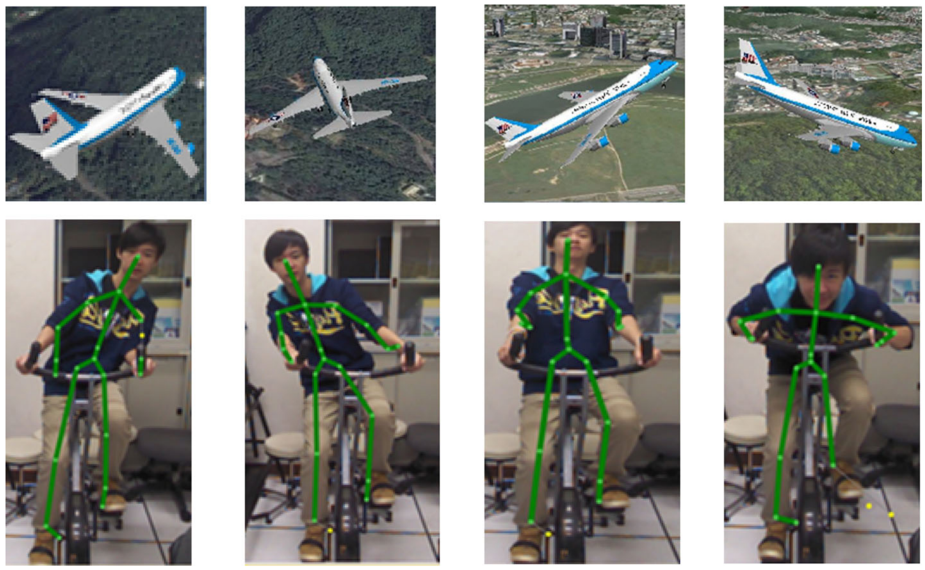


Fig. 2 Scenarios of airplane's right banked turn, left banked turn, climb, and descend

order to let the above movements be successfully analyzed, users were required to participate in system calibration before using it.

- M_0 : The unit design of system calibration

A red horizontal line would appear on the screen, over the user's head, to remind the user to sit upright. Figure 3 illustrates the concept. Sitting upright could prevent more serious neck or shoulder soreness resulting from using the system in a kyphotic position. Moreover, sitting upright also helped collect the initial information of the user. The information could be used as a reference for sebsequent movement analysis. The proposed system used the Z axis information (formed by linking the Head joint point and the Spine_Mid joint point) as a basis for judging whether the user is sitting upright. When the value of Z_{Head} and that of Z_{Spine_Mid} were close, it represented that the user was sitting upright or not. M_0 was used by the proposed system to denote whether the system had detected the user sitting upright or not. $M_0 = 0$ denoted that the user was not sitting upright; M_0 =1 denoted that the user was sitting upright currently. The complete equation for judging whether the user was sitting upright is shown as follows.

$$\begin{cases} \left|\mathrm{Z}_{\mathrm{Head}} - \mathrm{Z}_{Spine\_Mid}\right| < \mathrm{TH}_0, M_0 = 1 \\ \text{Otherwise}, M_0 = 0 \end{cases} \tag{1}$$

where TH_0 denotes the threshold value of sitting upright. When $M_0 = 1$, the system would send a message and automatically record the X, Y, and Z values of the user's current Head joint point in $\mathrm{X}^0_{\mathrm{Head}}$, $\mathrm{Y}^0_{\mathrm{Head}}$, and $\mathrm{Z}^0_{\mathrm{Head}}$ parameters. On the other hand, the X, Y, and Z values of the user's current Spine_Base joint point were recorded in

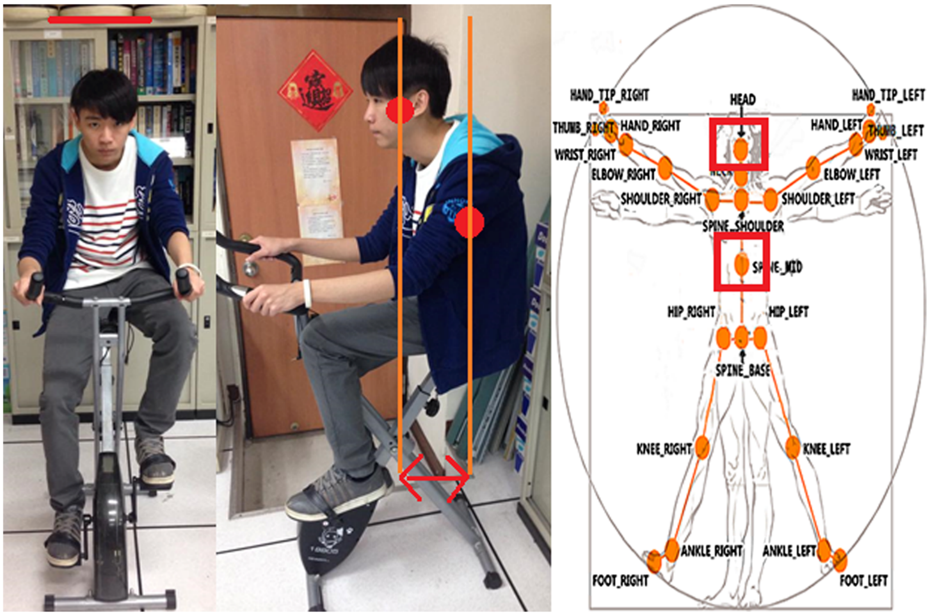


Fig. 3 The illustration of posture calibration unit

$X^0_{\text{Spine\_Base}}$, $Y^0_{\text{Spine\_Base}}$, and $Z^0_{\text{Spine\_Base}}$ parameters. The information was beneficial for judging other movements.

In order to sucessfully detect the push-pedals-down and pull-pedals-up movments, the users were asked to push the pedals of the exercise bike down at least 10 circles during the calibration stage. The system would record the Y value and Z value of Foot_Right during the process and then save the maximum and minimum values of Y in the $Y^{max}_{Foot\_Right}$ and $Y^{min}_{Foot\_Right}$ parameter, respectively.

Moreover, the users were asked to extend the arms overhead during the calibration stage, as shown in Fig. 4. This allowed the system to successfully detect the neck and arm movements. The system would caculate the length of the user's upper arm during this stage. The legnth value could be figured out using the Y value between Elbow_Right and Shoulder_Right, which was $|Y_{\text{Shoulder\_Right}} - Y_{\text{Elbow\_Right}}|$. The result was kept in $D_{\text{Elbow\_Shoulder}}$. In addition, the shoulder width was also recorded, which could be figured out by $D_{Shoulder} = |X_{\text{Shoulder\_Right}} - X_{\text{Shoulder\_Left}}|$.

- M_1 : The unit design of the airplane's right/left banked turn in Google Earth

The user had to right/left side-bend the trunk in order to start the selectedflying mode. The method of detecting the right side-bending of the trunk is demonstrated in Fig. 5. The X and Y values of the Head and the X and Y values of the Spine_Base, which had been detected by Kinect, were used for calculation in this unit. The red line in Fig. 5 was the reference line. This line was drawn by linking $(X^0_{Head} Y^0_{Head})$ and $(X^0_{\text{Spine\_Base}} Y^0_{\text{Spine\_Base}})$, which had been obtained during the posture calibration stage. The blue line was drawn by linking current (X_{Head}, Y_{Head}) and $(X^0_{\text{Spine\_Base}}, Y^0_{\text{Spine\_Base}})$. A correct right banked turn in M_1 was made by right side-bending the trunk. The two lines formed an included angle θ_1 when bending. The value of θ_1 hd to be larger than the threshold value, and the value of X_{Head} was larger than X^0_{Head}.

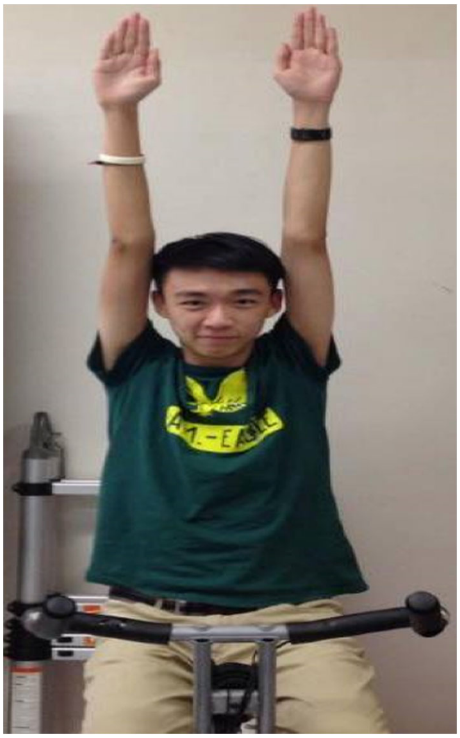


Fig. 4 The illustration of extending the arms overhead during the posture calibration stage

Similarly, when a correct left banked turn was made, the value of the included angle θ_1 had to be larger than the given threshould value, but the value of X_{Head} was smaller than $X^0{}_{Head}$. Therefore, the motion detection in M_1 could be obtained by firguring out Eq. (2).

$$\begin{cases} X_{Head} > X^0_{Head} \text{ and } \theta_1 > \mathrm{TH}_1, S_1 = 1 \\ X_{Head} < X^0_{Head} \text{ and } \theta_1 > \mathrm{TH}_1, S_1 = -1 \\ \text{Otherwise, } S_1 = 0 \end{cases} \tag{2}$$

Where, S_1 = 1 represents right side-bends of the trunk by the user and the system starts the airplane's right banked turn in Google Earth. S_1 = −1 indicates the left side bends of the trunk by the user and the system activates the airplane's left banked turn in Google Earth. S_1 = 0 indicates no corresponding movement. TH_1 is the threshold value the system sets for trunk right/left side-bending. $\theta_1 = \frac{a^2+b^2-c^2}{2ab}$.

$$a^2 = \sqrt{\left(X^0_{Head} - X^0_{\text{Spine\_Base}}\right)^2 + \left(Y^0_{Head} - Y^0_{\text{Spine\_Base}}\right)^2},$$
$$b^2 = \sqrt{\left(X^0_{\text{Spine\_Base}} - X_{\text{Head}}\right)^2 + \left(Y^0_{\text{Spine\_Base}} - Y_{\text{Head}}\right)^2},$$
$$c^2 = \sqrt{\left(X^0_{Head} - X_{\text{Head}}\right)^2 + \left(Y^0_{Head} - Y_{\text{Head}}\right)^2}.$$

- M_2: The unit design of the airplane's climb and descent in Google Earth

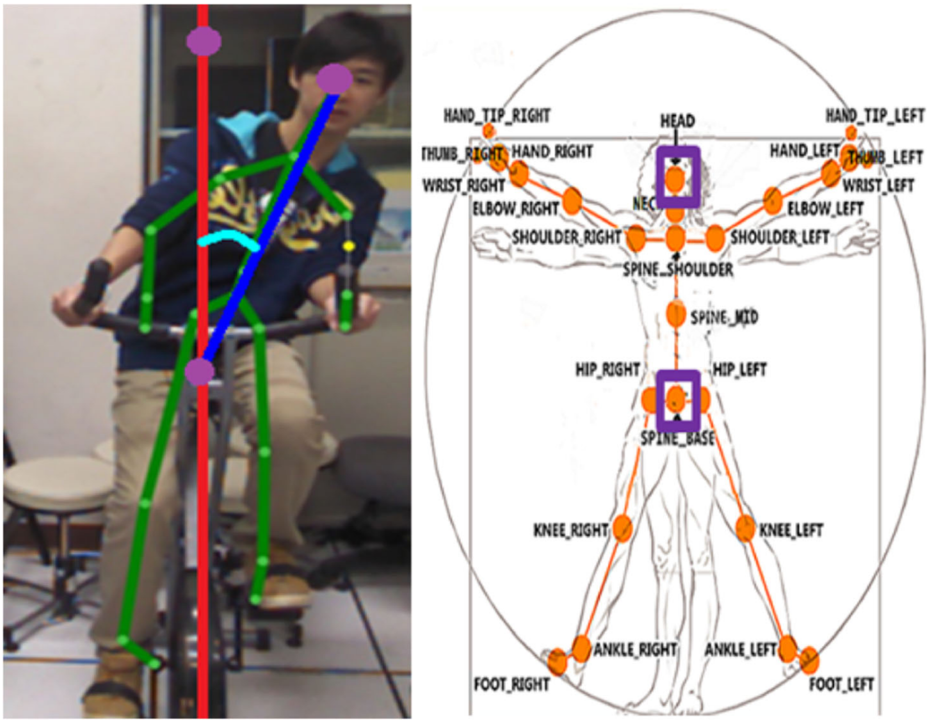


Fig. 5 The illustration of detecting right side-bending the trunk

The user needed to flex or extend the trunk in order to activate this type of flying mode. Figure 6 demonstrates how trunk flextion was detected. The Y and Z values of the Head joint point and the Y and Z values of the Spine_Base joint points detected by Kinect were used for calculation in this unit. The motion detection in M_2 is similar to the motion detection in M_1. The red line in Fig. 6 served as the reference line. This line was formed by linking $(Y^0{}_{Head}, Z^0{}_{Head})$ and $(Y^0{}_{\text{Spine\_Base}}, Z^0{}_{\text{Spine\_Base}})$. The blue line was drawn by linking (Y_{Head}, Z_{Head}) and $(Y^0{}_{\text{Spine\_Base}}, Z^0{}_{\text{Spine\_Base}})$. When the value of Z_{Head} was larger than that of $Z^0{}_{Head}$, and the value of the included angle θ_2 was over the given threshold value, it indicated that the trunk was extended. When the value of Z_{Head} was smaller than the value of $Z^0{}_{Head}$, and the value of the included angle θ_2 was over the given threshold value, it indicated that the trunk was flexed. The motion detection in M_2 could be calculated as in Eq. (3).

$$\begin{cases} Z_{Head} > Z^0_{Head} \text{ and } \theta_2 > \mathrm{TH}_2, S_2 = 1 \\ Z_{Head} > Z^0_{Head} \text{ and } \theta_2 > \mathrm{TH}_2, S_2 = -1 \\ \text{Otherwise, } S_2 = 0 \end{cases} \tag{3}$$

where $S_2 = 1$ represents trunk extension and that the system starts the airplane's climb mode in Google Earth. $S_2 = -1$ reprents trunk flexion and that the system starts the airplane's descent mode in Google Earth. $S_2 = 0$ means there was no

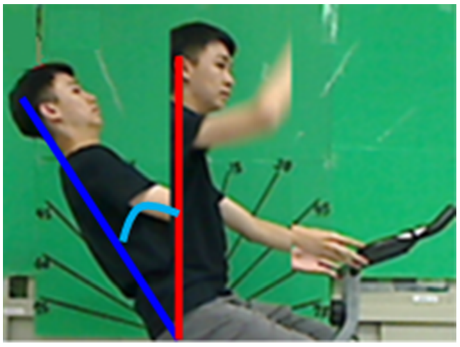


Fig. 6 The illustration of detecting trunk extension

corresponding movement. TH$_2$ was the threshould value set for trunk extension and flexion. $\theta_2 = \frac{d^2+e^2-f^2}{2de}$.

$$d^2 = \sqrt{\left(Y^0_{Head} - Y^0_{\text{Spine\_Base}}\right)^2 + \left(Z^0_{Head} - Z^0_{\text{Spine\_Base}}\right)^2},$$
$$e^2 = \sqrt{\left(Y^0_{\text{Spine\_Base}} - Y_{\text{Head}}\right)^2 + \left(Z^0_{\text{Spine\_Base}} - Z_{\text{Head}}\right)^2},$$
$$f^2 = \sqrt{\left(Y^0_{Head} - Y_{\text{Head}}\right)^2 + \left(Z^0_{Head} - Z_{\text{Head}}\right)^2}.$$

- M_3: The unit design of the airplane's advance and retreat in Google Earth

The users must push the pedals down or pull the pedals up in order to start the advance or retreat of the airplane in Google Earth, respectively. Figure 7 demonstrates how pushing pedals down and pulling pedals up were detected. The Y and Z values of the Foot_Right joint point, detected by Kinect, and the $Y^{max}_{Foot\_Right}$, $Y^{min}_{Foot\_Right}$, $Z^{max}_{Foot\_Right}$, $Z^{min}_{Foot\_Right}$ parameters, obtained during the calibration stage, were used in this unit. Take pushing pedals down as an example. It was done in a regular circular motion. When the foot moved to the back end of the pedal assembly, the Z value of the Foot_Right joint point reached the maximum (which was $Z^{max}_{Foot\_Right}$). The foot then moved to the top of the pedal assembly, when the Y value of the Foot_Right joint point reached the minimum (which was $Y^{min}_{Foot\_Right}$). The foot moved subsequently to the front end of the pedal assembly, when the Z value of the Foot_Right joint point reached the minimum (which was $Z^{min}_{Foot\_Right}$). After that, the foot moved to the bottom of the pedal assembly, when the Y value of the Foot_Right joint point reached the minimum (which was $Y^{max}_{Foot\_Right}$). Finally, the foot moved again to the back end of the pedal assembly. Pulling pedals up had the similar regular pattern. $Z^{max}_{Foot\_Right}$ came first, followed by $Y^{max}_{Foot\_Right}$, $Z^{min}_{Foot\_Right}$, $Y^{min}_{Foot\_Right}$, and finally $Z^{max}_{Foot\_Right}$.

We used this character to design a Finite State Machine (FSM^1) in this study to detect pushing-pedals-down and pulling-pedals-up movements. This FSM had four input signals, which were IS^1_1, IS^1_2, IS^1_3, and IS^1_4. $IS^1_1 = 1$ represents that the system had detected the maximum Z value of the Foot_Right joint point, which was $Z_{Foot\_Right} = Z^{max}_{Foot\_Right}$. $IS^1_2 = 1$ represents that the system detected $Z_{Foot\_Right} = Z^{min}_{Foot\_Right}$. $IS^1_3 = 1$ represents that the system detected $Y_{Foot\_Right} = Y^{max}_{Foot\_Right}$. $IS^1_4 = 1$ represents that the system detected

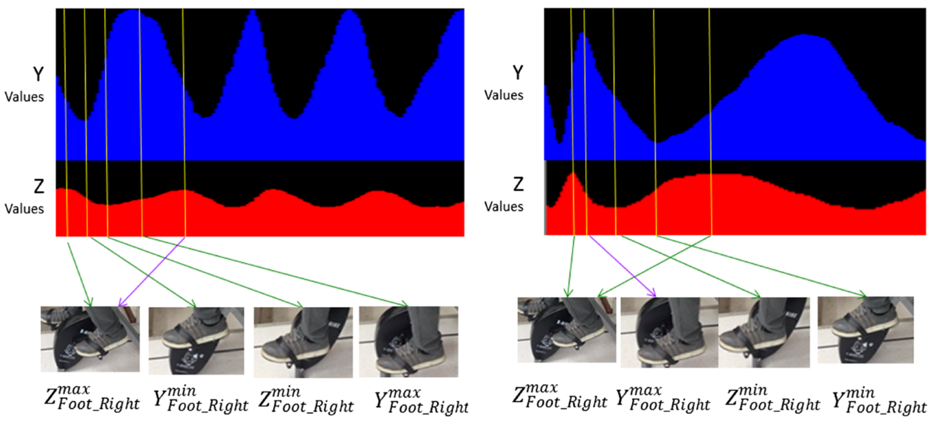


Fig. 7 Illustrations of the pushing-pedal-down and pulling-pedals-up movements

$Y_{Foot\_Right} = Y^{min}_{Foot\_Right}$. This FSM's output signal was OS^1. $OS^1 = 1$ represents that the system detected pushing-pedals-down movements. $OS^1 = -1$ represents that the system detected pulling-pedals-up movements.

This FSM had 8 states. Table 1 shows the State Table of FSM^1. ST^1_0 is the initial state. The FMS would enter ST^1_1 only when the system detected $IS^1_1 = 1$; otherwise, ST^1_0 remained in ST^1_0.

ST^1_1 represents that the system detected that the foot was currently at the back end of the pedal assembly. ST^1_1 would remain in ST^1_1 if ST^1_1 continued to receive the $IS^1_1 = 1$ signal. When ST^1_1 received the $IS^1_4 = 1$ signal, ST^1_1 would enter ST^1_2. This means that the system detected the next state of the pushing-pedals-down movements. If ST^1_1 received $IS^1_3 = 1$ signal, ST^1_1 would enter ST^1_3. This means that the system detected the next state of the pulling-pedals-up movements. If ST^1_1 received the $IS^1_2 = 1$ signal, ST^1_1 would return to ST^1_0. This means that the system was unable to recognize whether the user was pushing the pedals down or pulling the pedals up. Thus, the FMS would go back to the initial state , ST^1_0.

ST^1_2 represents that the system detected part of the pushing-pedals-down movements, that is, the foot moving from the back end to the top of the pedal assembly. The behavior of ST^1_2 was similar to that of ST^1_1. If $IS^1_2 = 1$ continued to be received by the system, then ST^1_2 would enter ST^1_4. ST^1_4 represents that the foot was pushing the pedals down and had already moved to

Table 1 State Table of the proposed FSM^1

	$IS^1_1 = 1$	$IS^1_2 = 1$	$IS^1_3 = 1$	$IS^1_4 = 1$
ST^1_0	ST^1_1	ST^1_0	ST^1_0	ST^1_0
ST^1_1	ST^1_1	ST^1_0	ST^1_3	ST^1_2
ST^1_2	ST^1_0	ST^1_4	ST^1_0	ST^1_2
ST^1_3	ST^1_0	ST^1_5	ST^1_3	ST^1_0
ST^1_4	ST^1_0	ST^1_4	ST^1_6	ST^1_0
ST^1_5	ST^1_0	ST^1_5	ST^1_0	ST^1_7
ST^1_6	ST^1_1	ST^1_0	ST^1_6	ST^1_0
ST^1_7	ST^1_1	ST^1_0	ST^1_0	ST^1_7

the front end of the pedal assembly. At this moment, if the system continued receiving the $IS_3^1 = 1$ signal, then ST_4^1 would enter ST_6^1. ST_6^1 represents that the foot was pushing the pedals down and had already moved to the bottom of the pedal assembly. If the system continued to receive the $IS_1^1 = 1$ signal, ST_6^1 would go back to ST_1^1 and output $OS^1 = 1$. This means that the system detected the pedal had been pushed down for one circle.

ST_3^1 represents that the system detected part of the pulling-pedals-up movement. In the course of a correct pulling-pedals-up movement, ST_3^1 would change to ST_5^1, then to ST_7^1, and then return to ST_1^1 and output the $OS^1 = -1$ signal.

B. The design of the neck and shoulder movement detection module

In this study, we designed multiple flying scenarios in Google Earth so as to elevate users' motivation to use the exercise bike and also to improve the drawback that exercise bikes only train the lower part of the body. Each special neck or shoulder movement was able to activate a corresponding flying scenario. These special neck and shoulder movements were all designed according to Pilates moves. The corresponding flying mode was activated after the user had done the specially designed neck or shoulder movement.

- M_4: The unit design of the shoulder rotation exercise detection

Figure 8 demonstrates a complete shoulder rotating exercise. This exercise helps stretch the muscles in the shoulders by rotating the shoulder blades. The exercise was designed on the basis of Pilates moves [16]. In this exercise, the first part was to flex the elbows, put the hands on the shoulders, and then extend the arms to the sides to bring the elbows parallel to the chest, as shown in Fig. 8a. In the second part, the elbows were moved inward until they were right in front of the shoulders. The distance between two elbows was close to the distance between two shoulders, as shown in Fig. 8b. In the third part of this exercise, the elbows were moved toward the ears and temples with both hands staying on the shoulders until the elbows were at the sides of the head, as shown in Fig. 8c. In the final part, the elbows were moved toward the back and the shoulder blades were rotated. The elbows were parallel to the shoulders at first and then returned to the initial position from the first part of the exercise.

The shoulder rotation exercise has another special feature. Both hands must stay on the shoulders. Because the X and Y distances between Hand_Right and Hand_Left joint points were close to the X and Y distances between the Shoulder_Right and

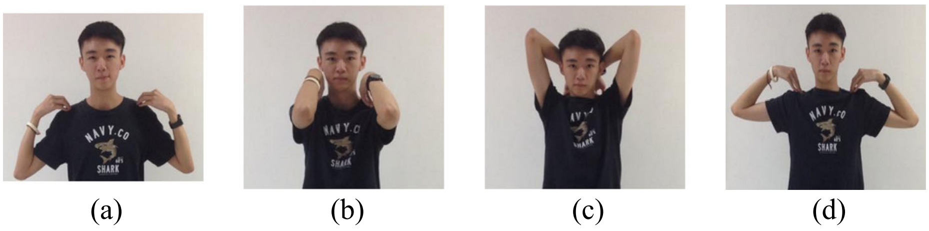


Fig. 8 The illustration of the shoulder rotation exercise

Shoulder_Left joint points. This feature was easy to detect. Detection was carried out as follows.

$$\begin{aligned} &\mathrm{D}_{\text{Hand\_Shoulder\_Left}} \cong \quad 0 \quad \text{and} \quad \mathrm{D}_{\text{Hand\_Shoulder\_Right}} \cong \quad 0 \quad , \\ &\mathrm{D}_{\text{Hand\_Shoulder\_Left}} = \sqrt{(X_{\text{Hand\_Left}} - X_{\text{Shoulder\_Left}})^2 + (Y_{\text{Hand\_Left}} - Y_{\text{Shoulder\_Left}})^2}, \\ &\mathrm{D}_{\text{Hand\_Shoulder\_Right}} = \sqrt{\left(X_{\text{Hand\_Right}} - X_{\text{Shoulder\_Right}}\right)^2 + \left(Y_{\text{Hand\_Right}} - Y_{\text{Shoulder\_Right}}\right)^2} \circ \end{aligned}$$

The shoulder rotation exercise constituted a series of movements, similar to the pushing-pedals-down exercise. In this study, we used the four movements shown Fig. 8a, b, c and d as checkpoints. As long as the user did the four movements in the correct order, the system would determine that the user had finished one shoulder rotation exercise. The four movements could be detected using the X, Y, and Z information of the Shoulder_Right, Shoulder_Left, Elbow_Right, and Elbow_ Left joint points. Another Finite State Machine (FSM^2) was designed in this study to detect the shoulder rotation exercise. There were four input signals, IS_1^2, IS_2^2, IS_3^2, and IS_4^2. $IS_1^2 = 1$、 $IS_2^2 = 1$, $IS_3^2 = 1$, and $IS_4^2 = 1$ represent that the system detected the movements in Fig. 8a, b, c and d, respectively.

In Fig. 8a, moreover, the arms were relaxed and were lower than the shoulders. Thus, the Y value of the Elbow_Right joint point was larger than the Y value of the Shoulder_Right joint point. Similarly, the Y value of Elbow_Left joint point would be larger than the Y value of Shoulder_Left joint point. The difference between the Y value of Elbow_Right and the Y value of Shoulder_Right was about half the distance of the user's forearm. This feature could be calculated by the equation below.

$$\begin{cases} Y_{\text{Elbow\_Right}} - Y_{\text{Shoulder\_Right}} \geq \dfrac{D_{\text{Elbow\_Shoulder}}}{2} \\ Y_{\text{Elbow\_Left}} - Y_{\text{Shoulder\_Left}} \geq \dfrac{D_{\text{Elbow\_Shoulder}}}{2} \end{cases} \tag{4}$$

$D_{\text{Elbow\_Shoulder}}$ in the equation was a parameter figured out during the calibration stage. If all four conditions were met, the system would generate the $IS_1^2 = 1$ signal, which means that the movement in Fig. 8a was detected. Otherwise, $IS_1^2 = 0$ would remain.

In Fig. 8b, the elbows were in front of the shoulders. The Shoulder_Right joint point would therefore be blocked by the Elbow_Right joint point. The X, Y, and Z values of Shoulder_Right were unknown in this state. Also, the Shoulder_Left joint point would be blocked by the Elbow_Left joint point. The X, Y, and Z values of Shoulder_Left were also unknown. The distance between the elbows was required to be close to that of the distance between the shoulders in Fig. 8b. Such a requirement resulted in $|X_{\text{Elbow\_Right}} - X_{\text{Elbow\_Left}}| \cong D_{Shoulder}$, where $D_{Shoulder}$ was a parameter figured out during the calibration stage. If the above conditions were met, the system would generate the $IS_2^2 = 1$ signal, meaning that the movement in Fig. 8b had been detected; otherwise, $IS_2^2 = 0$ would remain.

In Fig. 8c, the elbows were moved toward the head. In this situation, the Y value of the Elbow_Right joint point would be smaller than the Y value of the Shoulder_Right joint point. When the elbows reached a certain height, the difference between the two Y values was almost half the distance of the length of the user's forearm. It was the same for the Elbow_Left and Shoulder_Left joint points. The movement in Fig. 8c was figured out by the following equation.

$$\begin{cases} Y_{\text{Shoulder\_Right}} - Y_{\text{Elbow\_Right}} \geq \dfrac{D_{\text{Elbow\_Shoulder}}}{2} \\ Y_{\text{Shoulder\_Left}} - Y_{\text{Elbow\_Left}} \geq \dfrac{D_{\text{Elbow\_Shoulder}}}{2} \end{cases} \tag{5}$$

$D_{\text{Elbow\_Shoulder}}$ in the equation represents the parameter obtained during the calibration stage. If the above conditions were met, the system would generate $IS_3^2 = 1$, which means that the movement in Fig. 8c was detected. Otherwise, $IS_3^2 = 0$ would remain.

The biggest feature in Fig. 8d was that the elbows were parallel to the two shoulders. At this moment, the Z value of the Shoulder_Right joint point was close to that of the Elbow_Right joint point. The feature can be detected using Eq. (3). In addition, the X values of the Shoulder_Right and Elbow_Right joint points should be close to the length of the user's forearm. Thus, $|X_{\text{Shoulder\_Right}} - X_{\text{Elbow\_Right}}| \cong D_{\text{Elbow\_Shoulder}}$. It was the same for the Shoulder_Left and Elbow_Left joint points, and therefore, $|X_{\text{Shoulder\_Left}} - X_{\text{Elbow\_Left}}| \cong D_{\text{Elbow\_Shoulder}}$. If the above conditions were met, the $IS_4^2 = 1$ signal would be generated, meaning that the movement in Fig. 8d was detected. Otherwise $IS_4^2 = 0$ would remain.

There were 5 states in FSM^2. ST_0^2 was the initial state of FSM^2. ST_0^2 would change to ST_1^2 when it received $IS_1^2 = 1$ signal. This means that the system detected the movement in Fig. 8a. ST_1^2 would change to ST_2^2 when it received $IS_2^2 = 1$ signal. This means that the system detected the movement in Fig. 8b. ST_2^2 would change to ST_3^2 when it received the $IS_3^2 = 1$ signal, which means that the system detected the movement in Fig. 8c. ST_3^2 would change to ST_4^2 when it received the $IS_4^2 = 1$ signal, meaning that the system detected the movement in Fig. 8d. ST_4^2 would change to ST_1^2 when it received the $IS_1^2 = 1$ signal. The output signal OS^2 was set as 1 at this moment. This means that a complete shoulder rotation exercise was detected. Table 2 demonstrates the State Table of FSM^2.

- M_5: The unit design of the neck rotation exercise detection

A complete neck rotation exercise is shown in Fig. 9. The muscles around the neck were stretched by rotating the neck. The movements in this exercise were designed according to Pilates moves. The user sat upright and faced forward at the beginning of the exercise. Right after that, the user started bending the neck to the right side until the muscles on the left side of the neck felt tense, as shown in Fig. 9a. The user then slowly moved the head downward and

Table 2 State Table of the proposed FSM^2

	$IS_1^2 = 1$	$IS_2^2 = 1$	$IS_3^2 = 1$	$IS_4^2 = 1$
ST_0^2	ST_1^2	ST_0^2	ST_0^2	ST_0^2
ST_1^2	ST_0^2	ST_2^2	ST_0^2	ST_0^2
ST_2^2	ST_0^2	ST_0^2	ST_3^2	ST_0^2
ST_3^2	ST_0^2	ST_0^2	ST_0^2	ST_4^2
ST_4^2	ST_1^2	ST_0^2	ST_0^2	ST_0^2

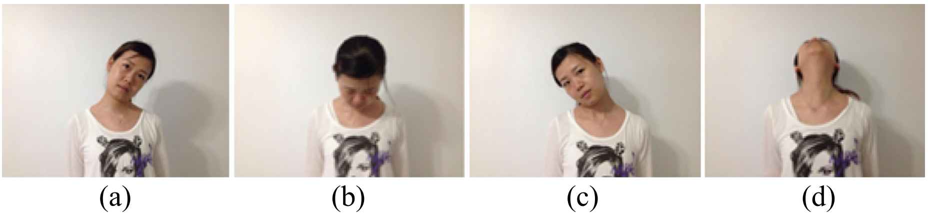


Fig. 9 Illustration of neck-rotating exercise

forward to the front, as shown in Fig. 9b. After that, the head was slowly moved to the left side and backward until it was over the left shoulder, as shown in Fig. 9c. The user then slowly moved the head backward and toward the right side until the head was at the back, as shown in Fig. 9d. Finally, the head was moved forward slowly, and the neck was side-bended to the right side until the head was moved back to the right shoulder, as shown in Fig. 9a.

A special feature about the neck rotation exercise was that the user had to keep sitting upright during the whole process with the trunk unmoved. This feature could be detected using the X and Y values of the Shoulder_Right and Shoulder_Left joint points. When the difference between the X value and the Y value of the Shoulder_Right continuously remained too big, the system would consider that the trunk had moved. Similarly, when the difference between the X value and the Y value of the Shoulder_Left was continuously too big, the system would also consider that the trunk had moved.

The neck rotation exercise was formed by a succession of movements. The movements in Fig. 9a, b, c and d were used as checkpoints in this study. As long as the user finished the movements in Fig. 9a, b, c and d in the correct order, the system would consider that a complete neck rotation exercise had been done by the user. Another Finite State Machine (FSM^3) was designed in this study to detect the neck rotation exercise. This exercise could also be detected using the face swinging angle information provided by Kinect's face tracking function, as shown in Fig. 10, where, pitch, yaw, and roll were the face rotation angles relative to the X, Y, and Z axes, respectively.

This FMS included four input signals as well: IS_1^3, IS_2^3, IS_3^3, and ST_4^3. $IS_1^3=1$, $IS_2^3=1$, $IS_3^3=1$, and $IS_4^3=1$ represent that the system detected the movements in Fig. 9a, b, c and d, respectively. FSM^3 had 5 states, too. The state transition mechanism of FSM^3 was similar to that of FSM^2. ST_0^3 was the initial state of FSM^3. When ST_0^3 received the $IS_1^3=1$ signal, it would transition to ST_1^3. When ST_1^3 received the $IS_2^3=1$ signal, it would transition to ST_2^3. When ST_2^3 received the $IS_3^3=1$ signal, it would transition to ST_3^3. When ST_3^3 received the $IS_4^3=1$ signal, it would transition to ST_4^3. When ST_4^3 received the $IS_1^3=1$ signal, it would transition back to ST_1^3. OS^3 was set to 1 to indicate that the system had detected one complete neck rotation exercise. During the course of the neck rotation exercise, the key point was to rotate the neck slowly. A timer mechanism was designed in this study to control the whole course of the neck rotation exercise. The time spent in each state had to be more than the timer's setting so as to transfer to the next state. Otherwise, the state transition would fail and return to the initial state. Table 3 demonstrates the State Table of the FSM.

Movements in Fig. 9a, b, c and d could be successfully detected with the pitch and yaw information provided by Kinect. Take Fig. 9a for instance. The neck was rotated to the right side, and therefore, the pitch angle was near 0. The yaw angle was larger than the threshold value TH_{yaw}. When the neck was rotated to the position in Fig. 9b, the yaw angle was near 0,

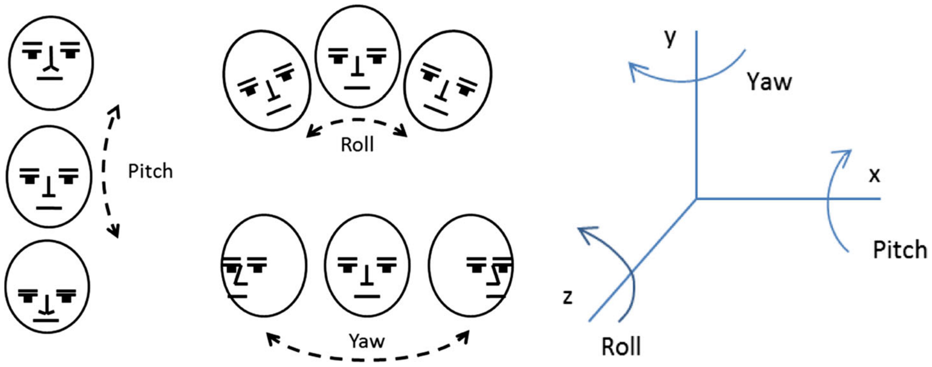


Fig. 10 The concept illustration of pitch, yaw, and raw. (Source of information: Microsoft)

and the pitch angle was larger than a threshold value TH_{pitch}. When the neck was rotated to the position in Fig. 9c, the pitch angle was again near 0, but the yaw angle was smaller than the threshold value TH_{yaw}. When the neck was rotated to the back, as shown in Fig. 9d, the yaw angle was close to 0. The neck extension angle was smaller than the neck flexion angle, according to human body structure. Thus, the pitch angle was smaller than the threshold value $-\frac{TH_{pitch}}{2}$. Equation 6 summarizes the detection method described above.

$$\begin{cases} \theta_{pitch} \cong 0 \; and \; \theta_{yaw} > TH_{yaw}, \; IS_1^3 = 1 \\ \theta_{pitch} > TH_{pitch} \, and \; \theta_{yaw} \cong 0, \; IS_2^3 = 1 \\ \theta_{pitch} \cong 0 \, and \, \theta_{yaw} < -TH_{yaw},, \; IS_3^3 = 1 \\ \theta_{pitch} < -\dfrac{TH_{pitch}}{2} \; and \, \theta_{yaw} \cong 0, \; IS_4^3 = 1 \end{cases} \tag{6}$$

where θ_{pitch} and θ_{yaw} were the pitch and yaw angles provided by Kinect at that time. TH_{yaw} and TH_{pitch} were both set at 25°.

C. The design of the airplane flying mode in Google Earth

The structure of the visualized aircraft dynamic system, which combined the application of Google Earth with the proposed Exergaming system, is shown in Fig. 11. The visualized element of the airplane's dynamic motion in Google Earth's virtual reality was designed by combining JavaScript and Google Earth API in this study.

In Fig. 12, OXYZ represents the ECEF (Earth-Centered, Earth-Fixed) coordinate system which used the center of the earth as its origin. A-XYZ represents the ENU (East, North, UP)

Table 3 State Table of FSM^3

	$IS_1^3 = 1$ and Timer=1	$IS_2^3 = 1$ and Timer=1	$IS_3^3 = 1$ and Timer=1	$IS_4^3 = 1$ and Timer=1
ST_0^3	ST_1^3	ST_0^3	ST_0^3	ST_0^3
ST_1^3	ST_0^3	ST_2^3	ST_0^3	ST_0^3
ST_2^3	ST_0^3	ST_0^3	ST_3^3	ST_0^3
ST_3^3	ST_0^3	ST_0^3	ST_0^3	ST_4^3
ST_4^3	ST_1^3	ST_0^3	ST_0^3	ST_0^3

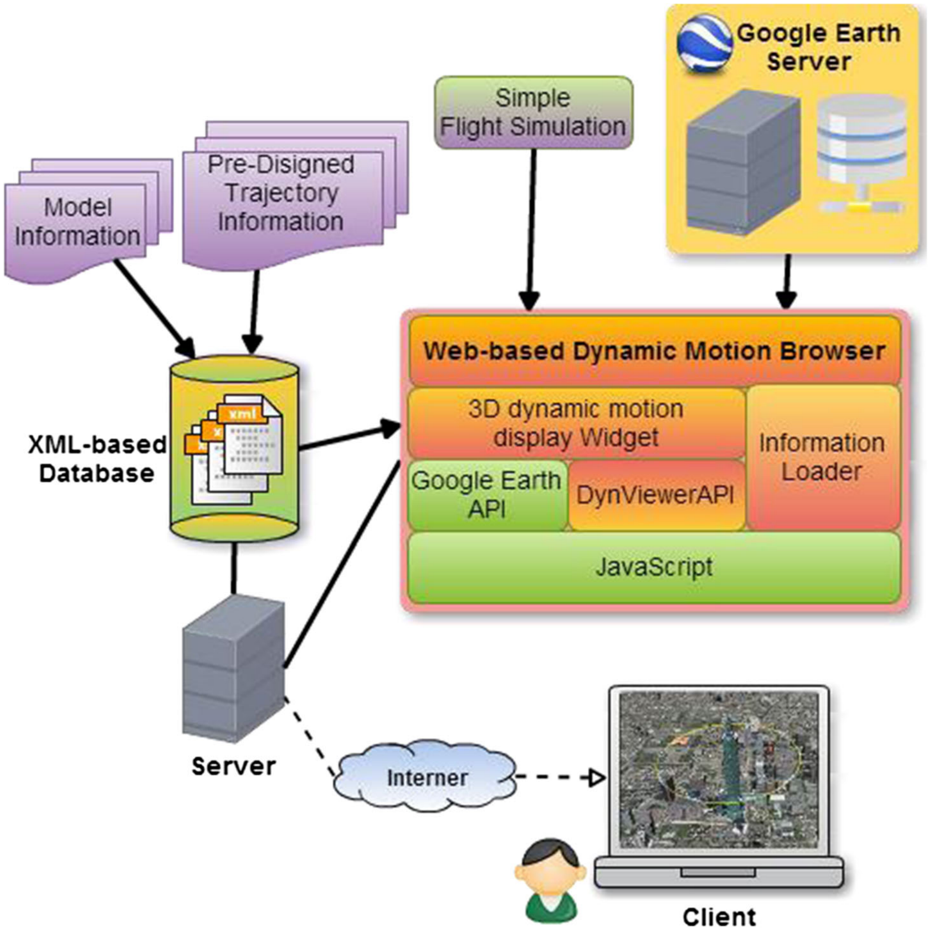


Fig. 11 The structure of the visualized aircraft dynamic system in Google Earth

coordinate system, which is fixed to the surface of the earth. The models of position vector and altitude angle vector are described as follows:

D. Model of position vector

In Fig. 12, suppose x, y and z are the Cartesian coordinates of point A with respect to ECEF system, then the position vector $\vec{OA}_{ECEF}$ is denoted as $\vec{OA}_{ECEF} = \vec{Apos}_{ECEF}$ = [x, y, z]

E. Model of attitude angle vector

In Fig. 12, suppose the orientation of Cartesian coordinate system, B-XYZ, can be obtained by performing a sequence of rotations on A-XYZ Cartesian coordinate system in the following order:

(1) Rotation about the z axis with an angle denoted as heading.

A positive rotation is clockwise around the z axis and specified in degrees from 0 to 360.

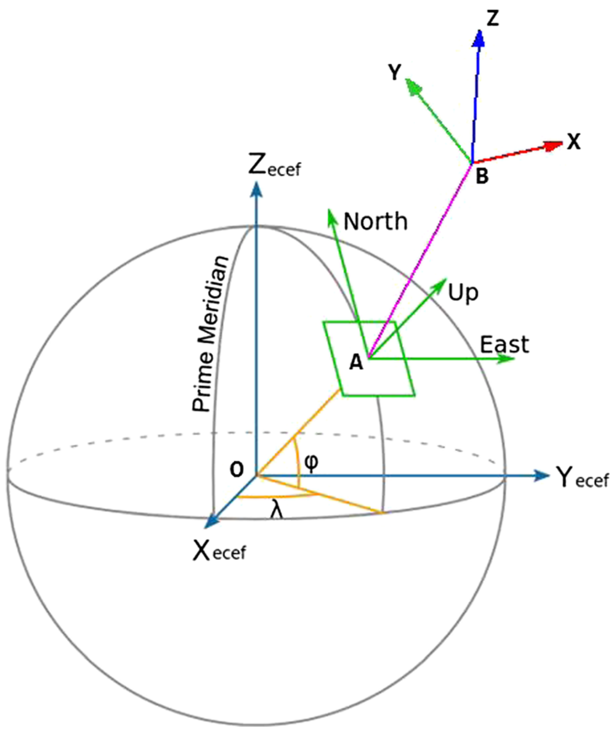


Fig. 12 The coordinate system used in flight simulation

(2) Rotation about the x axis with an angle denoted as tile.

A positive rotation is clockwise around the x axis and specified in degrees from 0 to 180.

(3) Rotation about the y axis with an angle denoted as roll.

A positive rotation is clockwise around the y axis and specified in degrees from 0 to 180.

Then the attitude angle vector of the above rotation transformation is denoted as $[\text{heading, tilt, roll}]_{B-XYZ}$

If point B denotes the center of reference of the moving object B and if B-XYZ denotes the coordinate system fixed to the moving object B, then the position vector $\overrightarrow{Bpos}_{ECEF}$ of the object B relative to the coordinate system ECEF could be figured out on the basis of the following coordinate transformation.

$$\overrightarrow{Bpos}_{ECEF} = Vector\ .\ add\left(\overrightarrow{Apos}_{ECEF}\quad,\ Matrix.transform\left(orientationM_{A-XYZ},\ \overrightarrow{Bpos}_{A-XYZ}\right)\right)$$

Where $\overrightarrow{Apos}_{ECEF}$ denotes the position vector of the A-XYZ coordinate system's origin relative to the ECEF coordinate system. $\overrightarrow{Bpos}_{A-XYZ}$ denotes the position vector of the moving object B relative to A-XYZ coordinate system. $orientationM_{A-XYZ}$ denotes the coordinate transformation matrix which rotated O-XYZ to A-XYZ in the order of Z, Y, and X axes.

In the above formula, $Vector.add()$ indicates the addition of two vectors; $Matrix.transform()$ defines the multiplication of the matrix and vector. The position vector, $[\text{lon, lat, alt}]_B$, of the moving object B can be figured out by the following equation expressed using the longitude and latitude coordinate system.

$$[\text{lon, lat, alt}]_B = Vector.CartesianToLatLonAlt\left(\overrightarrow{Bpos_{ECEF}}\right)$$

Where, $Vector.CartesianToLatLonAlt()$ indicates the transformation from the Cartesian coordinate system to the longitude and latitude coordinate system. If A-XYZ completely overlapped the B-XYZ coordinate system only when the angles of heading, tilt, and roll were rotated in the order of z-y-x, then the attitude angle vector $[\text{heading, tilt, roll}]_{B-XYZ}$ could be calculated as:

$$\begin{aligned}&[\text{heading, tilt, roll}]_{B-XYZ} = \\ &Matrix.orientationMatrixToHeadingTiltRoll(orientationM_{B-XYZ})\end{aligned}$$

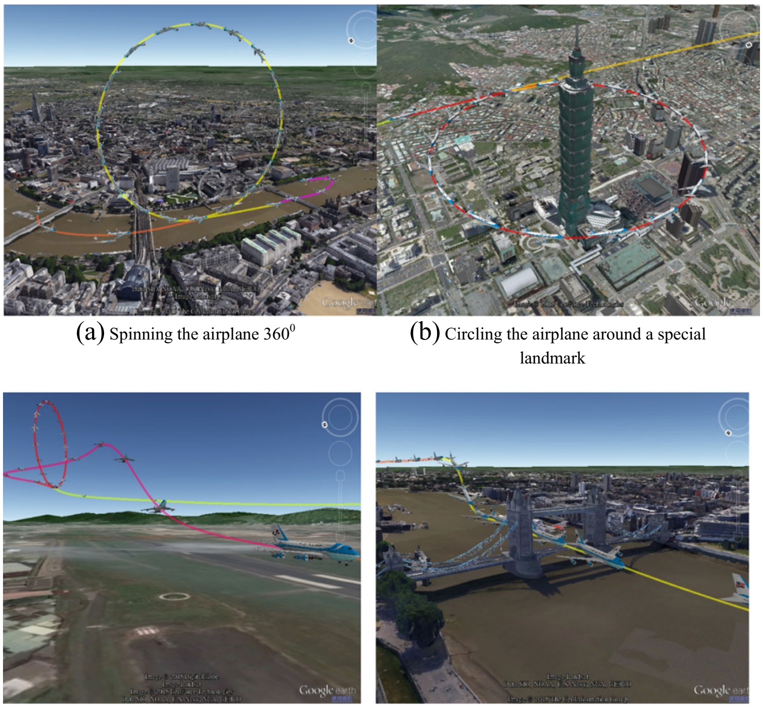
$\overrightarrow{Bpos_{ECEF}}$ and $orientationM_{B-XYZ}$ described above were controlled by parameters such as the flying speed, turning speed, and climbing speed, etc., of the airplane's motion. Once the position vector, $[\text{lon, lat, alt}]_B$, and the attitude angle vector, $[\text{heading, tilt, roll}]_{B-XYZ}$, of the moving object were figured out, the position and attitude of the moving object could be expressed by using the method provided in Google Earth API, which is shown as follows.

$$\begin{aligned}&\text{kmlModel.getOrientation().set(heading, tilt, roll);}\\&\text{kmlModel.getLocation().setLatLngAlt(lat, lon, alt);}\end{aligned}$$

where KmlModel denotes the moving object established in the Google Earth scenario. In addition, with the following formula, the FOV control element in the Google Earth scenario was able to track the moving object. As a result, the user could experience the effect of travelling through virtual reality.

$$\text{GEViewControlObject.setAbstractView([lat, lon, alt]);}$$

With the visualized 3-D dynamic motion element designed in the proposed system, the user could control the airplane's trajectory using basic flying modes such as climb, descend, left banked turn, right banked turn, accelerate, decelerate, etc. A trajectory database of various corresponding flying scenarios was established, allowing the user to trigger some kind of flying scenario after doing the assigned neck or shoulder exercise. The scenarios included vertically spinning the airplane, circling the airplane around some special landmark (for example, Taipei 101), or flying the airplane along the River Thames. By controlling the airplane's flying



(a) Spinning the airplane 360^0

(b) Circling the airplane around a special landmark

(c) Low-flying

(d) Flying through Tower Bridge (traveling along River Thames)

Fig. 13 The airplane's flying scenarios

mode, the user could travel in Google Earth's virtual world. Figure 13 demonstrates the flying scenarios.

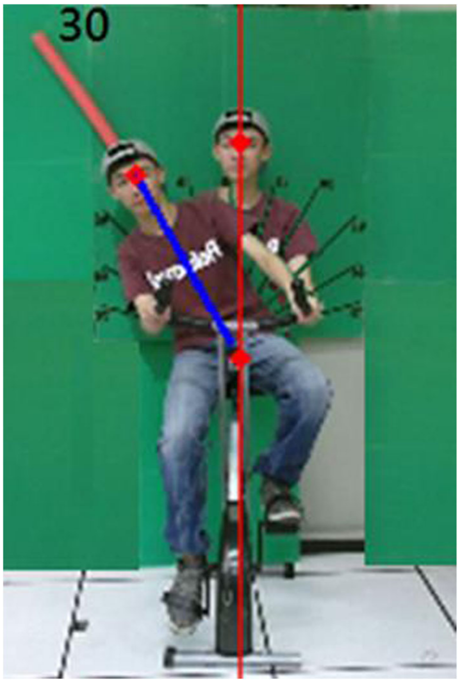
4 Experimental results

The system used a notebook as platform. The Kinect SDK library launched by Microsoft was used in this study to connect Kinect to the notebook. Thirty subjects participated in this experiment. The participants all sat on the exercise bike to execute the experiment. The Kinect device was set in front of the subject, 1.5 m away. The performance of the motion detection module which controlled the flying mode in Google Earth and the movement detection performance during neck and shoulder exercises proposed in this system are discussed as follows.

F. Analysis of the performance of the M_1 module

First, the performance in detecting right side-bending and left side-bending of the trunk by the M_1 module are discussed below. The subject had to side-bend the trunk to an assigned angle. The angles in the experiment were 15° for right side-bending, 30° for right side-

Fig. 14 The concept illustration of detecting the body's side-bending range



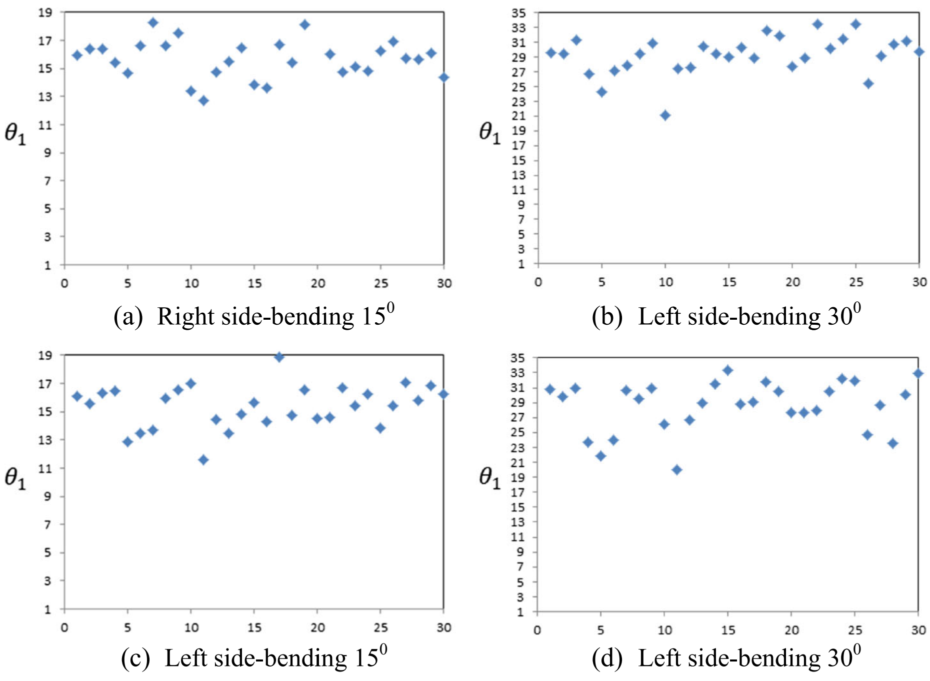


Fig. 15 The diagrams of the experimental results of detecting the trunk's left/right side-bending angles

bending, 15° for left side-bending, and 30° for left side-bending. Figure 14 shows the case where subjects side-bended the trunk 30° to the left. The θ_1 measured by the M_1 module are illustrated in Fig. 15. The average errors of θ_1 in the 15° right-side-bending test and the 15° left-side-bending test are 1.31° and 1.28°, respectively, as shown by the experimental results. The error becomes larger as the bending angle increases. In the tests of side-bending the trunk 30° to the right and the left, the average errors of θ_1 are 1.97° and 1.63°, respectively. The reason is that when the side-bending magnitude increases, the subject's pelvis, sitting on the exercise bike, inclines more. Hence, the error of the M_1 module increases as well. However, the average error remains below 3°, which is an acceptable range.

Next, the performance of controlling the airplane's right banked turn and left banked turn by M_1 in Google Earth is discussed as follows. A user experience questionnaire was designed and given to the 30 participants of this experiment. Table 4 demonstrates the satisfaction degree of the performance, which is fairly high. About 80 % of the users were strongly satisfied with the performance.

G. Analysis of the performance of the M_2 module

The subject had to extend or flex the trunk to assigned angles in order to verify the detection precision of the M_2 module. The assigned angles were 15° for trunk-flexion, 30° for trunk-flexion, 15° for trunk-extension, and 30° for trunk-extension. The θ_2 measured by the M_2 module is demonstrated in Fig. 16. The experimental result is similar to that of the trunk side-bending experiment. The average error is below 25°. The error increases as the side-bending magnitude increases. The average error of trunk-extension is larger than that of trunk-flexion, as shown in the data. A possible reason may be that the joint point of the head is farther away when performing trunk-extension, which results in larger error between the vectors from the head joint point to the pelvis joint point.

Table 5 shows the result of the satisfaction degree questionnaire with regard to the user experience of controlling the airplane's climbing and descending in Google Earth. The satisfaction degree is fairly high, as shown in the table.

H. Analysis of the performance of the M_3 module

First, the detection performance of the M_3 module for the pushing-pedals-down and pulling-pedals-up exercises is discussed below. The subject had to push the pedals of the exercise bike down or pull the pedals up as commanded. In order to analyze how the proposed algorithm performed at different speeds, the subject had to push down or

Table 4 Satisfaction degree of the performance of the airplane's right and left-banked turn in Google Earth

	Strongly satisfied	Satisfied	Neutral	Unsatisfied	Strongly Unsatisfied
Right/Left banked-turn	81 %	19 %	0 %	0 %	0 %

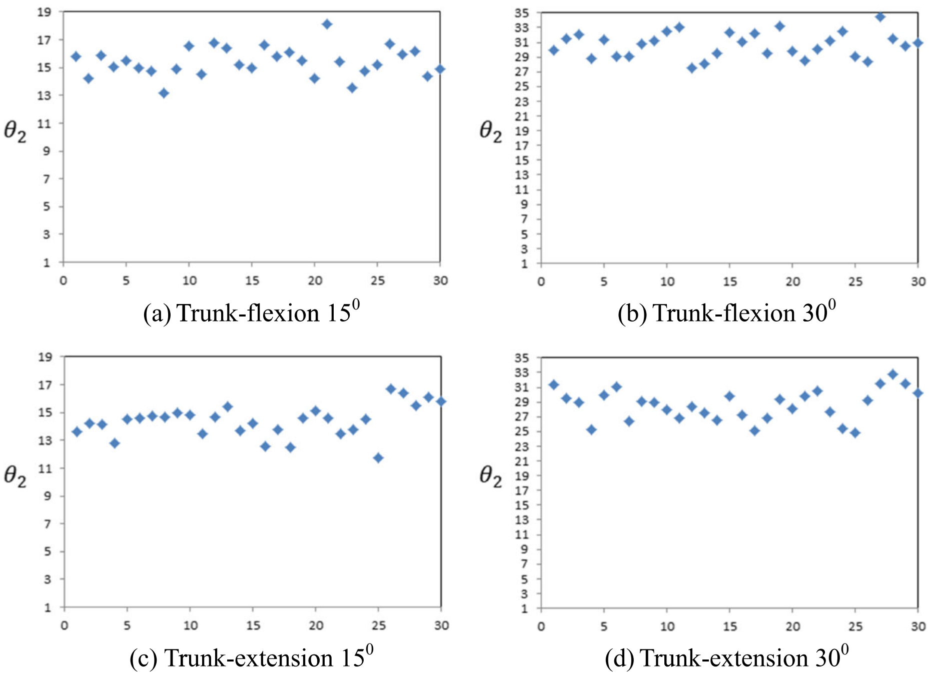
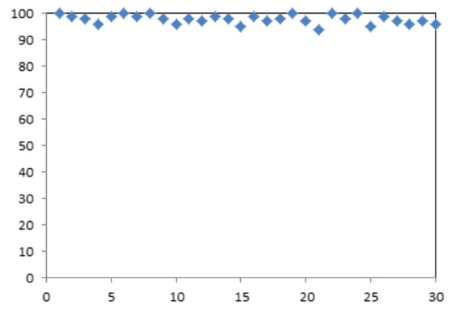


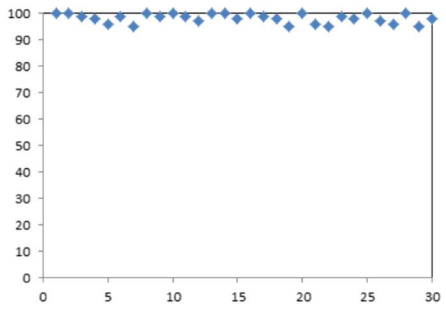
Fig. 16 The diagram of the experimental result of detecting trunk-flexion and extension

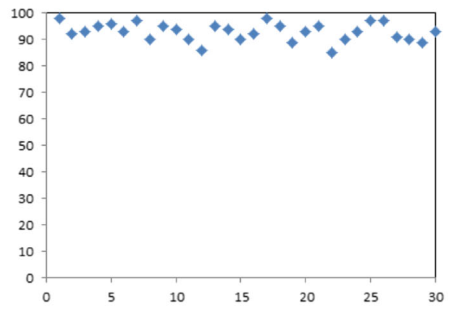
pull up the pedals for 100 circles at the same speed within 150 s. This state was defined as Low-Speed Case. The subject also had to push down or pull up the pedals for 100 circles at the same speed within 100 s. This state was defined as Middle-Speed Case. As for the High-Speed Case, the subject had to push down or pull up the pedals for 100 circles at the same speed within 60 s. Figure 17 illustrates how many circles that M_3 measured in this experiment. M_3 had better performance in the Low-Speed Case, as shown in the experimental results. The average number of circles measured in cases of pushing-pedals-down and pulling-pedals-up were 98.5 and 98.6, respectively. The M_3 module also performed well in the Middle-Speed Case. The average number of circles measured by M_3 in this case was 95 and 94.3. However, the M_3 module did not perform well in the High-Speed Case. The average number of circles measured in this case was 59.2 and 53.1. The reason was that in the High-Speed Case, the feet changed too fast, and therefore, Kinect was unable to precisely capture the Y values and Z values of each time point. Figure 18 shows the changes of

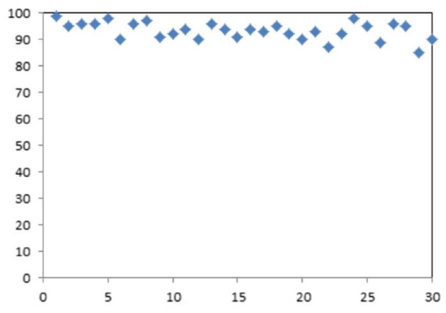
Table 5 Satisfaction degree with regard to the user experience of controlling the airplane's climbing and descending in Google Earth

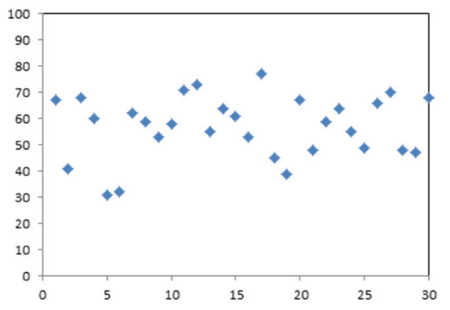
	Strongly agree	Agree	Neutral	Strongly disagree	Disagree
The climbing and descending of the airplane	80 %	11 %	9 %	0 %	0 %


(a) Pushing pedals down in Low-Speed Case


(b) Pulling pedals up in Low-Speed Case


(c) Pushing pedals down in Middle-Speed Case


(d) Pulling pedals up in Middle-Speed Case


(e) Pushing pedals down in High-Speed Case

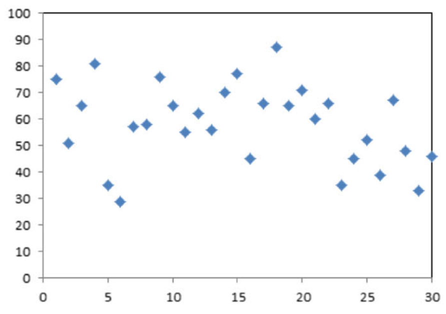
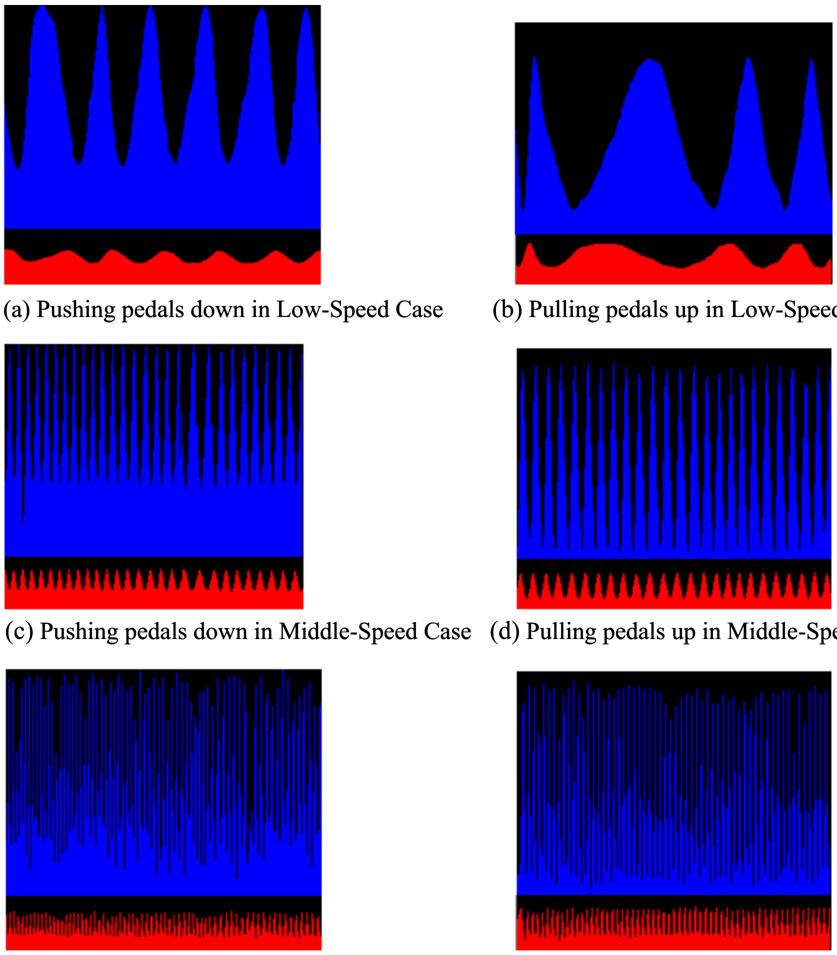

(f) Pulling pedals up in High-Speed Case

Fig. 17 The diagram of the number of circles figured out by M_3 module in the experiment

Y values and Z values in the Low-Speed Case, Middle-Speed Case, and High-Speed Case. The Y values and Z values in both Low and Middle-Speed Cases form clear and complete curves. Yet, in the High-Speed Case, the curve is broken, as result of the decision error caused by Kinect's inability to detect the Y values and Z values of the feet in real time.

Table 6 demonstrates the satisfaction degree results on the questionnaire with regard to the user experience of controlling the airplanes advance and retreat. 84 % of the users were satisfied, as shown in the results. However, worse than the M_1 and M_2 modules, 3 % of the


(a) Pushing pedals down in Low-Speed Case (b) Pulling pedals up in Low-Speed Case

(c) Pushing pedals down in Middle-Speed Case (d) Pulling pedals up in Middle-Speed Case

(e)Pushing pedals down in High-Speed Case (f) Pulling pedals up in High-Speed Case

Fig. 18 Changes of Y values and Z values in Low-Speed, Middle-Speed, and High-Speed Cases

users were unsatisfied. The reason relates to the bad performance of the M_3 module in the High-Speed Case.

I. Analysis of the performance of the M_4 module

Table 6 Satisfaction degree with regard to the user experience of controlling the airplane's advance and retreat in Google Earth

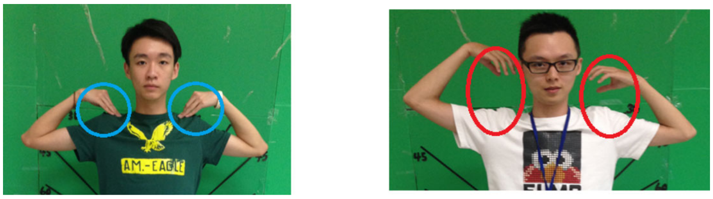
	Strongly satisfied	Satisfied	Neutral	Unsatisfied	Strongly unsatisfied
The airplane's advance and retreat	47 %	37 %	13 %	3 %	0 %

Table 7 Experimental result of M_4 module's detection rate

	The coach determined that the movements were correct	The coach determined that the movements were incorrect
M_4 module determined that the movements were correct	RR=137	NR=4
M_4 module determined that the movements were incorrect	RN=14	NN=25

The M_4 module detected mainly the shoulder exercise. When the subject performed the assigned shoulder exercise, the M_4 module would determine whether the subject completed the required movements. In addition, a coach would stand beside and record the accuracy of the movements. Therefore, there were four situations in this module. In the first situation, the coach determined that the movements were correct, and the M_4 module also determined that the movements were correct. The first situation is called RR. In the second situation, the coach determined that the movements were correct, but the M_4 module determined that the movements were wrong. We call the second situation RN. In the third situation, the coach determined that the movements were incorrect, and the M_4 module also determined the movements were incorrect. The third situation is referred to as NR. In the fourth situation, the coach determined the movements were incorrect, but the M_4 module determined the movements were correct. The fourth situation is called NN. Each subject performed each shoulder exercise 6 times. Thus, 30 subjects generate 180 exercises in total. Table 7 demonstrates the experimental results. The detection rate of M_4 module was calculated by (RR+NN)/(RR+RN+NR+NN). The detection rate was 90 %.

Figure 19 shows the cases where the M_4 module successfully detected the shoulder movements. Figure 19a demonstrates the cases where the coach determined the movement was correct, and the M_4 module also determined that the movement was correct. Figure 19b demonstrates the cases where the coach determined that the movement was incorrect, and the M_4 module also determined that the movement was incorrect. In these cases, the hands of the subject did not



(a) The coach determined the movement was correct, and M_4 module also determined that the movement was correct.

(b) The coach determined that the movement was incorrect, and M_4 module also determined that the movement was incorrect

Fig. 19 Cases of M_4 module's successful detection of the shoulder exercise

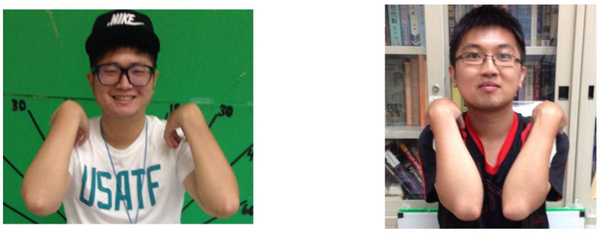

(a) The coach determined that the movement was correct, but M_4 module determined that the movement was wrong. (b) The coach determined that the movement was wrong, but M_4 module determined that the movement was correct.

Fig. 20 Cases of M_4 module's unsuccessful detection of the shoulder exercise

stay on the shoulders, and thus the movement should have been identified as incorrect.

Figure 20 demonstrates the cases where the M_4 module could not detect the shoulder exercise successfully. Figure 20a shows the cases where the coach determined that the movement was correct, but the M_4 module determined that the movement was incorrect. Figure 20b, however, shows the cases where the coach determined the movement was incorrect, but the M_4 module determined the movement was correct. In these cases, the elbows passed by the front of the body too quickly. As a result, the system was not able to identify the complete movement. Therefore, even if the coach determined that the movement was correct, the system considered the movement to be wrong. Another situation which often occurred was that the movement was not big enough, but the system still determined that the movement was correct. Thus, the M_4 module could not successfully detect the shoulder movement.

Table 8 demonstrates the result of the questionnaire with regard to the users' experience with the M_4 module's shoulder exercise. 87 % of the users believed that the exercise was effective. 13 % of the users felt neutral about the exercise. No user said that the exercise was ineffective.

J. Analysis of the performance of the M_5 module

M_5 detected mainly the neck exercise. The detection rate was calculated by a formula similar to that of the M_4 module. Table 9 demonstrates the detection rate of the M_5 module in the four situations. The total detection rate was 91 %.

Table 8 User experience about M_4 module's shoulder exercise

	Strongly agree	Agree	Neutral	Disagree	Strongly disagree
The exercise was effective	40 %	47 %	13 %	0 %	0 %

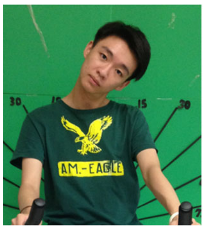
Table 9 Experimental result of M_5 module's detection rate

	The coach determined that the movements were correct	The coach determined that the movements were wrong
M_5 module determined that the movements were correct	RR=145	NR=12
M_5 module determined that the movements were wrong	RN=4	NN=19

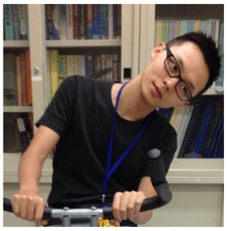
Figure 21 demonstrates the cases of successful detection of the neck exercise by the M_5 module. Figure 21.a shows the cases where the coach determined that the movement was correct, and M_5 module also determined that the movement was correct. Figure 21b demonstrates the cases where the coach determined that the movement was wrong, and the M_5 module also determined the movement was wrong. In this kind of case, the body inclined when the neck was bent, and therefore, the movement was determined to be incorrect.

Figure 22 demonstrates the cases where the M_5 module could not successfully detect the neck exercise. Figure 22a shows the cases where the coach determined that the movement was correct, but the M_5 module determined that the movement was wrong. Figure 22b, however, shows the cases where the coach determined that the movement was wrong, but the M_5 module determined that the movement was correct. In this kind of case, sometimes the coach determined that the movement was correct, but the movement did not pass the threshold of the system. Another situation that often occurred was that the user did look straight ahead, but the neck side-bending passed the threshold, and the system mistakenly determined that the movement was correct. Therefore, M_5 could not successfully detect the neck exercise.

Table 10 demonstrates the result of the questionnaire with regard to the users' experience with the M_5 module's detection of the neck exercise. 83 % of the users believed that the

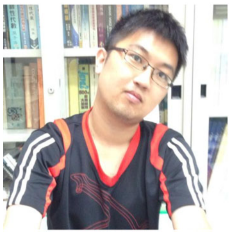


(a) The coach determined that the movement was correct, and M_5 module also determined that the movement was correct



(b) The coach determined that the movement was wrong, and M_5 module also determined that the movement was wrong

Fig. 21 Cases of M_5 module's successful detection of neck exercise


(a) The coach determined that the movement was correct, but M_5 determined that the movement was wrong

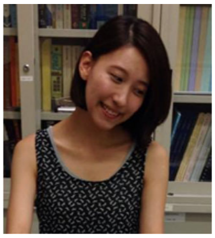

(b) The coach determined that the movement was wrong, but M_5 module determined that the movement was correct.

Fig. 22 Cases of M_5 module's unsuccessful detection the neck exercise

exercise was effective; 17 % of the users felt neutral about the exercise; no user considered the exercise ineffective, as shown in the result.

K. Analysis of user experience of special flying scenarios in Google Earth

Aerobatics motivated the users to do the neck and shoulder exercises. Table 11 shows the result of the questionnaire about how interesting the aerobatics in Google Earth was. 94 % of the users thought that the aerobatics was interesting or very interesting. 3 % felt it was OK, and 3 % of the users thought the aerobatics was dull. It is obvious that the proposed Exergaming system can effectively be applied to exercise bikes, as concluded from the experimental results of the system's movement detection rate, the effectiveness of the neck and shoulder exercise, and the entertainment provided by aerobatics.

Figure 23 demonstrates the trajectories of the airplane recorded after a complete Exergaming movement had been made (①->⑫). The users traveled virtually along the River Thames using the proposed system. The trajectory of spinning the airplane vertically and horizontally was activated after the user had completed special neck and shoulder movements. The other trajectories were made according to the user's trunk-flexion, trunk-extension, left side-bending, right side-bending, and pushing down or pulling up pedals. The corresponding trajectories, which were descend, climb, left banked-turn, right banked-turn, and flight speed control, of the airplane were triggered after the user's movement had been captured and determined by Kinect. The time spent in trunk-flexion, trunk-extension, left side-bending, right side-bending, and pushing down or pulling up pedals by the user influenced the final presentation of the trajectories.

Table 10 User experience about M_5 module's neck exercise

	Strongly agree	Agree	Neutral	Disagree	Strongly disagree
The neck exercise was effective	40 %	43 %	17 %	0 %	0 %

Table 11 The amusement of the aerobatics

	Strongly agree	Agree	OK	Disagree	Strongly disagree
The aerobatics was interesting	61 %	33 %	3 %	3 %	0 %

Figure 24 gives some snapshots of the proposed system. The upper left is an airplane's climb; the upper right shows an airplane's right banked turn; the lower left is an airplane's left banked turn; the lower right shows an airplane's descent. The proposed system efficiently control aircraft motions including climb, descend, left banked turn, right banked turn. It works well under pushing pedals down in Low-Speed Case and Middle-Speed Case. But, the proposed system doesn't work fluently under pushing pedals down in High-Speed Case. Some unsmoothed airplane trajectory would occur between successive frames. Since the proposed algorithm is performed frame by frame individually, the detection rate of the body movement algorithm is not 100 %. These errors would make wrong trajectories of the airplane, producing unsmoothed flying in the Google Earth. Therefore, a fuzzy scheme is necessary to overcome the problem in the future. The proposed system is only limited in indoor environment now. To implement such a form of exercise that would represent a form of cycling outdoors is also included in our future works.

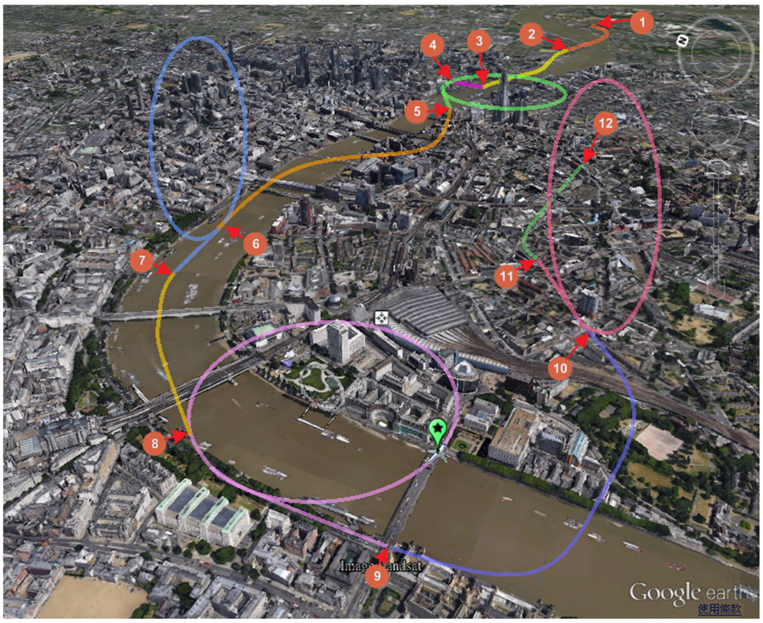
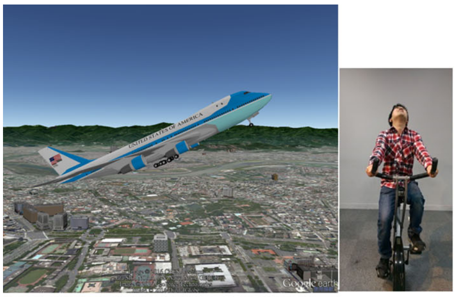
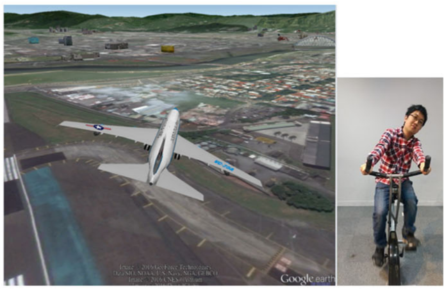
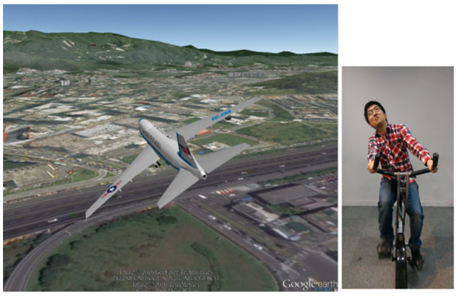


Fig. 23 Demonstrative trajectories of the airplane recorded after a complete Exergaming movement


(a) Snapshot for an airplane's climb.


(b) Snapshot for an airplane's right banked turn


(c) Snapshot for an airplane's left banked turn

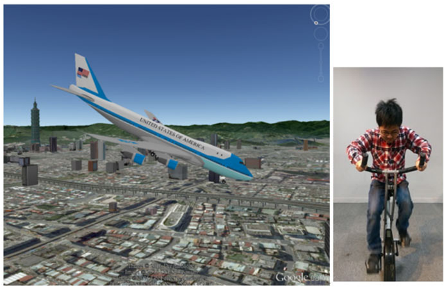

(d) Snapshot for an airplane's descent

Fig. 24 Some snapshots of the proposed system

5 Conclusion

The goal of this paper was to design an Exergaming system which can be applied to exercise bikes. The major contribution of this paper is the proposed detection algorithm for specific body movements and neck and shoulder exercises. The skeleton joint point information provided by Kinect and face tracking parameters were used for detecting the above movements. Three FSMs were proposed to detect consecutive movements. Moreover, we designed simple flying modules and adopted Google Earth API to make the visualized 3-D dynamic motion element. With the flying modules and the abovementioned element, the users were able to control the airplane's trajectory and travel in a virtual world with various exercise modes on the exercise bike. We also designed multiple flying scenarios to add interest to the system. The data of the experimental results show that the proposed Exergaming system is very entertaining. It also has good effect for stretching the muscles in the neck and the shoulders. In practical application, any user can easily setup the proposed Exergaming system by just integrating with Kinect and any type of exercise bike. The proposed Exergaming system is a good assisting system for exercise bikes.

Acknowledgments This research is supported in part by the Ministry of Science and Technology, Taiwan under the grant of MOST 103-2221-E-130 -018.

References

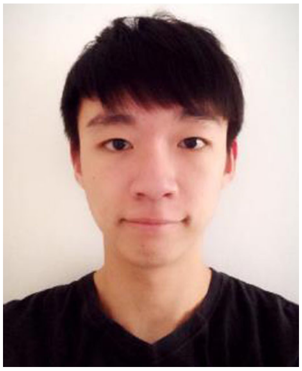
1. Adams RJ, Lichter MD, Krepkovich ET, Ellington A, White M, Diamond PT (2015) Assessing upper extremity motor function in practice of virtual activities of daily living. Neural Systems and Rehabilitation Engineering, IEEE Trans
2. Chen L, Wei H, Ferryman J (2013) A survey of human motion analysis using depth imagery. Pattern Recogn Lett 34.15, 1995–2006
3. Dowling AV, Barzilay O, Lombrozo Y, Wolf A (2014) An adaptive home-use robotic rehabilitation system for the upper body. Translational Engineering in Health and Medicine, IEEE J
4. Evett L, Burton A, Battersby S et al (2011) Dual cameramotion capture for serious games in stroke rehabilitation. In Proceedings of the IEEE 1st International Conference on Serious Games and Applications for Health (SeGAH '11), Braga, p 1–4
5. Gao Y, Wang M, Ji R, Wu X, Dai Q (2014) 3-D object retrieval with hausdorff distance learning. IEEE Trans Ind Electron 61(4):2088–2098
6. Garrido JE, Marset I, Penichet VMR, Lozano MD (2013) Balance disorder rehabilitation through movement interaction. Pervasive Computing Technologies for Healthcare (PervasiveHealth), 7th International Conference
7. Girshick R, Shotton J, Kohli P, Criminisi A, Fitzgibbon A (2011) Efficient regression of general-activity human poses from depth images. Computer Vision and Pattern Recognition (CVPR), IEEE Conference
8. Han J, Member, IEEE, Shao L, Senior Member, IEEE, Xu D, Member, IEEE, Shotton J Member, IEEE (2013) Enhanced computer vision with microsoft kinect sensor: a review. IEEE Trans Cybern 43(5)
9. Hong P, Turk M, Huang TS (2000) Gesture modeling and recognition using finite state machines. In Proceedings of fourth ieee international conference on Automatic face and gesture recognition, IEEE, p 410–415
10. Hossein Mousavi H, Khademi M (2014) A review on technical and clinical impact of microsoft kinect on physical therapy and rehabilitation. J Med Eng 2014, 846514
11. Jun S-k, Kumar S, Zhou X, Ramsey, DK, Krovi VN (2013) Adaptive for individualization of kinect-based quantitative progressive exercise regimen. Automation Science and Engineering (CASE), IEEE Int Conf
12. Khademi M, Dodakian L, Hondori HM, Lopes CV, McKenzie A, Cramer SC (2014) Free-hand interaction with leapmotion controller for stroke rehabilitation. In Proceedings of the 32nd Annual ACMConference on Human Factors in Computing Systems (CHI EA '14), ACM, New York, p 1663–1668
13. Lee J-d, Hsieh C-H, Lin T-Y (2014) A kinect-based tai chi exercises evaluation system for physical rehabilitation. Consumer Electronics (ICCE), IEEE Int Conf
14. Lin T-Y (2013) A Kinect-based system for physical rehabilitation: utilizing tai chi exercises to improve movement disorders in patients with balance ability. Modelling Symposium (AMS), 7th Asia. ID 846514
15. Obdrzalek S, Kurillo G, Ofli F et al (2012) Accuracy and robustness of Kinect pose estimation in the context of coaching of elderly population. In Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS '12), San Diego, p 1188–1193
16. Padma T, Balasubramanie P (2011) Domain experts' knowledge-based intelligent decision support system in occupational shoulder and neck pain therapy. Appl Soft Comput 11(2):1762–1769, **8p**
17. Poppe R (2007) Vision-based human motion analysis: an overview. Comput Vis Image Underst 108(1):4–18
18. Shotton J, Fitzgibbon A, Cook M, Sharp T, Finocchio M, Moore R, Kipman A, Blake A (2011) Real-time human pose recognition in parts from single depth images. Computer Vision and Pattern Recognition (CVPR), IEEE Conference
19. Sinclair J, Hingston P, Masek M (2007) Considerations for the design of exergames. Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia, p 289–295
20. Sucar LE, Azćarate G, Leder RS (2009) Gesture therapy: a vision-based system for arm rehabilitation after stroke. In: Fred A, Filipe J, Gamboa H (eds) Biomedical engineering systems and technologies. Springer, Berlin, pp 531–540
21. Tanaka K, Parker JR, Baradoy G, Sheehan D, Holash JR, KatzL (2012) A comparison of exergaming interfaces for use in rehabilitation programs and research. Loading 6(9)
22. Tang Y, Wu H, Liu P, Si Q (2012) Real-time 3D flight track and flight simulation based on Google Earth. Int J Digit Content Technol Appl 6(19):385–392, **8p**
23. van Diest M, Stegenga J, Wörtche HJ, Postema K, Verkerke GJ, Lamoth CJC (2014) Suitability of Kinect for measuringwhole bodymovement patterns during exergaming. J Biomech 47(12):2925–2932
24. Wang YK, Cheng KY (2010) A two-stage bayesian network method for 3D human pose estimation from monocular image sequences. EURASIP J Adv Signal Process

25. Wu C-H, Tseng Y-C (2011) Data compression by temporal and spatial correlations in a body-area sensor network: a case study in pilates motion recognition. IEEE Trans Mob Comput 10(10):1459–1472
26. Zhao S, Chen L, Yao H, Zhang Y, Sun X (2015) Strategy for dynamic 3D depth data matching towards robust action retrieval. Neurocomputing 151:533–543
27. Zhao S, Yao H, Zhang Y, Wang Y, Liu S (2015) View-based 3D object retrieval via multi-modal graph learning. Signal Process 112:110–118

Shih-Yu Huang received the BS degree in information engineering from Tatung Institute of Technology, Taipei Taiwan, Republic of China, in 1988, and the MS and PhD degrees from Department of Computer Sciences, National Tsing Hua University Taiwan, Republic of China, in 1990 and 1995, respectively. From 1995 to 1999, he worked in the telecommunication laboratories of Chunghwa Telecom Co., Ltd., Taiwan. In October 1999, he joined the Department of Computer Science and Information Engineering, Ming Chuan University, Taiwan. His current research interests are video processing and steganography.

Jen-perng Yu received the BS degree in Aeronautics from Tamkang University, Tamsui Taiwan, Republic of China, in 1983, and the MS degree in Aeronautics and Astronautics from National Cheng Kung University, Tainan Taiwan, Republic of China, in 1985 and the PhD degree in Aeronautics and Astronautics from University of Tokyo, Japan, in 1996. He worked in National Chung-Shan Institute of Science & Technology(CSIST), Taiwan, from 1985 to 2005. In August 2005, he joined the Department of Information Management, Ming Chuan University, Taiwan. His current research interests include Web3D Data Visualization, Web-Based and Smartphone Applications, Virtual Reality and Augmented Reality.

Yuan-Kai Wang received the B.S. degree of electrical engineering in 1990 and Ph.D. degree of computer science and information engineering in 1995, from National Central University. He was a postdoctoral fellow in the Institute of Information Science of Academia Sinica from 1995–1999. From 1999 he joined the department of electrical engineering, Fu Jen University as an associate professor. He has been the Chair and Program Committee member of many international conferences, and reviewers of many journals and IEEE transactions. His research interests include computer vision, pattern recognition, and machine learning. He studied applications of the above methods on video surveillance, face recognition, biometrics, and robotic vision. Currently he is interested in machine learning and video analysis for visual surveillance, embedded computer vision, human-computer interface, robotics and multimedia.



Jia-Wei Liu received the bachelor degree in information engineering from Ming-Chuan University, Taoyuan Taiwan, Republic of China, in 2013, and graduated Ming Chuan Institute, 2015. His major is computer science and information engineering. He worked in interactive media laboratories. His current job is related to image processing.