Recent Developments in Game-Based Virtual Reality Educational Laboratories Using the Microsoft Kinect

https://doi.org/10.3991/ijet.v13i01.7773

Mingshao Zhang(✉)
Southern Illinois University Edwardsville, IL, USA
mzhang@siue.edu

Zhou Zhang
New York City College of Technology, New York, NY, USA

Yizhe Chang, El-Sayed Aziz, Sven Esche, Constantin Chassapis
Stevens Institute of Technology, Hoboken, NJ, USA

Abstract—Virtual Reality (VR) is a well-known concept and has been proven to be beneficial in various areas. However, several disadvantages inherent in VR prevent its broad deployment in the educational arena. These limitations include non-realistic representation, lack of customizability and flexibility, financial feasibility, physical and psychological discomforts of the users, simulator sickness, etc. In this paper, an innovative method that uses the Microsoft Kinect as an essential component for developing game-based VR educational laboratories is presented. This technique addresses three different aspects. First, it represents an efficient method for creating the VE using the Kinect as a measuring tool. Second, the Kinect is employed as a substitute DAQ system for acquiring range data and tracking the motion of objects of interest. At last, the Kinect serves as a novel human-computer interface for tracking the users' entire body motion and recognizing their voices. Using the method described here, three major aspects of educational VR development can be accomplished with an inexpensive and commercially available Kinect.

Keywords—Microsoft Kinect; Game-based Virtual Reality; Educational Laboratory; 3-D Virtual Environment; Object Tracking; Natural Human-computer Interface

1 Introduction of Virtual Reality

1.1 VR in Education

VR is defined as a computer-generated simulation of a 3-D environment that users can interact with in a seemingly real or physical way using special electronic equipment, such as a helmet with a screen inside or gloves fitted with sensors. VR is becoming increasingly popular and it is now widely used in research, education and

training. Using virtual systems instead of physical ones can make the creation of multiple copies of physical devices unnecessary, thus reducing the consumption of natural and social resources. In addition, virtual systems are inherently safer and less failure prone than physical ones and can be shared remotely by multiple users. There are several important categories of VR, including desktop VR, immersive VR, distributed VR and augmented VR.

Since the idea of VR was introduced, there has been a dramatic surge of interest in using VR technology beyond the entertainment industry such as in professional education and training [1]. VR technology has also been used in K-12 and higher education, integrating various peripheral devices such as head-mounted display gear, data gloves and body suits to give users a fully immersive learning experience (see for instance Science Space and Safety World [2]). However, many practical concerns and limitations that have restricted the wide dissemination of VR technology in K-12 and higher education were reported. The financial feasibility is one of the many reasons that prevent VR technology from being implemented in schools. The cost of both developing and maintaining various sophisticated devices for creating immersive environments is high [3]. In addition, there are many physical and psychological discomforts that users experience in VR environments, for example the weight and fit problem of the peripheral devices, simulator sickness, disorientation, etc. [4]. Ever since VR attracted attention by the gaming industry, many low-cost VR devices have been developed in recent years, such as the HTC Vive, the Sony PlayStation VR, the Oculus Rift, the Google Cardboard, the Microsoft HoloLens, etc. The above-mentioned concerns regarding the high cost of the devices and physical discomforts have almost been eliminated. However, it is still a rather expensive proposition to develop and maintain a VR system, especially for the educational domain since only limited resources are devoted to developing such applications. Regarding these concerns, a desktop-based VR technology would be desirable in K-12 and higher education. Although it cannot provide a fully immersive experience compared to VR headsets, photo-realistic computer graphics have shown their adequacy in many educational applications [5].

There has been an increase in demands for innovations in higher education for engineering. Taking advantage of the advancements in 3-D visualization technologies, many engineering teaching and training materials can be implemented in VR environments. However, engineering education is predominantly descriptive and complex. One of the main techniques in engineering education is the use of laboratories to enhance the students' practical knowledge and skills. Educational laboratories are designed to improve the students' ability to investigate physical phenomena and to solve engineering problems with appropriate levels of independent thought and creativity [6]. Therefore, it is very important for educational institutions to design and implement VR laboratories properly.

1.2 Kinect-based Applications

Using the Kinect in education has become a popular topic since its introduction in 2010, when it was only sold for entertainment purposes. Online communities soon

cracked the Kinect code in order to utilize the functionalities that are not officially released by Microsoft. In 2012, Microsoft released the Kinect for Windows with an open-source Kinect software development kit (Kinect SDK), thus allowing customers to develop their own applications with fewer technical difficulties. Due to the high price and unavailability of commercial 3-D scanners, the Kinect is frequently used as an adequate substitute. Applications of the Kinect include scanning users' faces and creating personalized avatars [7], modeling indoor environments [8], reconstructing a virtual laboratory inside a game environment [9] and real-time 3-D reconstruction [10].

Using the Kinect to track the motion of objects other than the motion of human bodies is a less popular research topic. However, there is still some research progress being reported. For instance, the Kinect's color and depth data have been used to estimate 3-D motion flow [11], to estimate 3-D motions based on scene particles [12] and to track object motion inside of educational laboratories [13,14].The Kinect can potentially serve as a lecture tool. During lectures, it would be easier for instructors to use their gestures to control computers rather than to click a mouse or to tap on a keyboard repeatedly. For instance, an instructor can use a combination of gestures to control a slide show [15,16]. While an instructor wants to demonstrate the kinematics of a robot with the help of the Kinect, he/she can guide the robot by waving his/her arms, which is faster than operating the robot via a traditional interface and thus saves lecture time [17].

1.3 Overview of Game-based VR Using Kinect

As mentioned above, desktop-based VR technology is desirable in developing applications for K-12 and higher education. A desktop VR system consists of two major subsystems, the hardware and the software. The hardware can be further divided into VR engine (i.e. desktop computer) and I/O devices, while the software can be divided into application software and database. As parts of the hardware components of the desktop VR, the input devices are the means by which the users interact with the VE. The traditional way to accomplish this was to use keyboard and mouse. Nowadays, common desktop computers are powerful enough to serve as VR engines that perform tasks such as graphical models generating, object rendering, lighting, mapping, texturing, simulation and display in real-time. The output devices get feedback from the VR engines and pass it on to the users. The commonly seen output devices are monitors and speakers.

The VR application software is a collection of tools for designing, implementing and maintaining the VE and the database where the information is stored. The tools can be further classified as modeling tools and development tools. Widespread modeling tools include 3ds Max, CATIA, Pro/E, etc. A massive multiplayer online game (e.g. Garry's Mod) was chosen as the VR environment and development tool in the project described here. This game has several advantages such as open source, ease of animation, built-in collision detection modules and support of I/O devices and communications between users.

2 Traditional methods for creating virtual reality systems

2.1 Overview of Kinect-based VR system

In the application presented in this paper, namely the development of a game-based educational VR system, the Kinect is used for three different purposes. Firstly, the Kinect is used as a VR modeling tool for virtualizing a real laboratory scene and creating the corresponding VE inside the game engine, both of which are accomplished in a real-time manner. Secondly, the Kinect serves as a hardware interface between the experimental devices and the VR engine, thus replacing the traditional DAQ system for tracking the motion of objects. At last, the Kinect is also deployed as a novel human-computer interface that replaces the conventional input method with keyboard and mouse. The users' body motion and voice commands control the navigation and operation within the VE. In the following, the traditional methods for creating VEs, implementing hardware interfaces for data acquisition and constructing human-computer interfaces will be introduced first. The details of our innovative way of implementing these three aspects will be explained in the Sections 3, 4 and 5.

2.2 Traditional Methods for Virtualizing the Real World

In order to successfully develop a VR system, several challenges need to be overcome. One major aspect of these challenges is associated with the VE creation, which includes the creation of the VS and models of real objects. Especially desirable are the creation of the VE and models of physical devices in real time and the seamless integration of different platforms and systems. Another challenge is the efficient interaction between humans and the VE including 3-D sound synthesis, stereoscopic 3-D display technologies and perception feedback [18].

The traditional approach for creating a VE often involves CAD software, which makes the process complicated and time consuming. With the development of new electronic components, emerging electro-optical technologies, powerful algorithms and data processing tools, improvements in the efficiency of creating a VS are becoming feasible. 3-D reconstruction techniques have great potential in both object reconstruction and environment reconstruction. Recently, innovative applications of 3-D reconstruction including 3-D voxel human motion reconstruction [19], 3-D human face reconstruction [20] and urban scenes reconstruction from 2-D videos [21] have been reported.

There are two main approaches for obtaining the shape information of real objects. The first approach is to retrieve the surface information from static 2-D images. For instance, '3DSOM Pro' and 'insight3d' can be used to create 3-D models through a series of alignment pictures [22,23]. The other approach is to acquire the surface information of real objects with dynamic scanning devices. For example, 'ReconstructMeQt' can be employed to generate 3-D surfaces with the Kinect and convert 3-D surface files into other formats [24]. The 'KinectFusion' is one of the tools for developing 3-D reconstruction applications. It can be used to generate 3-D models in real time from the objects' shape information that was acquired by the Kinect [10,25].

2.3 Data Acquisition through Object Tracking

Laboratories are widely acknowledged to have significant educational value for engineering and science students. They often use a DAQ system for the measurement of certain parameters that characterize the investigated physical phenomena [26]. Range data (linear/angular position, velocity, etc.) is an example for such parameters. Many experiments in educational laboratories require data on the motion of certain objects, which makes object tracking an important aspect in the implementation of experiments. The task of determining and tracking the objects' location and orientation relative to the environment has received considerable attention over the past few years. Judged by the hardware used in object tracking systems, the most commonly deployed state-of-the-art techniques can be divided into four classes.

1. Dead reckoning: Encoders are used to compute translational movements from rotational measurements based on integration. This class is theoretically simple and easy to be integrated into experimental devices. However, it is considered noisy and less robust than other classes [27].
2. Inertial navigation systems: Inertial sensors, accelerometers and detectors for electromagnetic fields and gravity are used to compute motions based on integration. This class can lead to error accumulation, especially when drift-prone sensors are used [28].
3. Ranging: Laser, infrared, acoustic or radio signals are used to measure distance. This class tends to be unreliable in highly dynamic environments, limited in range and expensive [29].
4. Visual odometry: Single-camera, stereo-vision or even omnidirectional imaging is used to determine the position and orientation of objects of interest. This class is considered to be computationally expensive. However, it has become a popular research topic lately. It can be improved significantly by embedding signal processing techniques and adopting new computer-vision algorithm developments [30].

In order to build a simple system and render it easy to operate even for inexperienced users, the ideal experimental setup should consist of a single camera or groups of cameras fixed in the environment while the experimental devices should not require additional built-in sensors. The objects' motions are determined by observing and extracting relevant information on easily identifiable elements in the environment (i.e. walls, objects such as desks, chairs, etc.). Using a 3-D scanner (e.g., the Kinect) instead of stereo vision cameras allows further simplification of the hardware setup because of the depth information that this approach provides [13,14]. If appropriately placed in the environment, a single Kinect could suffice for capturing the entire motion of the objects of interest.

2.4 Human-computer Interaction

Compared with other simulated technologies, one important advantage of implementing VR technology is the ability to integrate natural human motion into real-

world scenarios. This provides an ideal simulation medium for many immersive educational applications, such as for example mechanical assembly activities. Novel VR I/O devices such as 3-D glasses, motion-tracking gloves, haptic sensors, etc. are able to satisfy fundamental operational needs in these immersive applications. However, most of these implementations focus on simulations such as computer-aided design, which are geared toward professionals rather than students, thus leading to complicated operating procedures that are not suitable for students. Furthermore, the costs of these novel VR devices and specially designed VR platforms represent an untenable financial burden for most educational institutions.

Unlike expensive immersive VR equipment, desktop VR can be run on personal computers with standard I/O devices such as desktop monitor, keyboard, mouse, etc. 3-D game engines, especially first-person-shooter (FPS) game engines, can be used as a type of desktop VR [31]. With the widespread availability of low-cost computer graphics hardware, screen-based 3-D game engines provide realistic, flexible, interactive and easy-to-learn tools for people to navigate and operate inside of 3-D environments [32]. A basic 3-D game engine includes graphics, physics, audio and artificial intelligence engines as well as a human-computer interface module. In most cases, small-scale multiplayer capabilities are also included in a game engine [33]. Since most game engines provide SDKs for programmers to make modifications, there are reports of integrating assembly validation components and machine dynamics simulation components into game engines to meet the needs of students [34,35].

For many years, the primary input devices for computers have been the keyboard and mouse. This combination is practical in daily computer uses such as document editing, webpage surfing, etc. However, for navigating in 3-D VEs, these devices fall short due to their unrealistic mapping of computer operation and real-world activities. For instance, most FPS game engines set the keys ‘A’, ‘S’, ‘D’ and ‘W’ on the keyboard to control translational movements in 3-D VEs and use the mouse to control rotational movements. Some other in-game commands (e.g. crouching down and walking slowly) are given through tapping or holding other keys. Experienced game players can perform these operations effortlessly and fluently. However, for beginners these operations are hard to perform. Since the level of gaming experience of students varies dramatically [36], the question arises whether it is possible to provide them with a versatile input method to promote their overall learning performance in game-based simulated experiments. New natural human-computer interface devices are making this possible. Driven by the entertainment industry, motion sensors such as the Nintendo Wii Remote, the PlayStation Move controller, the Microsoft Kinect Sensor, etc. are becoming more and more popular for facilitating natural human-computer interactions at low cost. Based on different physical principles, these devices each have certain advantages. For instance, the Wii and the PlayStation Move controller are hand-held devices that can only track the motion of the player’s hand with high accuracy.

Although simple in structure, the Kinect is capable of dynamically capturing the movement of the entire human body, thus making it the most suitable choice among these new devices for engineering educational laboratories. This is not only owed to the fact that immersive simulations in desktop VR need the kinematic information of

the entire body (e.g. a human avatar without feet would look really strange), but also because a lot of operations that are typically performed in engineering experiments (e.g. mechanical assembly) usually require the cooperation of one's multiple body parts.

3 Real-time Virtualization of the Real World

3.1 Methodology of Kinect-based Virtual Environment Creation

The efficiency of the current VE creation approaches is hampered by the methods for acquiring the geometric parameters of the real world and by the selection of the mapping tools. Therefore, the laborious work of surveying the real world could be completed more quickly if an efficient sensor was used to replace the traditional measuring devices. Moreover, the entire process of creating a VE can be simplified significantly if the system is adapted with some middleware that can process the acquired raw data and generate ready-to-use maps and models automatically.

Various kinds of non-contact measurement devices such as 2-D cameras and 3-D scanners are potential tools for surveying the real world. However, 2-D cameras can only acquire 2-D images. Therefore, complicated algorithms are required for generating 3-D geometries. These kinds of devices are only suitable for applications that have low requirements for both accuracy and computational speed. There are two kinds of commonly encountered 3-D scanners, namely laser scanners and infrared scanners. Laser scanners are usually more expensive and require experienced users to operate [37]. On the other hand, infrared scanners such as the Kinect are more affordable, accessible and flexible [38].

A hand-held Kinect was chosen here for scanning the real world. The main steps of the data acquisition and processing are depicted in Figure 1. The Kinect generates both depth and color information at the same time, and subsequently those data are mapped into the output in the form of a 3-D point cloud. The subsequent steps include finding the Kinect's pose, generating a global 3-D point cloud, subdividing this point cloud and meshing the resulting sub-point clouds.

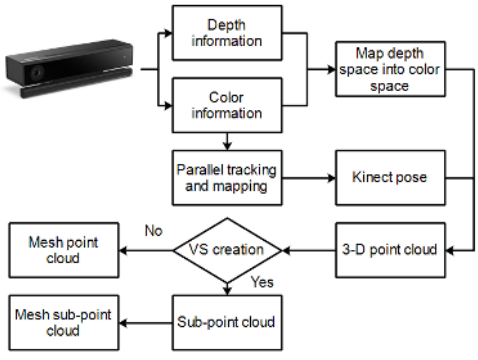

Fig. 1. Flow chart of data acquisition and processing

In order to determine the Kinect's pose, the parallel tracking and mapping (PTAM) method [39] is used, which splits the tracking and mapping processes into two computation threads that run in parallel. The tracking process uses the recorded sequential color frames to compute the Kinect's extrinsic and intrinsic parameters. The extrinsic parameters represent the camera's location and orientation in the world coordinate system. The intrinsic parameters represent the relationships between the camera coordinates and the pixel coordinates. Together they form the projection matrix P (see Equation 1)

$$\begin{cases} P = K_{3\times3}[R_{3\times3} \quad t_{3\times1}]_{3\times4} \\ \begin{bmatrix} u \\ v \\ w \end{bmatrix} = P \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \end{cases} \tag{1}$$

The intrinsic and extrinsic parameters are represented by a 3×3 matrix $K_{3\times3}$ and a 3×4 matrix $[R_{3\times3} \quad t_{3\times1}]_{3\times4}$, respectively [40]. The projection matrix P maps the points from homogeneous world coordinates $(x, y, z, 1)$ t to 2-D image coordinates (u, v, w). A 4×4 matrix M models the motion of the camera and a 4×4 matrix represents the camera's pose (i.e. location and orientation). Then, changes of the Kinect's pose can be described by Equation 2, where is an exponential map of a 6-dimensional vector v_m that represents the Kinect's motion (i.e. 3 rotations about the 3 axes and 3 translations along the 3 axes). Once v_m has been determined, the pose of the Kinect can be obtained. The procedures for tracking the pose of the Kinect are described in detail elsewhere [39].

$$C_p^{'} = MC_p = e^{v_m}C_p \tag{2}$$

When surveying the real world, the Kinect's pose in the first frame is taken as the reference for the entire world space. The remaining sequential data recorded by the Kinect are then transformed into the world coordinates using the Kinect's computed pose matrix C_p. For model creation purposes, the point cloud is processed directly and it generates a single model of the corresponding real object. For VS creation purposes, the acquired point cloud represents the surfaces in world coordinates, which is then divided into 6 sub-point clouds according to the 6 views of the world (top, bottom, left, right, front and back). Subsequently, each sub-point cloud is meshed with the 3-D Delaunay triangulation algorithm [41]. Finally, the sub-point clouds form 6 independent parts of the map. These parts are combined into one single model, which ensures that the VS created is a penetrable solid. More details about these processes can be found elsewhere [42].

3.2 Validation by Creation of VE of a Game-based Virtual Laboratory

After creating the triangle mesh, the models can be created by attaching corresponding textures, but the creation of the VS is more complicated. Since each side of the VS is composed of a set of triangles, the creation of the VS requires the construc-

tion of one model for every side of the VS, followed by joining them together. The textures of the model are defined as color pixel points, which are located both inside of the triangular domains and on their boundaries. In the next step, all these triangular texture patches are combined into 6 images corresponding to the 6 directions of the VE. After converting these 6 images into 6 VTF texture files using 'VTFCmd.EXE' (the EXE files mentioned in this chapter can found in the Valve Hammer Editor [43]), the models are created by storing the complete texture and vertex information in a SMD file. However, to create the VS, the texture and vertex information is stored in a VMF file. Finally, the SMD file is compiled by 'StudioMDL.EXE' into the file formats that can be directly used by the game engine and the VMF file is compiled into BSP map format in three steps. The geometry is rendered with 'VBSP.EXE', the geometry is created with 'VVIS.EXE' and the lighting conditions are determined with 'VRAD.EXE'. The entire procedure is illustrated in Figure 2.

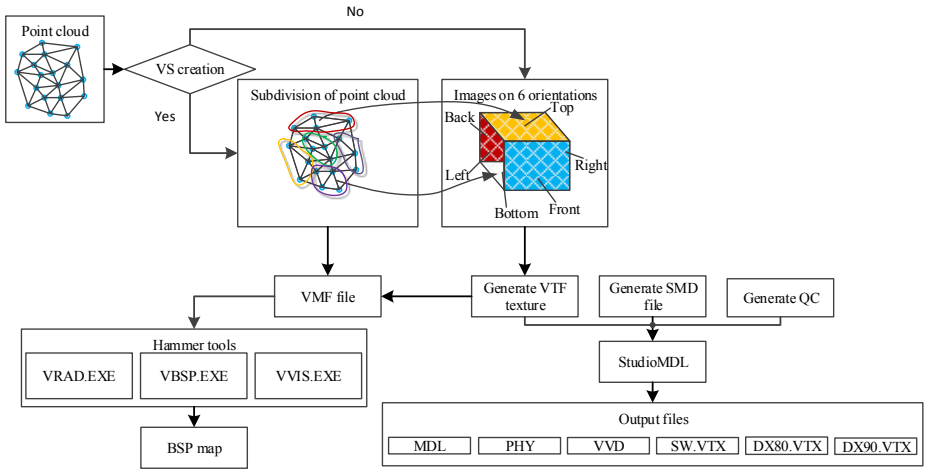

Fig. 2. Meshes, textures and reconstruction

In order to validate the feasibility of the method proposed here, a step motor and a map of a game-based virtual laboratory were created using the procedures described above (see Figure 3). A straightforward method for constructing the models is to detect all of their surfaces in the real world, mesh all of these surfaces into triangles and map textures to these triangles. However, for creating the VS, this approach is too costly in terms of computational time and resources, and it is also undesirable considering the many unnecessary details of the real world. In the map shown in Figure 3, each of the six views of the laboratory was simplified into 2 large triangles and directly attached to the image of the corresponding direction in the real laboratory. This method saves a lot of computational time and resources. Furthermore, in order to reduce the number of vertexes needed for meshing and texturing, some keypoints of the laboratory were extracted from the sub-point clouds (e.g. corner coordinates of one wall or a big cabinet of the laboratory). These keypoints can represent the 6 sides of the laboratory from 6 directions coarsely. Although this map is not perfect, it is

sufficient for the purpose of implementing experiments in a virtual laboratory. The quality of the VE versus the memory it required to store the map is always a difficult trade-off when selecting the software for creating a VR map. The above-mentioned KinectFusion [10] is not suitable for creating map for the entire room because it would need an extremely large memory to store the map.

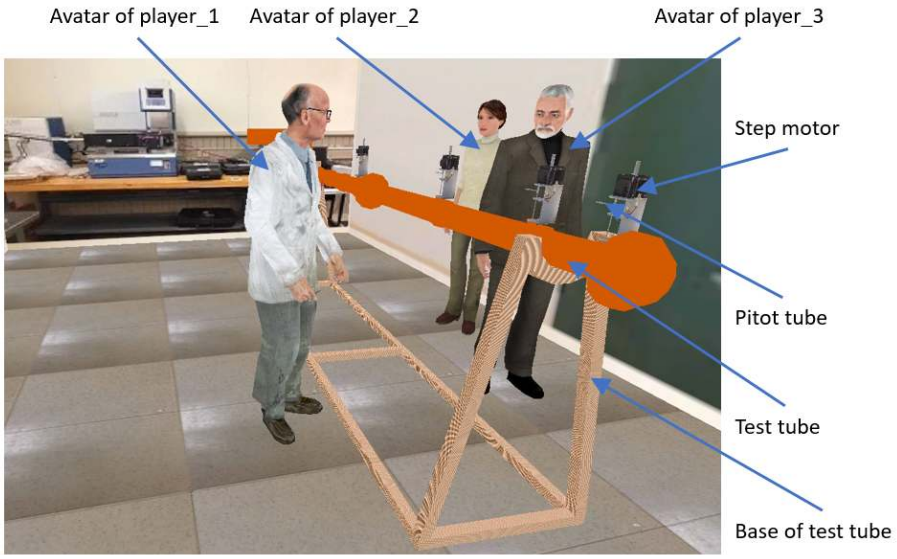

Fig. 3. Laboratory map with fluid experiment setup

4 Kinect-based Object Tracking

4.1 Methodology of Kinect-based Tracking

In order to serve as a substitute for a traditional DAQ system in educational laboratories, the proposed Kinect-based system should provide certain functions. The most important one of these functions is that it should be able to distinguish multiple moving objects both from the background and from each other, recognize the objects of interest among these moving objects, and track these objects of interest by computing the changes in their six degrees of freedoms over time. The raw data produced by the Kinect are sequences of point clouds, in which each frame contains a large number of points with associated location and color information. In order to achieve the above-mentioned functionality with appropriate efficiency and accuracy, the Kinect-based system consists of algorithms that can be divided into three stages. The first stage, referred to here as pre-processing, includes functions such as scene segmentation to improve the computational speed as well as noise removal to enhance the accuracy of the computed results if needed. Depending on the complexity of the geometry and texture of the objects of interest, different algorithms are chosen for object recognition in the second stage. The post-processing represents the last stage, in which the

changes in the objects' six degrees of freedom are computed. More detailed descriptions can be found elsewhere [13,14,44,45].

In order to distinguish the differences between various motions in the scene and to track only the objects of interest, algorithms that were originally designed for object recognition can be adapted and integrated into the applications discussed here. Based on the complexity of the potential objects' texture and geometry, recognition-based tracking methods can be divided into three groups: color/texture-based tracking (suitable for simple scenes but exhibiting limited reliability of results in other conditions), simple geometry tracking (suitable when objects have simple geometries regardless of texture conditions), and complex geometry tracking (preferred for intricate objects or scenes and performing better if integrated with texture-based tracking methods).

In many educational laboratories, especially those for conducting physics experiments, most of the targeted moving objects tend to have simple geometries. Among the algorithms that can be used for tracking objects with simple geometries, two are widely known, namely the Hough transform [46] and the RANSAC paradigm [47]. In the prototype application presented here, the RANSAC approach is used for shape detection. This method has been proven to work effectively in 2-D as well as 3-D applications. Its advantages include that its conceptual simplicity make it straightforward to be implemented and easily to be extended, it requires only limited modification for being integrated in a wide range of applications and it works robustly even in the presence of a high proportion of outliers (i.e. high noise level). The main disadvantage of the traditional RANSAC method is that it has low computational efficiency and high memory demands. Both of these disadvantages are less severe in the application presented here because the size of the input point cloud is dramatically reduced by the above mentioned segmentation process. Also, a more efficient RANSAC method was adopted and implemented elsewhere [48].

The RANSAC paradigm can be used for detecting several different shapes, including planes, spheres, cylinders, cones, etc. Here, the sphere detection is used as a prototype implementation. The RANSAC paradigm works by randomly drawing minimal point sets from the input point cloud and computing the corresponding shape primitives. Two points (p_1, p_2) and their corresponding normal vectors (n_1, n_2) can be used to fully define a sphere. The center c of this computed sphere is defined as the midpoint of the shortest line segment between the two lines that are given by the above mentioned two points and their normals, and the sphere radius is computed with Equation 3. The computed sphere is treated as an acceptable candidate if all points are within a certain distance ε from the sphere's surface and their normal vectors deviate by less than a certain angle α from that of the sphere. The motion changes of this sphere can be tracked by implementing the RANSAC detection in all consecutive point cloud frames. More details about the usage of the RANSAC for detecting the motions of objects other than spheres and using the Kinect for tracking objects with complex geometries and textures can be found elsewhere [44].

$$r = \frac{\|p_1 - c\| + \|p_2 - c\|}{2} \tag{3}$$

4.2 Prototype Implementation of Kinect-based Hardware Interface

As an illustrative example of the RANSAC sphere detection method described above, a Foucault pendulum experiment involving a spherical bob was chosen as a prototype implementation (see left part of the Figure 4). Only those points that represent the spherical part are extracted from a series of point clouds and the bob positions (x, y, z coordinates of the sphere's centroid) are computed and recorded. The results are shown in Figure 5. For small pendulum amplitudes, the bob stays in almost the same latitude (i.e. small changes in y coordinate). The noisiest part is the change in the z coordinate (i.e. the depth direction for the Kinect's camera coordinate, which is horizontal for the pendulum's setup), which is reasonable because the Kinect has the lowest accuracy in the depth direction. From the acquired position data, the gravitational acceleration g is computed to be 9.77 m/s^2, which is acceptably accurate in light of the measurement errors caused by the Kinect and the small angle approximation used in the computations.

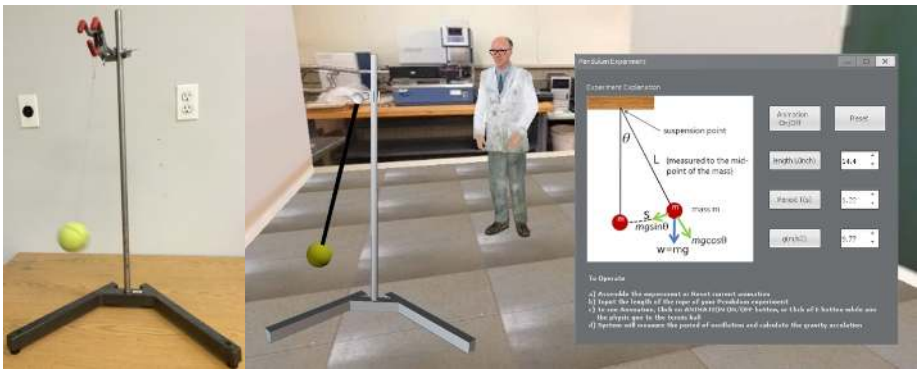

Fig. 4. Experimental setup of Foucault pendulum (left); Foucault pendulum experiment implementation in game-based VE (right)

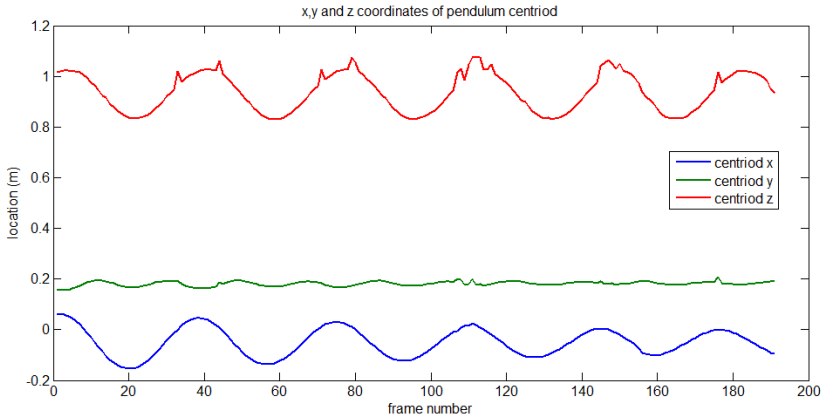

Fig. 5. Pendulum tracking result; left: result plot of x, y, z coordinates, right: result plot in x-z plane

The acquired point cloud data and the computed position results can be used in the pendulum experiment for education purposes. In the VR laboratory presented in this paper, the users (e.g. instructor, teaching assistant and students) conduct the experiments and interact with each other within a game-based VE. The pendulum experiment is recorded by the Kinect, position information of the pendulum bob is then computed, and the results are sent to the game-based VR laboratory as input data to animate the experiment inside the VE. Usually, this detection, computation and communication process can be accomplished either in real-time or in batch mode, depending on the complexity of the experiment. Inside the VE, the users are represented as avatars. They can send commands to the actuators in the physical experimental setup through the game interface (e.g. to initialize or change the input parameters for the experiment), conduct the experimental procedure, collect the feedback from the sensors (i.e. the Kinect) and observe the real-time animation of the experiments [49]. This approach mimics a hands-on laboratory and gives the users a feel of immersion, allows them to observe the physical phenomenon in action and benefit from collaborative learning. The implementation of this concept is shown in the right part of Figure 4. The users can see a real-time animation (or choose to start the entire animation if the experiment is conducted in batch mode) of the pendulum experiment and are presented with a summary of the computed results in the form of a pop-up window.

5 Kinect-based Human-computer Interface

5.1 Synchronization of Human Body Motion and Avatar

The game-based VE chosen here is the massive multiplayer online game Garry's Mod (GMod), which is built on the "Source" game engine. It provides high quality human avatars, which can either serve as game player characters or non-player characters. The models of these avatars can be animated and are called "ragdolls" by the developer community. The term "ragdoll" indicates that, like dolls, these models have virtual bones that cannot bend and joints that allow connected bones to rotate relative to each other within certain limits. An avatar's gestures are actually animated by controlling the movement of the corresponding joints.

One of the most exciting features of the Kinect is its human skeleton tracking. Based on the acquired depth data, the machine learning algorithm that supports the Kinect is capable of extracting the positions of the human body's major joints at very high time rates and with promising accuracy [50]. Currently, the Kinect SDK supports the tracking of 20 body joints: 4 for each arm, 4 for each leg, 2 for the torso, 1 for the head and 1 for the neck. The position of each joint in 3-D skeleton coordinates (x, y, z), which form a Cartesian coordinate system with its origin located at the Kinect, can be derived in real time. Figure 6 depicts the skeleton as recognized by the Kinect and the skeleton of the physics model in GMod. A detailed description of their minor differences can be found elsewhere [51].

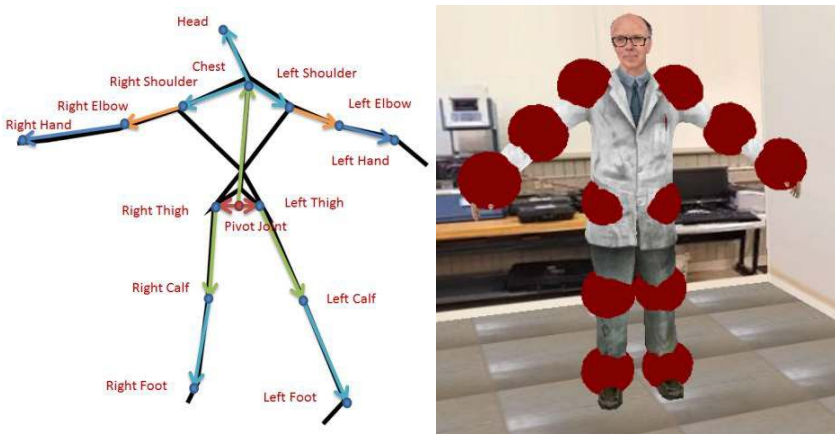

Fig. 6. Skeleton recognized by Kinect (left); corresponding skeleton in GMod (right)

Knowing the joint positions provided by the Kinect, animating an avatar so as to let it display the real-time motion of a human seems straightforward. After the Kinect acquires the 3-D joint positions of the human, they are transmitted to GMod by the shared memory approach [18]. However, there are still three problems that need to be solved in order to facilitate smooth and correct animations inside the game environment, namely the error caused by the differences between the Kinect's skeleton model and GMod's avatar model, the error caused by the differences between the frame rate of the Kinect and that of the game engine, and the communication delay between the Kinect SDK and the game engine.

Therefore, in order to animate an avatar, a better method is to specify the velocity of the selected physics model, as the game engine can automatically process the velocity and render the models at a high frame rate. The velocity of the avatar within the VE is defined as $\boldsymbol{v} = \frac{\boldsymbol{P}_t - \boldsymbol{P}_c}{\Delta t}$, where $\boldsymbol{P}_t$ is the target position acquired by the Kinect SDK from the actual person, $\boldsymbol{P}_c$ is the current position of the physics model inside the game engine and is the time between two Kinect frames. The solution to this challenge resulting from the differences between the Kinect skeleton and the avatar model is to normalize the bones' length to unity and convert the Kinect joint positions to bone orientations using Equation 4:

$$N = \frac{\boldsymbol{P}_{j1} - \boldsymbol{P}_{j2}}{\|\boldsymbol{P}_{j1} - \boldsymbol{P}_{j2}\|} \tag{4}$$

In this equation, the $\boldsymbol{P}_j$s are the positions of the two joints of a bone given by the Kinect SDK and the 3-D unit vector $\boldsymbol{N}$ indicates the orientation of the bone. The position of a joint in the game engine is defined as $\widetilde{\boldsymbol{P}}_{j2} = \widetilde{\boldsymbol{P}}_{j1} + \boldsymbol{L}_b\boldsymbol{N}$, where $\widetilde{\boldsymbol{P}}_{j2}$ is the position of the joint to be determined, $\widetilde{\boldsymbol{P}}_{j1}$ is the position of the known joint in the game engine and $\boldsymbol{L}_b$ is the length of the bone connected by these two joints.

In order to navigate an avatar inside the VE, it is necessary to differentiate between the world coordinate system and the avatar coordinate system. The world coordinate

system is fixed to the VE. It is defined by the VE map author and cannot be changed easily. In the case presented here, the avatar coordinate system is attached to the avatar, with the x-axis pointing to the right, the y-axis pointing up and the z-axis pointing to the front. Since left turns and right turns of the avatar model always occur about the vertical y-axis, a joint's position in the world coordinate system is obtained from its avatar coordinates by Equation 5:

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix} \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} + \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix} \tag{5}$$

In this equation, are the avatar coordinates of a joint's position, are the pivot point's world coordinates and is the direction that the avatar looks at in the world coordinate system.

5.2 Speech Recognition and Voice Commanding

The traditional navigation method supported by game engines is to use a keyboard and mouse for commanding. In the context of a virtual laboratory environment, using these devices in combination with the Kinect is impractical since the students are not able to carry a keyboard and a mouse while they conduct the experiments (e.g. mechanical assembly) using their body motion. Input using simple gestures or hand signals is a method for replacing these traditional input devices [52,53]. While this approach is acceptable for an application that only requires a few commands, for more complex experiments such as mechanical assembly training involving more than 10 commands, it is unlikely that the students could manage these gestures easily.

Using voice commands can greatly reduce the students' difficulties in getting started with the game-based VR platform. The Kinect contains an array of 4 microphones to collect voice data. These data can then be processed and recognized by the Kinect SDK and by the Microsoft Speech API. The speech recognition can be accomplished at the same time as the body tracking. In the prototype mechanical assembly platform presented here, there are currently 49 voice commands in 4 categories: 9 for navigation, 2 for assembly, 2 for GUI operation and 36 for pop-up menu operation [51].

In order to make the commanding process convenient and easy-to-use for students, a command usually includes several speeches whose semantics are close to the meaning of the command in natural language. For instance, the words "accelerate" and "faster" are both interpreted as "move faster". Thus, these two words are chosen for the speech of "accelerate". Since there are 4 categories of voice commands, the method to process these commands is more complicated than that for processing the body tracking. A simplified flowchart of the voice command processing is shown in Figure 7.

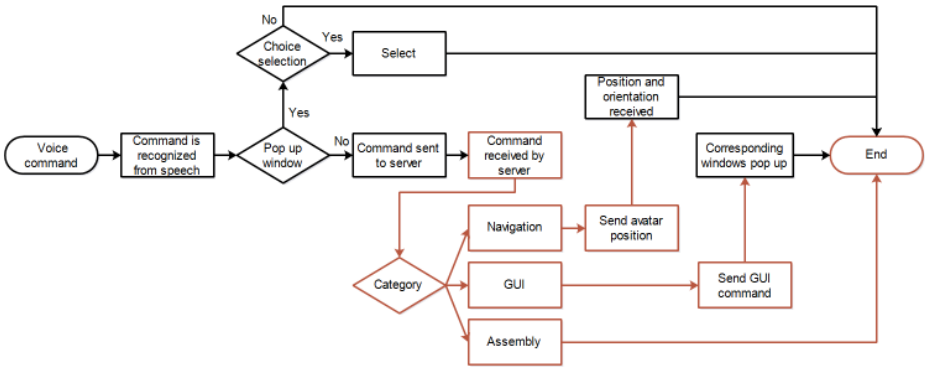

Fig. 7. Flow chart for voice command processing

5.3 Prototype Mechanical Assembly in Game-based VR

In the mechanical assembly platform presented in detail elsewhere [51], a part can be picked up and manipulated either by one hand or by both hands. Generally, for large parts, the two-hand approach is preferred while for small parts, one hand suffices. The picking positions within the parts are virtual handles for avatars to hold during the assembly. These picking positions are pre-defined in the part's local coordinates. When the physical model of an avatar's hand contacts a part, a picking-validation process is triggered [51]. If the validation is passed, a kinematic constraint is imposed at the place of the "handle" between the physics model and the part. For one-hand picking, the constraint imposed is a "welding" constraint, which removes all six relative degrees of freedom between the hand and the part. For two-hand picking, the constraint imposed is "ball socket", which removes only the three relative translational degrees of freedom. In this way, the avatar can rotate the part with flexibility by grasping the part with both hands. Figure 8 shows an avatar using both hands to pick up a gear in GMod.

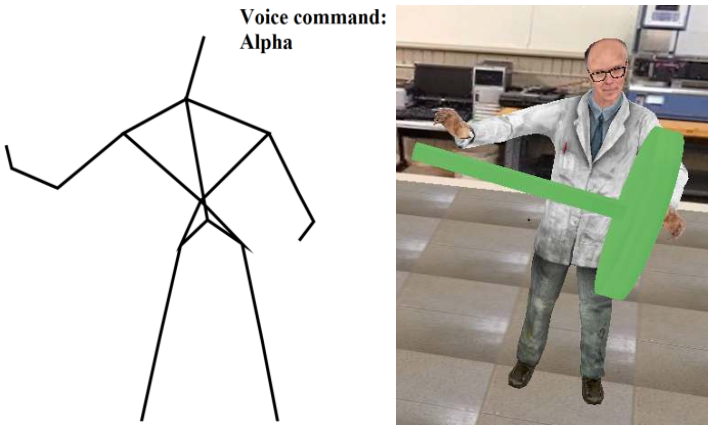

Fig. 8. Skeleton captured by Kinect (left); Avatar picking up a part with both hands (right)

In the mechanical assembly platform, an algorithm for versatile assembly representation was developed and implemented. The assemblies are described by topological models. This allows the VE to simulate the different assembly procedures in a unified fashion, thus reducing the educators' time required for designing pedagogically effective assembly exercises. More details of the implementation are described elsewhere [54].

In order to simplify the assembly platform, a validation process is triggered whenever a part held by an avatar approaches or touches another part. Based on this validation process, the system decides whether these two parts are suitable to be mated. If they have matching features that allow them to be connected, an auto-guidance process is triggered so that pre-defined constraints are imposed. At the same time, the avatar releases its hand(s), thus ceasing to grasp the part. After the assembly is completed, the system determines whether the new sub-assembly is eligible to be picked by the avatar. For instance, when a part of the sub-assembly is fixed to the ground, it cannot be picked up again. In some other cases, the sub-assembly can be picked up and manipulated again. The flowchart of the assembly procedure is shown in Figure 9.

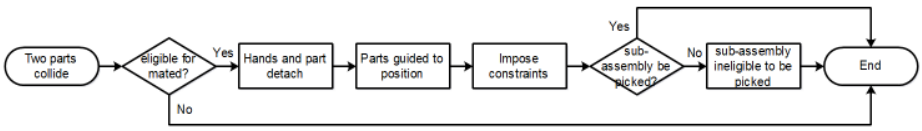

Fig. 9. Flowchart for assembly procedure

6 Benefits of Kinect-based VR Educational Laboratory

The traditional method for creating a VE that represents a real laboratory requires several steps, which include measuring the dimensions of the actual scene, taking pictures of the laboratory and then post-processing them, creating the map with a CAD software and then converting the map into formats that are compatible with the VR engine. These processes could easily take hours of work, even for an expert. In contrast, the only work required when using the method described here is to scan the real scene with a hand-held Kinect, and then the VE is created automatically. This procedure may take only about 30 to 60 minutes with the average of today's computer configurations. In addition to speeding up the process, this method is also less complicated and requires fewer tools. Therefore, any users even without any measuring and modeling experience can easily and quickly create their own maps for VR.

Traditional DAQ systems for obtaining 3-D range information consist of sensors, DAQ measurement hardware and a processor with software. This kind of DAQ system has to be modified and calibrated constantly for being used in various applications, thus imposing significant up-front costs that hamper its broad usage in educational laboratories. In contrast, using the Kinect as a substitute DAQ system can bring several benefits. It can meet some fundamental requirements desired by educational applications, such as a short time for getting familiarized with the use of the system,

ease of operation, the ability to add resources as the students' knowledge grows and robustness to prevent certain kinds of errors [55]. Furthermore, the ability to track the positions, velocities, accelerations and even deformations of multiple moving objects in three-dimensional space would make the Kinect superior to most of the existing DAQ systems currently used in educational laboratories.

Using the Kinect as human-computer interface for the VR environment, the students can use their bodies' motions to control fully animated avatars and complete tasks. They can also interact with computers by voice commands. Compared to simulations controlled by keyboard and mouse, simulations with the Kinect as the interface are more realistic. In addition, compared to experiment simulations based on immersive VR technology, the platform described here is more affordable.

7 Conclusions

In this paper, some background on VR, the Kinect and the usage of desktop VR in educational applications was provided. An innovative approach of using the Kinect as an essential component for developing game-based VR systems for engineering education was described, which addresses three different aspects of the development. First, a method for creating a VE automatically with a hand-held Kinect was discussed, which can dramatically improve the efficiency of VE creation. Second, a technique of using the Kinect as a substitute DAQ system that can track the motions of moving objects in educational laboratories was developed, which can simplify the hardware setup of the experiments. Finally, the Kinect was used to replace the traditional keyboard and mouse human-computer interface, in which the users can use their body motions and voice as input commands for the VR system. This interface is more realistic and affordable compared to traditional approaches. Prototype implementations that can validate each of these three techniques have all been developed and tested using a game-based VE. Using the methods proposed here, three major components of educational VR development, namely VE modeling, hardware interface implementation and human-computer interface development, can all be accomplished with a low cost and commercially available 3-D scanner, the Microsoft Kinect.

Future work will be focused on several aspects. First, the precision of the environment modeling and the quality of the models will be improved. Then, more tracking algorithms will be integrated into this DAQ software package. Last, studies on the usability of the new human-computer interface and its learning effectiveness will be conducted to identify the advantages and disadvantages of the platform presented here.

8 References

[1] Hawkins, D. G., Virtual reality and passive simulators: the future of fun. *Communication in the Age of Virtual Reality,* 159-189 (1995).

[2] Youngblut, C., Educational uses of virtual reality technology. *IDA Document No. D-2128*, Institute for Defense Analyses, Alexandria, VA (1998).
[3] Andolsek, D. L., Virtual reality in education and training. *International Journal of Instructional Media*, 22, **2**, 145-155 (1995).
[4] Costello, P. J., Health and safety issues associated with virtual reality: a review of current literature. *Advisory Group on Computer Graphics*, (1997).
[5] Dickey, M. D., Teaching in 3D: pedagogical affordances and constraints of 3D virtual worlds for synchronous distance learning. *Distance Education*, 24, **1**, 105-121 (2003). https://doi.org/10.1080/01587910303047
[6] Balamuralithara, B. and Woods, P. C., Virtual laboratories in engineering education: the simulation lab and remote lab. *Computer Applications in Engineering Education*, 17, **1**, 108-118 (2009). https://doi.org/10.1002/cae.20186
[7] Zollhöfer, M., Martinek, M., Greiner, G., Stamminger, M. and Süßmuth, J., Automatic reconstruction of personalized avatars from 3D face scans. *Computer Animation and Virtual Worlds*, 22, **2-3**, 195-202 (2011). https://doi.org/10.1002/cav.405
[8] Henry, P., Krainin, M., Herbst, E., Ren, X. and Fox, D., RGB-D mapping: using Kinect-style depth cameras for dense 3D modeling of indoor environments. *The International Journal of Robotics Research*, 31, **5**, 647-663 (2012). https://doi.org/10.1177/0278364911434148
[9] Zhang, Z., Zhang, M., Chang, Y., Aziz, E.-S., Esche, S. K. and Chassapis, C., Real-time 3D model reconstruction using Kinect for a game-based virtual laboratory. *Proceedings of the ASME International Mechanical Engineering Conference & Exposition IMECE'13*, San Diego, USA, 13-21, (2013). https://doi.org/10.1115/IMECE2013-64518
[10] Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D. and Davison, A., KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, Santa Barbara, USA, 559-568 (2011). https://doi.org/10.1145/2047196.2047270
[11] Herbst, E., Ren, X. and Fox, D., RGB-D flow: dense 3-D motion estimation using color and depth. *IEEE International Conference on Robotics and Automation*, Karlsruhe, German, 2276-2282 (2013).
[12] Hadfield, S. and Bowden, R., Kinecting the dots: particle based scene flow from depth sensors. *IEEE International Conference on Computer Vision (ICCV)*, Barcelona, Spain, 2290-2295 (2011). https://doi.org/10.1109/ICCV.2011.6126509
[13] Zhang, M., Zhang, Z., Aziz, E.-S., Esche, S. K. and Chassapis, C., Kinect-based universal range sensor for laboratory experiments. *Proceedings of the ASME International Mechanical Engineering Conference & Exposition IMECE'13*, San Diego, USA, (2013). https://doi.org/10.1115/IMECE2013-62979
[14] Zhang, M., Zhang, Z., Esche, S. K. and Chassapis, C., Universal range data acquisition for educational laboratories using Microsoft Kinect. *Proceedings of the 2013 ASEE Annual Conference & Exposition*, Atlanta, USA, 23-26 (2013).
[15] Lui, A. K., Ng, V. S. and Chan, C. H., *Gesture-based interaction for seamless coordination of presentation aides in lecture streaming*. Knowledge Sharing through Technology. Springer Berlin Heidelberg, 108-119 (2013).
[16] Jing, P. and Guan, Y., Human-computer interaction using pointing gesture based on an adaptive virtual touch screen. *International Journal of Signal Processing, Image Processing & Pattern Recognition*, 6, **4**, 81-92 (2013).
[17] Cheng, L., Sun, Q., Su, H., Cong Y. and Zhao, S., Design and implementation of human-robot interactive demonstration system based on Kinect. *Proceedings of 2012 24th Chinese*

Control and Decision Conference (CCDC), Taiyuan, China, 971-975 (2012). https://doi.org/10.1109/CCDC.2012.6242992

[18] Zhang, Z., Zhang, M., Tumkor, S., Chang, Y., Esche, S. K. and Chassapis, C., Integration of physical devices into game-based virtual reality. *International Journal of Online Engineering*, 9, **5**, 25-38 (2013). https://doi.org/10.3991/ijoe.v9i5.2705

[19] Cheung, G. K., Kanade, T., Bouguet, J. Y. and Holler, M., A real time system for robust 3D voxel reconstruction of human motions. *Proceedings of the 2000 IEEE Conference on Computer Vision and Pattern Recognition*, Hilton Head Island, USA, 714-720 (2000). https://doi.org/10.1109/CVPR.2000.854944

[20] Choi, J., Medioni, G., Lin, Y., Silva, L., Regina, O., Pamplona, M. and Faltemier, T. C., 3D face reconstruction using a single or multiple views. *Proceedings of 2010 International Conference on Pattern Recognition,* Istanbul, Turkey, 3959-3962 (2010). https://doi.org/10.1109/ICPR.2010.963

[21] Pollefeys, M., Nistér, D., Frahm, J. M., Akbarzadeh, A., Mordohai, P., Clipp, B., Engels, C., Gallup, D., Kim, S.-J., Merrell, P., Salmi, C., Sinha, S., Talton, B., Wang, L., Yang, Q., Stewénius, H., Yang, R., Welch, G. and Towles, H., Detailed real-time urban 3D reconstruction from video. *International Journal of Computer Vision*, 78, **2**, 143-167 (2008). https://doi.org/10.1007/s11263-007-0086-4

[22] http://www.3dsom.com, 9 April 2017.

[23] http://insight3d.sourceforge.net, 9 April 2017.

[24] http://reconstructme.net, 9 April 2017.

[25] https://msdn.microsoft.com/en-us/library/dn188670.aspx, 9 April 2017.

[26] Estrada, H. and Kim, R., Application of data acquisition systems in the undergraduate laboratories of mechanical engineering and engineering science. *Proceedings of the 27th Annual Frontiers in Education Conference*, Pittsburgh, USA, 454-460. (1997). https://doi.org/10.1109/FIE.1997.644921

[27] Tian, Z., Zhang, Y., Zhou, M., and Liu, Y., 2014, Pedestrian dead reckoning for MARG navigation using a smartphone. *EURASIP Journal on Advances in Signal Processing*, 2014, **1**, 65. https://doi.org/10.1186/1687-6180-2014-65

[28] Braasch, M. S. *Inertial Navigation Systems*. Aerospace Navigation Systems, (2016). https://doi.org/10.1002/9781119163060.ch1

[29] Guna, J., Jakus, G., Pogačnik, M., Tomažič, S., and Sodnik, J. An analysis of the precision and reliability of the leap motion sensor and its suitability for static and dynamic tracking. *Sensors*, 14, **2**, 3702-3720 (2014). https://doi.org/10.3390/s140203702

[30] Foxlin, E., Wormell, D., Browne, T. C., and Donfrancesco, M. Motion tracking system and method using camera and non-camera sensors. U.S. Patent 8,696,458, (2014).

[31] Jacobson, J. and Lewis, M., Game engine virtual reality with CaveUT. *Computer*, 38, **4**, 79-82 (2005). https://doi.org/10.1109/MC.2005.126

[32] Ausburn, L. J. and Ausburn, F. B., Desktop virtual reality: A powerful new technology for teaching and research in industrial teacher education. *Journal of Industrial Teacher Education*, 41, **4**, 1-16 (2004).

[33] Gregory, J., Engine differences across genres, *Proceedings in Game Engine Architecture*, 13-15 (2009).

[34] Chang, Y., Aziz, E.-S., Esche, S. K. and Chassapis, C., A framework for developing collaborative training environments for assembling. *Computers in Education Journal*, 23, **4**, 44-59 (2013).

[35] Chang, Y., Aziz, E.-S., Esche, S. K. and Chassapis, C., A multi-user virtual laboratory environment for gear train design. *Computer Applications in Engineering Education Journal*, 22, **4**, 788-802 (2013).

[36] Aziz, E.-S., Corter, J. E., Chang, Y., Esche, S. K. and Chassapis, C., Evaluation of the learning effectiveness of game-based and hands-on gear train laboratories. *Proceedings of the 42nd ASEE/IEEE Frontiers in Education Conference*, Seattle, USA, 1-6. (2012). https://doi.org/10.1109/FIE.2012.6462269

[37] Hoffman, A., Goetz, M., Vieth, M., Galle, P. R., Neurath, M. F. and Kiesslich, R., Confocal laser endomicroscopy: technical status and current indications. *Endoscopy*, 38, **12**, 1275-1283 (2006). https://doi.org/10.1055/s-2006-944813

[38] Khoshelham, K. and Elberink, S. O., Accuracy and resolution of Kinect depth data for indoor mapping applications. *Sensors*, 12, **2**, 1437-1454 (2012). https://doi.org/10.3390/s120201437

[39] Klein, G. and Murray, D., Parallel tracking and mapping for small AR workspaces. *Proceedings of 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, Washington, DC, USA, 225-234 (2007). https://doi.org/10.1109/ISMAR.2007.4538852

[40] Szeliski, R., *Computer vision: algorithms and applications*. Springer Science & Business Media. (2010).

[41] Devillers, O., Delaunay triangulations, theory vs. practice. *Proceedings of the 28th European Workshop on Computational Geometry*. Assisi, Italy, (2012).

[42] Zhang, Z., Zhang, M., Chang, Y., Esche, S. K. and Chassapis, C., An efficient method for creating virtual spaces for virtual reality. *Proceedings of the ASME International Mechanical Engineering Conference & Exposition IMECE'14*. Montreal, Canada, (2014). https://doi.org/10.1115/IMECE2014-37149

[43] https://developer.valvesoftware.com/wiki/Valve_Hammer_Editor, 9 April 2017.

[44] Zhang, M., Zhang, Z., Chang, Y., Esche, S. K. and Chassapis, C., Kinect-based universal range sensor and its application in educational laboratories. *International Journal of Online Engineering*, 11, **2**, 26-35 (2015). https://doi.org/10.3991/ijoe.v11i2.4299

[45] Zhang, M., Zhang, Z., Esche, S. K. and Chassapis, C., Algorithm modification approach to improve the Kinect's performance in point cloud processing. *Proceedings of the ASME International Mechanical Engineering Conference & Exposition IMECE'14*, Montreal, Canada, (2014). https://doi.org/10.1115/IMECE2014-37064

[46] Hough, P. V., Method and means for recognizing complex patterns. US Patent, No. 3069654, (1962).

[47] Fischler, M. A. and Bolles, R. C., Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24, **6**, 381-395 (1981). https://doi.org/10.1145/358669.358692

[48] Schnabel, R., Wahl, R. and Klein, R., Efficient RANSAC for point-cloud shape detection. *Computer Graphics Forum*, 26, **2**, 214-226 (2007). https://doi.org/10.1111/j.1467-8659.2007.01016.x

[49] Tumkor, S., Zhang, Z., Zhang, M., Chang, Y., Esche, S. K. and Chassapis, C, Integration of a real-time remote experiment into a multi-player game laboratory environment. *Proceedings of the ASME International Mechanical Engineering Conference & Exposition IMECE'12*, Houston, USA, 181-190 (2012). https://doi.org/10.1115/IMECE2012-86944

[50] Shotton, J., Sharp, T., Kipman, A., Fitzgibbon, A., Finocchio, M., Blake, A., Cook, M. and Moore, R., Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56, **1**, 116-124 (2013). https://doi.org/10.1145/2398356.2398381

[51] Chang, Y., Esche, S. K. and Chassapis, C., A platform for mechanical assembly education using the Microsoft Kinect. *Proceedings of the ASME International Mechanical Engineering Conference & Exposition IMECE'14*, Montreal, Canada, (2014). https://doi.org/10.1115/IMECE2014-38606

[52] Francese, R., Passero, I. and Tortora, G., Wiimote and Kinect: gestural user interfaces add a natural third dimension to HCI. *Proceedings of the International Working Conference on Advanced Visual Interfaces*, Naples, Italy, 116-123 (2012). https://doi.org/10.1145/2254556.2254580

[53] Boulos, M. N. K., Blanchard, B. J., Walker, C., Montero, J., Tripathy, A. and Gutierrez-Osuna, R., Web GIS in practice X: a Microsoft Kinect natural user interface for Google Earth navigation. *International Journal of Health Geographics*, 10, **1**, 1-14 (2011).

[54] Aziz, E.-S., Chang, Y., Esche, S. K. and Chassapis, C., Capturing assembly constraints of experimental setups in a virtual laboratory environment. *Proceedings of the ASME International Mechanical Engineering Conference & Exposition*, Houston, USA, 191-200 (2012). https://doi.org/10.1115/IMECE2012-87828

[55] Yáñez, J., Quintana, D., Quintáns, C., Fariña, J. and Rodríguez-Andina, J., FPGA-based system for the education in data acquisition and signal generation. *Proceedings of the 31st Annual Conference of IEEE Industrial Electronics Society*, Raleigh, USA, 2168-2173 (2005).

9 Authors

Mingshao Zhang is with Southern Illinois University Edwardsville, IL, USA (mzhang@siue.edu).

Zhou Zhang is with New York City College of Technology, New York, NY, USA.

Yizhe Chang, El-Sayed Aziz, Sven Esche, and **Constantin Chassapis** are with Stevens Institute of Technology, Hoboken, NJ, USA.

Article submitted 28 September 2017. Published as resubmitted by the authors 05 December 2017.