ABSTRACT

Virtual Environment (VE) system offers a natural and intelligent user interface. Hand gesture recognition is more efficient and easier interaction in VE than human-computer interface (HCI) devices like keyboards and mouse. We propose a hand gesture recognition interface that generates commands to control objects directly in a game. Our novel hand gesture recognition system utilizes both Bag-of features and Support Vector Machine (SVM) to realize user-friendly interaction between human and computers. The HCI based on hand gesture recognition interacts with objects in a 3D virtual environment. With this interface, the user can control and direct a helicopter by a set of hand gesture commands controlling the movements of the helicopter. Our system shows the hand gesture recognition interface can attain an enhanced and more intuitive and flexible interaction for the user than other HCI devices.

INTRODUCTION

Virtual environments (VE) give three-dimensional (3D) environment in which a person is able to interact directly with virtual objects naturally and efficiently in the environment. The methods to interact with VE using hand gestures are two types. First type is to generate an event or command for functional interaction. The other is to control object directly. Several systems exploit hand gesture recognition to generate commands like our system. While some of the others employ 3D menu with 3D-cursor to generate command in 3D environment [1].

Devices that detect object location and direction, speech and sound, facial expression, hand gestures, haptic response and other features of human actions can be used for interactions between humans and VEs. Immersive and natural Human-Computer Interfaces (HCI) for systems in 3D VE become promising by using these devices and approaches [2, 3, 4].

In the past, study on man-machine interaction has been confined to methods that make use of graphic display, a keyboard and a mouse. However, this concept has been changed recently. Approaches like vision, sound, speech recognition, projective displays and context aware devices permit for a much richer, multimodal interaction between human and computer. Gestures are helpful for machine interaction because they are the most important and meaningful type of human communication. In our approach, we will deal with man-machine interaction using the bare hand. The position and direction of the hand gesture will be used to control our application directly.

Gestural interfaces are continuously investigated since the work in [5]. Gestural interfaces opened up largely by using of the Wiimote [6]. Gesture interfaces for gaming that used hand/body gesture technology must be aimed to attain social and commercial achievement. Most research on the area of gestural interaction concentrated on algorithms and robust recognition of gestures [6, 7, 8]. However, gestural interfaces have to satisfy the same requirements like the other interaction methods [40].

HCIs in the future will allow more natural and intelligent interaction between human and all types of sensor-based devices. Development in the human-computer interaction field has introduced novel technologies that allow users to interact with computer systems in increasingly natural and intuitive ways; applications implementing them demonstrate increased effectiveness, speed, power, and realism. However, users used to usual interaction methods like mouse and keyboards are most likely reluctant to accept new, alternative interfaces. Usually, new alternative interfaces have to be more accessible with short periods of learning and adjustment. They have to offer more natural human-machine interaction. There are a lot of researches on hand-gesture interfaces over the past 30 years in multiple application fields. There are four types of these applications, which are human-robot interaction, entertainment, medical systems and assistive technologies, and crisis management and disaster relief. Any hand gesture recognition approach will not work properly for every application because every hand gesture recognition technique is affected by environment, application domain, and human cultural background [41].

Hand-gesture recognition applications have three main advantages over conventional human computer interaction systems: First, retrieving information while keeping total sterility. Touchless interfaces are particularly helpful in healthcare environments; Second, helping physical handicaps. Interact with home devices and appliances for users with physical handicaps and/or elderly people with disability to move; finally, navigating large data. Navigation of big complex data volumes and manipulation of high-quality images through intuitive actions benefit from 3D interaction, rather than restricted traditional 2D techniques [41].

This is organized as follows: section two introduces related works; the third section describes our 3D gaming VE; section four describes our system in details, including the training stage to build the cluster and the SVM classifier models and the testing stage for recognition; section five provides experimental results; section six explains how our system generates gesture commands; section seven describes the interaction with our 3D gaming VE by hand gestures; the last section gives the conclusion of our method.

3D VIRTUAL GAME ENVIRONMENT

The game “Fly over the city” presented in this paper is developed using Java 3D. Java 3D is a client-side API which was developed at Sun Microsystems. It uses java programming language to render interactive 3D graphics. Java 3D aims to enable quick development of complex 3D applications. It also enables fast and efficient implementation on a variety of platforms such as PCs, and workstations. Java 3D provides rich libraries for building shapes, describing behaviors, interacting with the use, and controlling rendering details. It allows software developer to build 3D scenes programmatically, or through loading 3D content from VRML, OBJ and other external files.

The game “Fly over the city” is composed of different objects that represent different parts of the city such as high buildings, town homes, stadium, markets, city hall, streets, and airport as shown in Figure 1. These objects are all static, and have been imported from Alice v2.0 libraries.

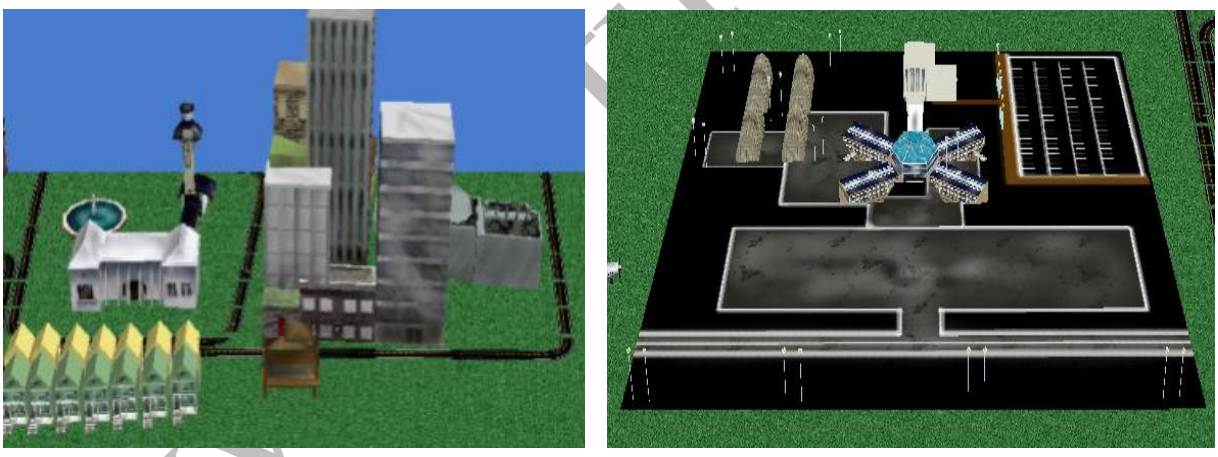

Figure 1: Part of the city

Dynamic objects such as people and birds perform certain behaviors, where people walk in the streets and birds flies in the sky. The main object in this game is a helicopter as shown in Figure 2(a). Creating the Helicopter object require to define its geometry, components, and Sound. The game starts when the helicopter departs the airport as shown in Figure 2(b). The player uses the 4 arrows keys in keyboard to fly helicopter. To move up/down the helicopter, the user simply presses the up/down arrow key accordingly. To move right/left the helicopter, the

user just presses the right/left arrow key accordingly. The user can also speed up or slow down helicopter simply by pressing the letter key A/Z on the keyboard accordingly. If the user is willing to hold helicopter in the sky, she/he does not press any keys on the keyboard. To fly it again, simply she/he presses any of the 4 arrows keys in keyboard.

Figure 2: (a). Helicopter (b). Flying over the city

While flying the helicopter might collide with other objects such as birds, buildings, and High-rise buildings. A function called collision is used to address the collision. When the helicopter is about to collide with the building, it will move backward and when it is about to collide with a bird, it will move down. Figure 3 describes the state machine diagram for the helicopter.

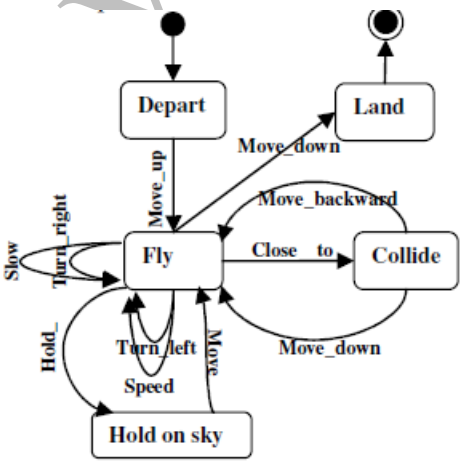

Figure 3. State machine diagram

HAND GESTURE RECOGNITION SYSTEM OVERVIEW

Our hand gesture recognition system consists of two stages: the offline training and the online testing. The cluster and multi-class SVM classifier models will be built in the training stage and will be used in the testing stage to recognize hand gestures captured from a webcam.

Training Stage

The training stage model is shown in Figure 4. Before building the bag-of-features model, we captured 100 training images for each hand gesture, which are the fist, index, palm, and little finger gestures, for different people, scales, and rotations and under different illuminations conditions to increase the robustness of the multi-class SVM classifier and the cluster model. The system can also recognize any other gestures, such as two, three, and five. All the training images illustrate the hand gestures without any other objects and the background has no texture or objects (white wall). In this way, we guarantee that all the key points extracted from training images using the SIFT algorithm will represent the hand gesture only. Processing time for extracting key points using SIFT will be reduced when the image resolution is reduced and converted into portable gray map (PGM) format. Therefore, SIFT is real time performance for low resolution PGM images. The size of training images have been reduced to 50×50 pixels and converted into PGM format to coincide with the size of the small image (50×50 pixels) that contains the detect hand gesture only for every frame captured from the video file in the testing stage.

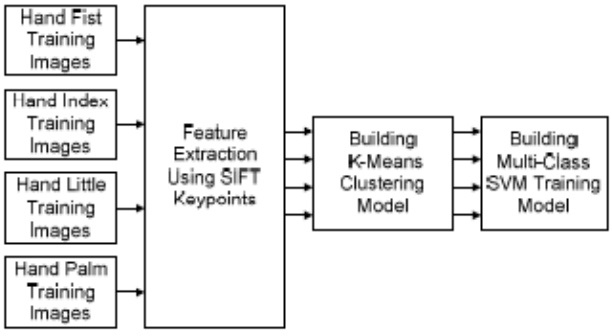

Figure 4. Training stage.

The bag of features model is built using features extraction, learning a "visual vocabulary" by k-means clustering, quantizing features using visual vocabulary and finally representing images by frequencies of "visual words", as will be discussed in the following:

Sift Algorithm

We used the SIFT algorithm to extract the key points (vectors) for each training image. Figure 5 shows some training images with their key points. The number of key points decreases when the hand gets away from the camera and increases when the hand comes closer to the camera, because the area of the hand increases. For the same distance from the camera, we notice that the palm gesture has the maximum number of key points as it has the largest area. We can increase the number of training images to train the system as we wish for all the hand gestures for different people with different scales, orientations and illumination conditions. The more training images used with different illumination conditions, the more accurate in building k-means cluster and SVM models since extracted features for training images using SIFT are invariant to scale, orientation and partially to illumination changes. Therefore, the time will increase for building the cluster model in the training stage. However, this will not affect the testing stage speed.

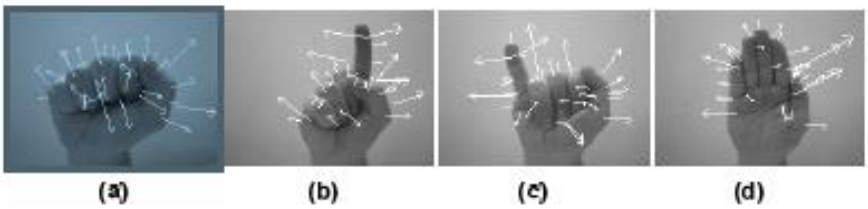

Figure 5. Training images (a) fist with 23 features. (b) index with 31 features. (c) little finger with 27 features. (d) palm with 48 features

K-means Clustering

Among many different types of clustering, in this paper, we used the k-means Clustering algorithm [36]. The number of the clusters (codebook size) will be determined depending on the structure of the data. There will be a sort of compromise for how to choose vocabulary size or number of clusters. If it is too small, then each bag of words vector will not represent all the key points extracted from its related image. If it is too large, then there will be quantization artifacts

and over fitting because of insufficient samples of the key points extracted from the training image.

In the training stage, when the training images contain only hand gestures on a white background, the key points that are extracted will represent the hand gesture only and this will not exceed 75 key points for the palm gesture, which has the largest number of key points. From this information, we know that the number of clusters must be larger than 75. Therefore, the training stage provides the minimum number of clusters that we can use. In the testing stage the webcam camera will capture other objects besides the hand gesture such as the face and background. The key points will be around 400 key points of all the objects in the image. We choose the value 750 as the number of clusters (visual vocabularies or codebook) to build our cluster model.

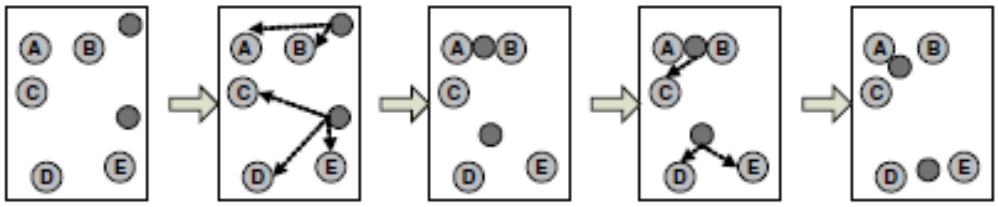

Figure 6. K-means clustering with two clusters.

The first step in k-means clustering is to divide the vector space (128-dimensional feature vector) into k clusters. K-means clustering starts with k randomly located centroids (points in space that represent the center of the cluster), and assigns every key point to the nearest one. After the assignment, the centroids (code-vectors) are shifted to the average location of all the key points assigned to them, and the assignments are redone. This procedure repeats until the assignments stop changing. Figure 6 shows this process in action for five key points: A, B, C, D, and E and two clusters.

Once this is done, each feature vector (key point) is assigned to one and only one cluster center that is in the nearest distance with respect to the Euclidean metric in 128 dimensional feature vectors. The key points that are assigned to the same cluster center will be in the same subgroup so that after clustering we have k disjoint subgroups of key points. Therefore k-means

clustering decreases dimensionality for every training image with n key points (n × 128) to 1 × k, where k is number of clusters.

Key points of each training image will be fed into the k-means clustering model to reduce its dimensionality into one bag-of-words vector with components equal to the number of clusters (k). In this way, each key point, extracted from a training image, will be represented by one component in the generated bag-of-words vector with value equal to the index of the centroid in the cluster model with the nearest Euclidean distance. The generated bag-of-words vector, which represents the training image, will be grouped with all the generated vectors of other training images that have the same hand gesture and labeled with the same number and this label will represent the class number. For example, label or class 1 for the fist training images, class 2 for index training images, class 3 for little training images and class 4 for palm training images.

Building the Training Classifier Using Multi-Class SVM

After mapping all the key points that represent every training image with its generated bag-of-words vector using k-means clustering, we fed every bag-of-words vector with its related class or label number into a multi-class SVM classifier to build the multi-class SVM training classifier model.

Testing Stage

Figure 7 shows the testing stage by using face detection and subtraction and hand gesture detection. After capturing frames from webcam or video file, we detected the face and subtracted it before using a skin detection and hand postures contours comparison algorithm because the skin detection will detect the face and the face's contours very close to the fist hand gesture contours. To get rid of other skin like objects existing in the image, we make sure that the contours of the detected skin area comply with the contours of any hand gestures contours to detect and save the hand gesture only in a small image (50×50 pixels). Then, the key points were extracted from the small image that contains the hand gesture only and will be fed into the cluster model to map them into a "bag of words" vector and finally this vector will be fed into multi-class SVM training classifier model to recognize the hand gesture.

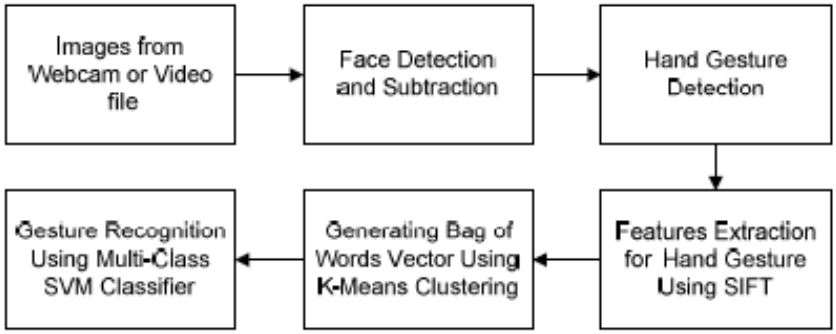

Figure 7. Testing stage.

Face Subtraction

We used the skin detection and contours comparison algorithm to detect the hand gesture and this algorithm can also detect the face because the face has a skin color and its contours like the hand fist gesture contours. To get rid of face area, we detected the face using Viola and Jones method [26] and then subtracted the face before applying the skin detection algorithm to detect the hand gesture only by replacing face area with a black circle for every frame captured. The Viola and Jones algorithm has a real time performance and achieving accuracy as the best published results [26]. It needs around 0.023 second/frame to detect the face.

First, we loaded a statistical model, which is the XML file classifier for frontal faces provided by OpenCV to detect the faces from the frames captured from a webcam during the testing stage. Once the face had been detected by the XML file classifier for every frame captured, we replaced the detected face area with a black circle to remove the face from the skin area. In this way, we make sure that the skin detection will be for hand gesture only as shown in Figure 8.

Figure 8. Face detection and subtraction.

Hand Gesture Detection

Detecting and tracking human hand in a cluttered background will enhance the performance of hand gesture recognition using bag-of features model in terms accuracy and speed because the key points extracted will represent the hand gesture only. Besides, we will not be confined with the frame resolution size captured from a webcam or video file, because we will always extract the key points of the small image (50×50 pixels) that contains the detected hand gesture area only not the complete frame. In this way the speed and accuracy of recognition will be the same for any frame size captured from a webcam such as 640×480, 320×240 or 160×120 and the system will be also robust against cluttered background because we process the detected hand gesture area only. The small image size (50×50 pixels) that contains the detected hand gesture area only has to be complied with the training images size of training stage.

Figure 9. Templates of hand gestures.

For detecting hand gesture using skin detection, there are different methods including skin color based methods. In our case, after detecting and subtracting the face, skin detection and contours comparison algorithm was used to search for the human hands and discard other skin colored objects for every frame captured from a webcam or video file. Before capturing the frames from a webcam, we loaded four templates of hand gestures as shown in Figure 9: fist, index, little and palm to extract their contours and saved them for comparison with the contours of skin detected area of every frame captured. After detecting skin area for every frame captured, we used contours comparison of that area with the loaded hand gestures contours to get rid of other skin like objects exist in the image. If the contours comparison of skin detected area complies with any one of the stored hand gestures contours, a small image (50×50 pixels) will enclose the hand gesture area only and that small image will be used for extracting the key points using SIFT algorithm.

In our implementation we used the hue, saturation, value (HSV) color model for skin detection since it has shown to be one of the most adapted to skin-color detection [37]. It is also compatible with the human color perception. Besides, it has real time performance and robust against rotations, scaling and lighting conditions and can tolerate occlusion well.

From a classification approach, skin-color detection can be considered as a two class problem: skin-pixel vs. non-skin-pixel classification. There are many classification techniques such as thresholding, Gaussian classifier, and multilayer perceptron [38]. We used the thresholding method, which has the least time on computation compared with other techniques and this is required for real time application. The basis of thresholding classification is to find the range of two components H and S in HSV model as we discarded the Value (V) component. Usually a pixel can be viewed as being a skin-pixel when the following threshold ranges are simultaneous satisfied: $0° < H < 20°$ and $75 < S < 190$.

Once the skin area had been detected, we found contours of the detected area and then compared them with the contours of the hand gestures templates.

If the contours of the skin area comply with any of the contours of the hand gestures templates, then that area will be the region of interest by enclosing the detected hand gesture with a rectangle, which will be used in tracking the hand movements and saving hand gesture in a small image (50×50 pixels) for every frame captured as shown in Figure 10. The small image will be used in extracting the key points to recognize hand gesture.

Figure 10. Small images of detected hand gestures.

Hand Gesture Recognition

We converted the small image (50×50 pixels) that contains the detected hand gesture only for every frame captured into a PGM format to reduce the time needed in extracting the key points. For every small PGM image, we extracted the key points (vectors) using the SIFT algorithm. The key points will be fed into the k-means clustering model that was built in the training stage to map all the key points with one generated vector (Bag of Words) with components (k), which is equal to the number of clusters (k) used in the training stage. Each feature vector in the key points will be represented by one component in the generated vector with value equals to the index of centroid in the cluster model with nearest Euclidean distance. Finally, the generated bag-of-words vector will be fed into the multi-class SVM training classifier model that was built in the training stage to classify and recognize the hand gesture.

EXPERIMENTAL RESULTS

We tested four hand gestures: the first gesture, the index gesture, the little finger gesture and the palm gesture. The camera used for recording video files in our experiment was a low-cost Logitech QuickCam web-camera that provides video capture with different resolutions such as 640×480, 320×240 and 160×120, at 15 frames-per second, which is adequate for real time speed image recognition.

Ten video files with the resolution of 640×480 had been recorded for each hand gesture: fist, index, little finger and palm using the Logitech webcam. The length of each video file was 100 images. The hand gestures were recorded with different scales, rotations and illuminations conditions and with a cluttered background. The test was run for the forty video files to evaluate the performance of the multiclass SVM classifier model for each gesture. Hand gesture recognition will start from the first frame captured. The trained multi-class SVM classifier shows a certain degree of robustness against scale, rotation, illumination and cluttered background.

Table 1 shows the performance of the multi-class SVM classifier for each gesture with testing against scale, rotation, illumination and cluttered background. The recognition time will not be affected with cluttered background or increasing the resolution of video file because the key points will be extracted using SIFT from the small image (50×50 pixels) that contains the detected hand gesture only. The recognition time for every frame captured is 0.017 second, which includes frame capture, hand detection, extracting key points and classification. The total time needs for detecting face and hand gesture detection and recognition will be 0.04 seconds which satisfies the real time performance.

Table 1. The performance of the multi-class SVM classifier

Gesture Name	Number of frames	Correct	Incorrect	Recognition Time (Second/frame)
Fist	1000	967	33	0.017
Index	1000	990	10	0.017
Little	1000	956	44	0.017
Palm	1000	994	6	0.017

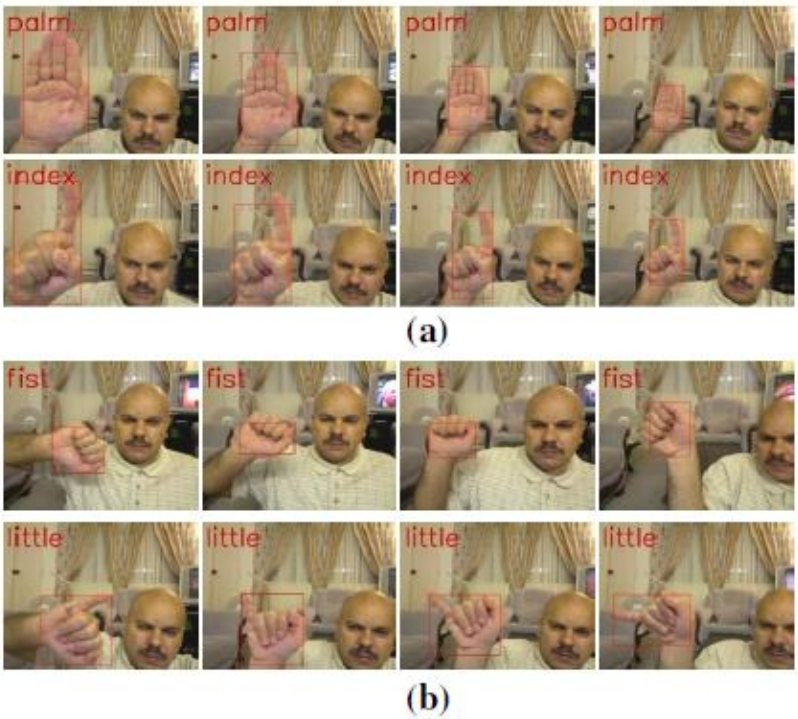

Figure 11. Hand gesture detection and recognition with cluttered background against: (a) scale. (b) rotation.

Figure 11 shows some correct samples for hand gesture recognition using multi-class SVM classifier for fist, index, little and palm gestures with testing against scale, rotation, illumination and cluttered background.

BUILDING GESTURE COMMANDS

A hand gesture is an action, which consist of a sequence of hand postures. The rules for the composition of hand postures into various hand gestures can be determined by a grammar [39]. We built a grammar for our system that generates 40 gesture commands, which can be used to control or interact with an application or a videogame instead of keyboard or mouse, by sending events to be executed such as double click, close, open, go left, right, up, or down and so on. Those gesture commands can be generated by three ways. First, by observing the gesture transitions from posture to posture, such as from fist to index, fist to palm, fist to little or stay fist and so on for the other postures. For each posture we have four transitions to other states, thus, we have 16 events or gesture commands can be sent from all of the transitions among of the four postures. Second, by observing the direction of movement for each posture: up, down, left or right. We have four directions or gesture commands for each posture or 16 gesture commands for four postures. We did not take the case of no movement for each posture as it is counted already in the first way when the transition occurred from fist to fist or palm to palm and so on. Finally, by observing the hand posture size or scale: when it comes close (zoom in) or far away (zoom out) from camera.

As we have two cases for each posture, we have eight gesture commands for the four postures. Therefore, from the three ways, we have totally 40 gesture commands can be sent to interact with an application or video game. In the following sections, we will explain how our system can generate gesture commands using three ways.

Transitions among Postures

This way depends on saving every two postures recognized from every two consecutive frames of a video sequence in a two states of a queue: the new state of the queue, which holds the current or new posture recognized from a video or webcam and the old state of the queue, which holds the previous or old recognized posture. The first posture recognized from first frame will be saved in the two states of the queue because they are empty. The next recognized posture will be saved in the new state after transferring its posture state to the old state. Thus for every

frame captured, the posture state of the old state is emptied. Then, the posture state of the new state is transferred to the old state. Finally, the current posture recognized from the frame will be saved in the new state. By observing the two states for every two consecutive frames captured, the system will keep monitoring the transitions among every two consecutive recognized postures and generates a specific gesture command for a specific transition among recognized postures. Figure 12 shows all the transition states of fist posture with all other postures.

Figure 12. Transitions among postures.

Movement Direction for Each Posture

This way depends on tracking the movement direction of the detected posture using the rectangle, which captures the detected hand posture only, and the transition of recognized posture still the same such as palm to palm. Once the hand posture is detected from each frame by the rectangle and the transition of recognized posture still the same, the coordinates X and Y point, which located in the middle of the rectangle, is recorded. The system will always monitor the absolute difference of distance between the two points of rectangle in the X and Y coordinates for every two successive frames that have the same posture. If the absolute difference of distance in X direction is larger than absolute difference of distance in Y direction, then the hand posture is moved left or right. If the difference of distance in X direction is positive, then the hand posture is moved right and if it is negative, then the hand posture is moved left. If the absolute difference of distance in Y direction is larger than absolute difference of distance in X direction, then the hand posture is moved up or down. If the difference of distance in Y direction is positive, then the hand posture. Is moved down and if it is negative,

then the hand posture is moved up. Figure 13 shows all the movement direction cases of palm posture.

Figure 13. Movement direction cases of palm posture.

Distance from Camera for Each Posture

This way depends on tracking the size of height for the rectangle, which captures the detected hand posture only, and the transition of recognized posture still the same. Once the hand posture is detected from each frame by the rectangle and the transition of recognized posture still the same such as little to little, the height of the rectangle is recorded. The system will always monitor the difference between the heights of two rectangles for every two successive frames that have the same posture. If the difference of rectangle height between the new frame and the previous frame is positive, then the posture gets closer to camera (zoom in) and if the difference is negative, then the posture gets away from camera (zoom out). Figure 14 shows all the zoom cases of little posture.

Figure 14. Zoom cases of little posture.

INTERACTION WITH 3D GAMING VE BY HAND GESTURES

To show the effectiveness of our gesture recognition system for human-computer interaction, our application of gesture-based interaction was tested with a 3D gaming VE by sending events to the keyboard. With this system, the user can navigate the 3D gaming world by flying the helicopter with a set of hand gestures by generating events or commands to control and direct its movements. Instead of using static hand postures to control 3D gaming VE, we develop our system into dynamic hand gesture to generate gesture command set in order to make the user experience natural and intuitive.

The C# was used to integrate our DLL file generated from C-based gesture recognition component. With this framework, the C# methods can call applications and libraries written in C/C++. The program in C# was used to send events to the keyboard using the SendKeys.SendWait method. Those events or commands are generated using our hand gestures.

The user, who is characterized by the avatar helicopter, can explore through the 3D gaming VE to navigate the virtual world and interact with objects. A good exploration and interaction interface to the 3D gaming VE should be natural and intelligent. In the meantime, the hand gesture interface should also be easy to learn and effective to permit the user to execute the actions. The interactions should be direct and accurate in order that the user and the VE are in concurrence on what objects is being interacted with.

Since the user is characterized by the avatar helicopter in the VE, we used our vision-based hand gesture recognition system to explore the helicopter by a set of hand gesture commands. In order to use hand gestures as an HCI device for VE applications, the hand gesture recognition system must satisfy the requirements in terms of real-time, accuracy, and robustness. The avatar helicopter can be moved forward/backward and turn left/right. Besides the helicopter can be hold in the sky and resumed flying and it can be speed up or slow down. To implement these functions, we map these commands by integrating the recognized hand gestures and hand motions directions according to Table 2. These gesture commands take into account the intuitiveness for the user to explore the 3D gaming VE. To move up/down the helicopter, the

user simply moves his palm up/down accordingly in front of the camera. To move right/left the helicopter, the user just moves her/his palm right/left accordingly. The user can also speed up or slow down helicopter simply by moving her/his palm forth/back accordingly in front of the camera. If the user is willing to hold helicopter in the sky, she/he changes the hand posture from palm to fist. To fly it again to any direction, simply she/he changes the hand posture from fist to palm and then moves her/his palm to that direction. It can also be zoomed in/out the screen by moving her/his fist back and forth. The screen will be zoomed in if the scale of the fist becomes bigger and vice versa.

Table 2. Table captions should be placed above the table

Current State	Next State	Direction	Send Key	Event
palm	Palm	Up	{UP}	Move Up
Palm	Palm	Down	{DOWN}	Move Down
Palm	Palm	Right	{RIGHT}	Move Right
Palm	Palm	Left	{LEFT}	Move Left
Palm	Palm	Forth	A	Speed Up
Palm	Palm	Back	Z	Slow Down
Palm	Fist	Hold	H	Hold
Fist	Palm	Hold	Nothing	Hold
Fist	Fist	Forth	^{+}	Zoom In
Fist	Fist	Back	^{-}	Zoom Out

We have two independent applications that can be run at the same time, which are helicopter game and our hand gesture detection and recognition system. The helicopter game can be run and controlled by keyboard without any relation with the webcam. We first ran the 3D gaming virtual environment in which we can control the movements of helicopter using keyboard. Then, we ran our hand gesture recognition system independently to send events or commands to the keyboard to control movements of helicopter. We tested the exploration of the VE with the defined hand gesture commands. Our system shows the gesture-based interface attain an improved interaction. Compared with controlling the avatar helicopter with the arrow keys on the keyboard, it is more intuitive and comfortable for users to use hand gestures to direct the movement of the avatar helicopter.

CONCLUSION

We proposed how a game running on a computer can be controlled with a bare hand using our approach to hand gesture recognition which can be utilized in natural interaction between human and computers. A novel vision-based hand gesture interface for interacting with objects in a 3D VE is designed and implemented. With the vision-based hand gesture interface, the user is able to navigate the 3D VE and access the objects by controlling the movements of the helicopter using a set of hand gesture commands. The system combines Bag-of-features and SVM to achieve real-time hand gesture recognition with high accuracy. Compared with traditional HCI devices such as keyboards and mouse, this vision-based hand gesture interface provides more natural and quick movement and entertainment for the user.

REFERENCES

[1] Dong A. Bowman et. al, "The Virtual Venue: User-Computer Interaction in Information-Rich Virtual Environments," *Presence*, Vol. 7, No. 5, pp. 478-493, 1998.

[2] V. Pavlovic, R. Sharma, and T. Huang. 'Gestural interface to a visual computing environment for molecular biologists', *Proc. Second International Conference on Atuomatic Face and Gesture Recognition*, Killington, Vermont, USA, pp.30–35, 1996.

[3] T. Kirishima, K. Sato, and K. Chihara. 'Real-time gesture recognition by learning and selective control of visual interest points', *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 27, No. 3, pp.351–364, 2005.

[4] Y. Wu, and T. Huang. 'Hand modeling analysis and recognition for vision-based human computer interaction', *IEEE Signal Processing Magazine, Special Issue on Immersive Interactive Technology*, Vol. 18, No. 3, pp.51–60, 2001.

[5] R. Bolt. "Put-that-there": Voice and gesture at the graphics interface. In Proc. SIGGRAPH, 1980.

[6] T. Schlomer, B. Poppinga, N. Henze, and S. Boll. Gesture recognition with a wii controller. In Proceedings of TEI, 2008.

[7] R. Liang and M. Ouhyoung. A Real-Time Continuous Gesture Recognition System for Sign Language. Proc. FG, 1998.

[8] T. B. Moeslund, M. St orring, and E. Granum. A natural interface to a virtual environment through computer visionestimated pointing gestures. Gesture and Sign Language in Human-Computer Interaction, pages 59{63, 2002.

[9] S. Mitra and T. Acharya. Gesture Recognition: A Survey. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 37(3), 2007, pp. 311-324.

[10] Q. Chen, A. El-Sawah, C. Joslin, and N.D. Georganas. 'A dynamic gesture interface forVE based on hidden Markov models', *Proc. IEEE International Workshop on Haptic, Audio and Visual Environments and their Applications (HAVE2005)*, pp.110–115, 2005.

[11] T. Metais, and N.D. Georganas. 'A glove gesture interface', *Proc. Bienneal Symposium on Communication (23rd)*, Kingston, Ontario, Canada, 2004.

[12] J. Yang, Y. Xu, and C.S. Chen. 'Gesture interface: modeling and learning', *Proc. IEEE International Conference on Robotics and Automation*, Vol. 2, San Diego, CA, USA, pp.1747–1752, 19994.

[13] M. Oskoei and H. Hu. Myoelectric control systems— A survey. *Biomedical Signal Processing and Control*, Vol. 2(4), 2007, pp. 275-294.

[14] E. Costanza, S.A. Inverso, and R. Allen. Toward subtle intimate interfaces for mobile devices using an EMG controller. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, Portland, Oregon, USA, April 02-07, 2005, pp. 481-489.

[15] C. Joslin, A. El-Sawah, Q. Chen. and N.D. Georganas. 'Dynamic gesture recognition', *Proc. IEEE Instrumentation and Measurement Technology Conference (IMTC2005)*, Ottawa, Canada, 2005.

[16] C. Keskin, A. Erkan, and L. Akarun. 'Real time hand tracking and gesture recognition for interactive interfaces using HMM', *Joint International Conference ICANN-ICONIP 2003*, Istanbul, Turkey, 2003.

[17] H. Zhou, T. Huang, "Tracking articulated hand motion with Eigen dynamics analysis", In Proc. Of International Conference on Computer Vision, Vol 2, pp. 1102-1109, 2003.

[18] JM. Rehg, T. Kanade, "Visual tracking of high DOF articulated structures: An application to human hand tracking", In Proc. European Conference on Computer Vision, 1994.

[19] A. J. Heap and D. C. Hogg, "Towards 3-D hand tracking using a deformable model", In 2nd International Face and Gesture Recognition Conference, pages 140–145, Killington, USA,Oct. 1996.

[20] Y. Wu, L. J. Y., and T. S. Huang. "Capturing natural hand Articulation". In Proc. 8th Int. Conf. On Computer Vision, volume II, pages 426–432, Vancouver, Canada, July 2001.

[21] B. Stenger P. R. S. Mendonc¸a R. Cipolla, "Model-Based 3D Tracking of an Articulated Hand", In proc. British Machine Vision Conference, volume I, Pages 63-72, Manchester, UK, September 2001.

[22] B. Stenger, "Template based Hand Pose recognition using multiple cues", In Proc. 7th Asian Conference on Computer Vision: ACCV 2006.

[23] L. Bretzner, I. Laptev, and T. Lindeberg. Hand gesture recognition using multi-scale colour features, hierarchical models and particle filtering. In Proc. 5th IEEE International Conference on Automatic Face and Gesture Recognition, pages 405–410, 2002.

[24] S. Mckenna and K. Morrison. A comparison of skin history and trajectory-based representation schemes for the recognition of user- specific gestures. Pattern Recognition, 37:999– 1009, 2004.

[25] K. Imagawa, H. Matsuo, R. Taniguchi, D. Arita, S. Lu, and S. Igi. Recognition of local features for camera-based sign language recognition system. In Proc. International Conference on Pattern Recognition, volume 4, pages 849–853, 2000.

[26] P.Viola and M.Jones, "Robust Real-time Object Detection", *International Journal of Computer Vision*, 2004, 2(57), pp. 137-154.

[27] D. G. Lowe. Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision, 60(2):91-110, 2004.

[28] S. Lazebnik, C. Schmid, J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In Proc. of IEEE Conference on Computer Vision and Pattern Recognition, 2006.

[29] Y. Jiang, C. Ngo, and J. Yang. Towards optimal bag-of-features for object categorization and semantic video retrieval. In ACM Int'l Conf. on Image and Video Retrieval, 2007.

[30] H. Jhuang, T. Serre, L. Wolf, and T. Poggio. "A biologically inspired system for action recognition", IEEE International Conference on Computer Vision (ICCV), 2007.

[31] J. Niebles, H. Wang, and L. Fei-Fei. Unsupervised learning ofhuman action categories using spatial-temporal words. International Journal of Computer Vision. 79(3): 299-318. 2008.

[32] C. Schüldt, I. Laptev, and B. Caputo. "Recognizing human actions: A local SVM approach", International Conference on Pattern Recognition (ICPR), 2004.

[33] A. Gilbert, J. Illingworth, and R. Bowden. "Fast realistic multiaction recognition using mined dense spatio-temporal features", IEEE International Conference on Computer Vision (ICCV), 2009.

[34] M. Marszalek, I. Laptev, and C. Schmid. "Actions in context", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009.

[35] L. Yeffet and L. Wolf. “Local trinary patterns for human action recognition”, IEEE International Conference on Computer Vision (ICCV), 2009.

[36] David J.C. MacKay. Information Theory, Inference, and Learning Algorithms, 2003.

[37] B. Zarit, B. Super, and F. Quek. Comparison of five color models in skin pixel classification. In ICCV’99 Int’lWorkshop on recognition, analysis and tracking of faces and gestures in Real-Time systems, 1999.

[38] N.Krishnan, R. Subban, R.Selvakumar et al., “Skin Detection Using Color Pixel Classification with Application to Face Detection: A Comparative Study”, IEEEInt. Conf. on Computational Intelligence and Multimedia Applications, Volume 3, 2007, Page(s):436-441.

[39] Q. Chen, N. Georganas, E. Petriu, “Hand Gesture Recognition Using Haar-Like Features and a Stochastic Context-Free Grammar,” IEEE Transactions on Instrumentation and Measurement -2008.

[40] N. Henze, A. Löcken, S. Boll, T. Hesselmann, M. Pielot. “Free-Hand Gestures for Music Playback: Deriving Gestures with a User-Centred Process”. MUM ’10: 9th International Conference on Mobile and Ubiquitous Multimedia, 2010.

[41] J. Wachs, M. Kölsch, H. Stern, Y. Edan. “Vision-Based Hand-Gesture Applications”, Communications of the ACM, Volume 54 Issue 2, February 2011.

[42] N. Dardas, Q. Chen, N. Georganas, E. Petriu, “Hand Gesture Recognition Using Bag-of-Features and Multi-Class Support Vector Machine”, HAVE 2010, 9th IEEE Int. Workshop on Haptic Audio Visual Environments and Games, Oct. 16-17, 2010, Phoenix, AZ, USA.

[43] N. Dardas, N. Georganas, “Real Time Hand Gesture Detection and Recognition Using Bag-of-Features and Multi-Class Support Vector

[44] Machine”, IEEE Transactions on Instrumentation and Measurement, 2011.