Multimed Tools Appl (2014) 70:2211–2233
DOI 10.1007/s11042-012-1236-4

Target-shooting exergame with a hand gesture control

Nasser H. Dardas · Juan M. Silva · Abdulmotaleb El Saddik

Published online: 18 September 2012

Abstract Exertion games (exergames) pose interesting challenges in terms of user interaction techniques. Players are commonly unable to use traditional input devices such as mouse and keyboard, given the body movement requirements of this type of videogames. In this work we propose a hand gesture interface to direct actions in a target-shooting exertion game that is played while exercising on an ergo-bike. A vision-based hand gesture interface for interacting with objects in a 3D videogame is designed and implemented. The system is capable to issue game commands to any computer game that normally responds to mouse and keyboard without modifying the underlying source code of the game. The vision system combines Bag-of-features and Support Vector Machine (SVM) to achieve user-independent and real-time hand gesture recognition. In particular, a Finite State Machine (FSM) is used to build the grammar that generates gesture commands for the game. We carried out a user study to gather feedback from participants, and our preliminary results show the high level of interest from users use this multimedia system that implements a natural way of interaction. Albeit some concerns in terms of comfort, users had a positive experience using our exertion game and they expressed their positive intention to use a system like this in their daily lives.

Keywords Posture recognition · Gesture recognition · Scale Invariant Feature Transform (SIFT) · K-means · Bag-of-features · Support Vector Machine (SVM) · Human-computer interaction

Electronic supplementary material The online version of this article (doi:10.1007/s11042-012-1236-4) contains supplementary material, which is available to authorized users.

N. H. Dardas (✉)
Electrical and Computer Engineering, University of Ottawa, Ottawa, ON, Canada
e-mail: ndard076@uottawa.ca

J. M. Silva
Computer Science with the Multimedia Communications Research Laboratory (MCRLab),
School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, ON, Canada

A. El Saddik
School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, ON, Canada

1 Introduction

Human Gestures are natural, expressive, and significant body movements which include physical motions of the fingers, hands, arms, face, or head with the aim to send events or interact with the virtual environment. Gestures can be static, where the human takes on a specific pose, or dynamic, defined by motion.

Natural interaction assumes that users must not be concerned with mice and keyboards but use instead gesture, speech, and human actions to interact with the system. Using "Hand Gesture" as an interaction style is gaining interest in human-computer interaction context in recent years. The main objective of gesture recognition research is to build a system which can recognize human gestures and utilize them to control an application or a video game instead of keyboard or mouse. The utilization of hand gesture interaction with video games enables players to interact with computer environments in a natural, immersive, and intuitive manner. Another benefit of utilizing hand gestures in this context is that visual data makes it possible to interact with computerized device at a distance, without requirement for a physical connection with the device to be controlled.

Hand gestures are a significant modality for man machine interaction. Comparing to several interfaces, hand gestures have the advantages of being simple to use, natural, immersive, and intuitive. Vision-based hand gesture recognition has been an active research area recently with applications such as sign language recognition [10], socially assistive robotics [1], directional indication through pointing [22], control through facial gestures [1], human computer interaction (HCI), immersive game technology, virtual controllers, affective computing and remote control. Within the broad range of application scenarios, hand gestures can be categorized into at least four classes [32]: controlling gestures, conversational gestures, communicative gestures, and manipulative gestures.

Computer games are a mainly technologically promising and commercially worthwhile field for innovative interfaces because of the entertaining nature of the communication. People are willing to try new interface technologies because they have the chance to be immersed in a challenging game-like environment.

Hand gesture interaction with video games poses several challenges. The response should be real time. The player should feel no noticeable delay between when she/he makes a hand gesture and when the computer reacts. The computer vision algorithms should be flexible, user-independent, and work against cluttered backgrounds. The robustness and reliability of a natural interface is also very critical. If the interface breaks frequently or does not work consistently, the "magic" of involvement and immersion in the interactive experience disappears. An additional issue is "gesture spotting and immersion syndrome," intending to recognize actual gestures from accidental motion. A method to solve this issue is by choosing a specific gesture to mark the "beginning" of a sequence of gestures.

"User adaptability and feedback" is the most necessity tackled in gaming applications. In gaming systems, players benefit from having to learn the gesture vocabularies used by the games. A training session is needed to train them how the hand gestures should be carried out, involving speed, trajectory, and finger configuration. Immersion is a strong experience of gaming, and has been regarded as a significant issue of interaction by gamers, designers, and researchers [2]. In order to provide immersive player experience, real-time bare hand-tracking and gesture recognition without particular setup procedures got a great interest recently. Direct manipulation based on tracking and recognition of hand gestures provides a more immersive interaction with virtual objects.

Our aim is to design a useful and natural man–machine interface that recognizes hand gestures without the help of any markers and gloves. In previous work [5, 7] we have designed and developed a hand gesture recognition algorithm that utilizes both Bag-of-features [14, 18], and SVM to realize natural, user-independent, and real time interaction between human and computers. In this paper we have extended such research to realize a hand gesture interface capable of issuing commands to an exertion computer game without the need to modify the game's source code. In this case we have employed a finite state machine (FSM) to build a grammar that generates the gesture commands. The FSM works by integrating the spatiotemporal (space-time) relation between every two consecutive frames in a video sequence in terms of the transition between recognized postures and their locations. The system starts with capturing images from a webcam to detect, track, recognize different hand gestures, and generate gesture commands for controlling the exertion game. Such gesture commands are translated into keyboard input commands that the underlying operating system can recognize and pass on to the running video game. This integration approach between interface and game allows for the extension of existing computer game titles using a hand gesture recognition interface.

The paper is organized as follows: section 2 introduces related works; the third section describes our hand gesture recognition system in details, including the training stage to build the cluster and the SVM classifier models and the testing stage for recognition; section 4 discusses how our proposed system builds a grammar; section 5 explains how our proposed system generates gesture commands; section 6 describes our 3D exertion game and the interaction with hand gestures; section 7 presents the user study to test the proposed system for user-independence; the last section gives the conclusion of our method.

2 Related works

Vision-based hand gesture interface gained a lot of interest in recent years since they can be used to control other applications or video games. In [27], a natural interface to navigate inside a 3D Internet city was presented using hand gestures. The user stands in front of the screen and uses hand gestures to navigate through the Internet 3D city. All gestures begin from a rest position given by the two hands on the table in front of the body. Gesture recognition is achieved by Hidden Markov Model (HMM) modelling of the navigating gestures. The feature vector contains velocity and position of hands and head, and blobs' shape and rotation.

In [4], a real-time hand tracking and hand posture recognition technique was utilized for the Jing-Hang Grand Canal Serious Heritage Game. This method permitted the players to interact with their customized avatar by natural hand gestures, observe specific models and navigate in the virtual constructions. The hand was detected using MCT-based (modified census transform) method [15]. Then, a multi-cue hand tracking algorithm [24] was utilized to track the hand. In the third step, the hand was segmented using a Bayesian skin-color model [31] and the hand tracking result. Finally hand posture was recognized by the feature based on density distribution.

In [13], games and gesture-based recognition were augmented into mobile phone interfaces. Finger tracking was experimented in an augmented reality (AR) board game on mobile phones and showed that it sustains an increased level of engagement and entertainment. Using markers attached to the fingers, canonical interactions were evaluated such as translating, scaling, and rotating virtual objects in a mobile AR setting.

In [28], a human gesture recognition technique was presented that uses 4-D spatiotemporal features. The technique used a time-of-flight camera for input so that depth information can be obtained. Besides, a man–machine interface was developed that senses human gestures and postures for TV viewing that allows intuitive operation through a device-free interface. In [26], a localized, continuous and probabilistic video representation was explored for human action recognition. The proposed representation makes use of the probabilistic distribution to encode the visual-motion information of an ensemble of local spatial temporal features in a continuous and localized manner. In [23], an integrated framework for analyzing human actions in video streams was proposed. The proposed approach introduces the implicit user in-the-loop concept for dynamically mining semantics and annotating video streams.

In terms of videogame implementation that leverage on vision-based gesture recognition in general, we can find work such as that of Kostomaj and Boh [16], who developed an Ambient Interactive Storybook framework for children, which includes videogames that promote physical activity by doing motion detection of the body to trigger actions in the game. Their approach is to use a webcam to track gross motion changes of the center of mass of the body to know if the player is moving left, right, down or up, and detects the transition between any of these positions and recognize them as postures. The location of the player is handed over to the game engine which is responsible to respond and manage all game variables.

In a similar work, Varona et al [29] focus on detecting user motion to derive body gestures. In their approach, they use non-parametric techniques to recognize the body gestures, which then they apply in the control of a video game in real time. In this case, the gesture recognition is not limited to center of body mass, but makes a more fine detection in different joints of parts of the body including the feet, thorax, shoulders and elbows. Their application of body gestures is used to control a Tetris video game.

In terms of hand gesture recognition applied to games, Li et al [19] make use of a rear projector and a traditional web cam placed on a table for implementing a game using ordinary hand gesture primitives for manipulating the game scenario. Their technique leverages on the fact that the rear projector under the table provides a backlit image of the hand making gestures over the table. They make the segmentation by specifying an appropriate light intensity threshold. They support the tracking of several hands interacting over the surface of the table.

To the best of our knowledge there are not other hand gesture recognition systems where a finite state machine (FSM) is used to build a grammar that can allow users to issue hand gesture commands for an exertion game, using an off-the-shelf web camera. Besides, there is not an implementation where the resulting hand gesture interface can be integrated with pre-existing computer game titles without the need to modify the game's source code.

3 Hand gesture recognition system

The proposed hand gesture recognition system consists of two stages: the offline training stage and the online testing stage. The cluster and multi-class SVM classifier models will be built in the training stage and will be used in the testing stage for recognizing hand gestures captured from a webcam. Three critical factors affect the accuracy of the system: the quality of the webcam in the training and testing stages, the number of the images used for training, and choosing the number of clusters to build the cluster model.

3.1 Training stage

The training stage model is shown in Fig. 1. In Fig. 1, the first step is extracting the keypoints for every training image using the scale invariance feature transform (SIFT). Then, a vector quantization technique maps keypoints from every training image into a unified dimensional histogram vector (bag-of-words) after K-means clustering. Finally, this histogram was treated as an input vector for a multiclass SVM to build the training classifier.

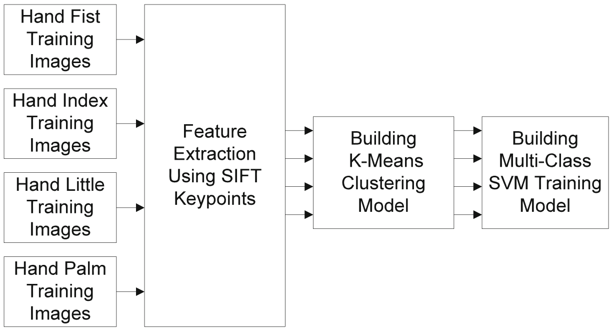

Hand Fist Training Images
Hand Index Training Images
Hand Little Training Images
Hand Palm Training Images
Feature Extraction Using SIFT Keypoints
Building K-Means Clustering Model
Building Multi-Class SVM Training Model

Fig. 1 Training stage

Before building the bag-of-features model, we captured 100 training images for each of the four hand gestures, for 10 people and under different illumination conditions to increase the robustness of the multi-class SVM classifier and the cluster model. All the training images illustrate the hand postures without any other objects and the background has no texture or objects (white wall). In this way, we guarantee that all the keypoints extracted from the training images using the Scale Invariant Feature Transform (SIFT) algorithm [21] will represent the hand posture only. SIFT is real time performance for low resolution portable gray map (PGM) images. Therefore, processing time for extracting the keypoints using SIFT can be reduced when the image resolution is reduced and converted into PGM format. The size of training images has been reduced to 50×50 pixels and converted into PGM format like the size of the small image (50×50 pixels) that contains the detected hand posture only for every frame captured from the video file in the testing stage.

The bag of features model is built using feature 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 sections.

3.1.1 Scale invariant feature transform (SIFT) algorithm

We used the Scale Invariant Feature Transform (SIFT) algorithm to extract the keypoints (features vectors) for each training image. The size of every feature vector depends on the number of histograms and the number of bins in each histogram. In Lowe's original implementation [21] a 4-by-4 patch of histograms with 8 bins each is used, generating a 128-dimensional feature vector. Figure 2 shows some training images with their keypoints. We can increase the number of training images to train the system as we wish for all the hand postures for different people with different scales, orientations and illumination conditions. The more training images used with different illumination conditions will provide more accurate k-means clustering and SVM models since extracted features for training images using SIFT are invariant to scale, orientation and partially to illumination changes [21]. Therefore, the time will increase for building the cluster model in the training stage [5–7]. However, this will not affect the testing stage speed.

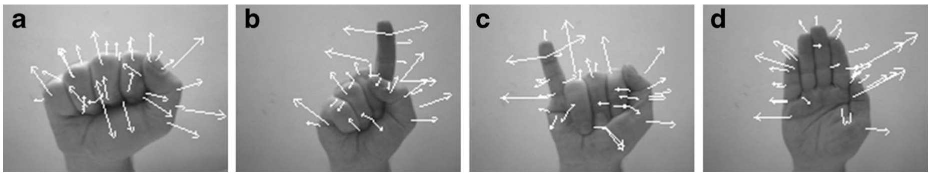


Fig. 2 Training images (**a**) fist with 23 features. (**b**) index with 31 features. (**c**) little finger with 27 features. (**d**) palm with 48 features

3.1.2 K-means clustering

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 keypoint to the nearest one based on Euclidean distance. After the assignment, the centroids (codevectors) are shifted to the average location of all the keypoints assigned to them, and assignments are redone. This procedure repeats until the assignments stop changing. Figure 3 shows this process in action for five keypoints: A, B, C, D, and E and two clusters.

Keypoints of each training image will be fed to 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 keypoint, extracted from a training image, will be represented by one component in the generated bag-of-words vector with a 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 will be 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 the index training images, class 3 for the little finger training images and class 4 for the palm training images.

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 keypoints extracted from its related image. If it is too large, then there will be an overfitting because of insufficient samples of the extracted keypoints from the training image. We choose the value 750 as the number of clusters k (visual vocabularies or codebook) to build our cluster model. This number provides the most accurate recognition rate by trying different values [7].

3.1.3 Building the training classifier using multi-class support vector machine (SVM)

A variety of approaches for decomposition of the multiclass problem into several binary problems using SVMs as binary classifiers have been proposed. In our implementation, multi-class SVM training and testing are performed using the library for SVM (LIBSVM) described in [Chang01]. This library supports multi-class classification and uses a one-against-one (OAO) approach for multi-class classification in SVM [12].

After mapping all the keypoints that represent every training image with its generated bag-of-words vector using the k-means clustering, we fed all bag-of-words vectors with their related classes into a multi-class SVM classifier to build a multi-class SVM training classifier model.

3.2 Testing stage

Figure 4 shows the testing stage by using face detection and subtraction and hand gesture detection before recognition. In order to detect the hand posture in the image, a four step system was designed

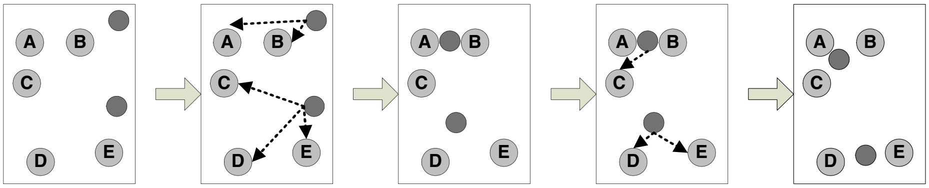


Fig. 3 K-means clustering with two clusters

according to our approach [5, 7] and as shown in Fig. 4. First, the templates of hand postures shown in Fig. 6 were loaded and their contours were extracted before capturing images from a webcam or video file. Second, the face was detected using Viola&Jones algorithm [Viola04] and then subtracted with a black circle because the skin detection will detect the face and the face's contours very close to the fist hand gesture contours. After face subtraction, skin color locus for the image was extracted for the user's skin color using the hue, saturation, value (HSV) color model since it has real-time performance, and it is robust against rotations, scaling, and lighting conditions. Then as the fourth step, hand posture was detected by eliminating false positive skin pixels and identifying hand posture and other real skin color regions using contours matching with the loaded hand postures templates contours. The detected hand posture was saved in a small image (50×50 pixels). The keypoints were extracted from the small image that contains the detected hand posture only and were fed into the cluster model to map them into a "bag of words" vector and finally this vector was fed into multi-class SVM training classifier model to recognize the hand posture.

3.2.1 Face subtraction

Viola and Jones method [30] is used for face deduction. We subtract the face before applying the skin detection algorithm [5, 7] to detect the hand gesture only by replacing the face area with a black circle for every frame captured as shown in Fig. 5.

3.2.2 Hand posture detection

Detecting and tracking human hand in a cluttered background will enhance the performance of hand gesture recognition using the bag-of- features model in terms of accuracy and speed because the keypoints extracted will represent the hand gesture only. Besides, we will not be confined to the frame resolution size captured from the webcam, because we will always extract the keypoints 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 also be robust against the cluttered background because we process the detected hand gesture area only.

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 [5, 7] will be used to search for the human hands and discard other skin colored objects for every frame captured. Before capturing the frames, we loaded the templates of the four hand postures as shown in Fig. 6: 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 to get rid of other skin like objects.

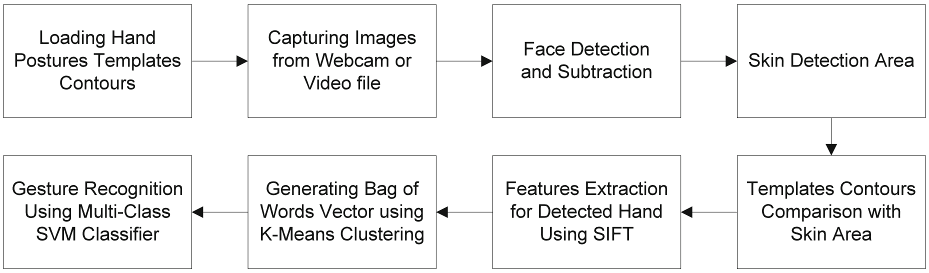

Loading Hand Postures Templates Contours
Capturing Images from Webcam or Video file
Face Detection and Subtraction
Skin Detection Area
Templates Contours Comparison with Skin Area
Features Extraction for Detected Hand Using SIFT
Generating Bag of Words Vector using K-Means Clustering
Gesture Recognition Using Multi-Class SVM Classifier

Fig. 4 Testing stage

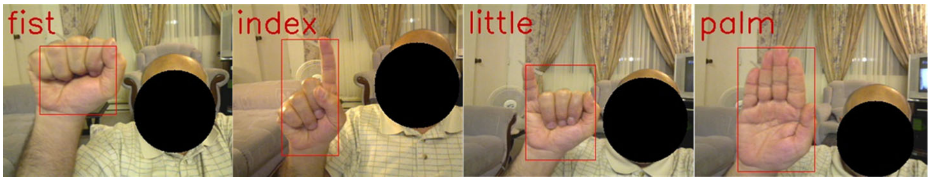


Fig. 5 Face detection and subtraction as well as hand posture detection (in red square)

In our implementation [5, 7], 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 [33]. 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. 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 [17]. 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 threshold ranges are simultaneously satisfied: $0° < H < 20°$ and $75 < S < 190$.

Once the skin area had been detected, we found contours of the detected area and then compare them with the contours of the hand postures templates. If the contours of the skin area comply with any of the contours of the hand postures templates, then that area will be the region of interest by enclosing the detected hand posture with a rectangle, which will be used in tracking the hand movements and saving hand posture in a PGM format small image (50×50 pixels) for every frame. The small image will be used in extracting the keypoints to recognize hand gesture.

4 Building grammar

Behaviour classification has developed to be an active research subject in the computer vision field. There are several methods for classifying high-level behaviour of vision based information. In [11], the method of decomposing behavioural classification was divided into two stages: first, a low-level event classifier based on features of raw video information, and second, a high-level behavioural classifier based on sequences of events that were generated from the first stage. Therefore, a behaviour classifier can be considered as a classifier of sequences of events.

Human motion analysis techniques can be classified into stochastic algorithms such as hidden Markov models (HMMs), or deterministic algorithms such as finite state machine (FSM). HMMs were used as state-space models for behaviour classification. However, using

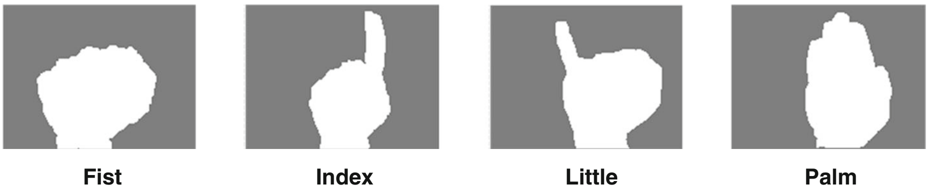


Fig. 6 Templates of hand posture

syntactic (grammar-based) structural methods are better than HMMs when some sequences of low-level information inherently fall into meaningful behaviours.

A series of events or features, over time and space, create behaviour. In the proposed system, the events are based on monitoring the transitions among detected hand postures and their locations for every two consecutive frames captured from a webcam. Low-level classifications are based on features of raw video information resulted from the real-time posture recognition classifier as discussed in section 3. In the high-level, the recognized hand postures for every two consecutive frames are concatenated in a temporal sequence to form behaviour, which is a hand gesture. 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 [3]. A method is required to monitor the sequences of detected hand postures and classify them such as a syntactical grammar-based method, which was presented in [11]. The different hand gesture behaviour compositions have to be defined before classifying sequences. Every hand gesture behaviour will have a different temporal composition of the detected hand postures. These sequence compositions are defined with syntax rules such as regular grammars. A set of syntax rules, describing hand gesture behaviour, is known as a grammar. The grammar is implemented through a finite state machine (FSM). The FSM monitors the sequence of the hand postures. If a sequence of hand postures goes with a specific hand gesture behaviour, its related FSM will accept this sequence. Thus, the sequence of hand postures is recognized as hand gesture behaviour if its related FSM agrees with the sequence.

We used spatiotemporal (space-time) correlation between every two consecutive frames in a video sequence in terms of the transition between recognized postures and their locations to develop our system into real time dynamic hand gesture recognition. The timing relation for transition among recognized postures is monitored for every two consecutive frames of a video sequence by saving every two recognized consecutive postures in two states queues, where the recognized previous posture is saved in the old state queue, while the recognized current posture is saved in the new state queue. The movement direction or space relation between every two recognized postures from every two consecutive frames is tracked by monitoring the difference between the two locations of the recognized previous and current hand postures.

The gesture classifier recognizes the detected hand posture and the result of the recognition are fed into a FSM. The grammar of the FSM decides the real tasks. These tasks can be commands or events to be executed from the keyboard. The FSM of the grammar for the palm posture is shown in Fig. 7. The FSM of the grammar for the other postures such as fist, index, and little will be the same as the palm posture.

The non-terminal symbols of the grammar are the states={Palm, Little, Index, Fist}.
The terminal symbols of the grammar are the outputs={Turn right/Left, Move Up/Down, Zoom In/Out, Shoot}.
The input symbols of the FSM are the movement directions of palm posture={Up/Down, Right/Left, Forth/Back, Hold}

Table 1 shows FSM of the grammar for the palm posture. We notice from Table 1 that the output depends on the input and the current state. Besides, the next state depends on the current state and the input.

To implement the grammar of the FSM, we mapped these commands sent to the keyboard by correlating the transition between recognized hand postures states and hand motions directions for every two consecutive frames for the palm posture case as shown in Fig. 7. This spatiotemporal (space-time) correlation develops our system into real time dynamic hand gesture recognition.

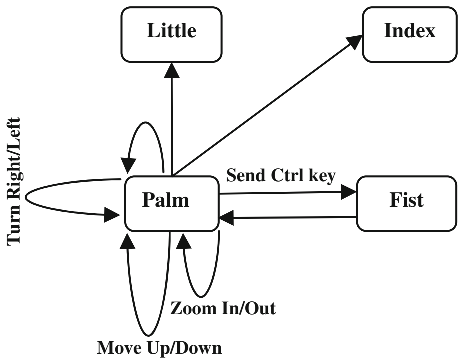


Fig. 7 FSM of the grammar for the palm posture

5 Generating gesture commands

Based on the grammar built in the previous section, our hand gesture recognition system can generate gesture commands, which can be used to control or interact with an application or a video game 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. We define 4 postures: fist, index, little, and palm as in Fig. 6. The 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, etc. For each posture we have four transitions to other states, hence, we define 16 events or gesture commands. 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. The case of no movement for each posture had not taken into consideration as it were counted already in the first way when the transition occurred from fist to fist or palm to palm and so on as in Fig. 8. Finally, by observing the hand posture size or scale: when it comes close (zoom in) or far away (zoom out) from the camera. As we have two cases for each posture, we have eight gesture commands for the four postures. In the following sections, we will explain how our system can generate gesture commands.

5.1 Transitions among postures

Transitions between postures depend on saving every two recognized postures from every two consecutive frames of a video sequence in two states queues: the new state queue, which

Table 1 FSM of the grammar for the palm posture

Current state	Next state	Direction (input)	Send key	Event (output)
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	^{+}	Zoom In
Palm	Palm	Back	^{-}	Zoom Out
Palm	Palm	Hold	Nothing	Nothing
Palm	Fist	Hold	Send Ctrl key ^	Shoot
Fist	Palm	Hold	Nothing	Nothing

holds the recognized current posture and the old state queue, which holds the recognized previous posture. The recognized current posture will be saved in the new state queue after transferring its posture state to the old state queue. Thus, for every frame captured, the posture state of the old state queue is emptied. Then, the posture state of the current state queue is transferred to the old state queue. Finally, the recognized current posture from the frame will be saved in the current state queue. The system will keep observing the two states queues for every two consecutive frames captured and 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 8 shows all the transition states of fist posture with all other postures.

5.2 Movement direction for each posture

Movement direction for each posture depends on tracking the movement direction of the detected posture using rectangles, which captures the detected hand posture. Once a posture is detected, the coordinates of the middle of the rectangle X and Y are recorded. The system will always monitor the absolute difference of distance between the two points of the middle of the rectangle in the X and Y coordinates for every two successive frames that have the same posture. If the absolute difference of distance in the X direction is larger than absolute difference of distance in Y direction and the absolute difference is larger than 1 cm, then the hand posture is moved left or right. If the difference of distance in the 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 the X direction and the absolute difference is larger than 1 cm, 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 is moved up. Figure 9 shows all the motion direction cases of palm posture.

5.3 Distance from the camera for each posture

Distance from the camera for each posture depends on tracking the size of the height of 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

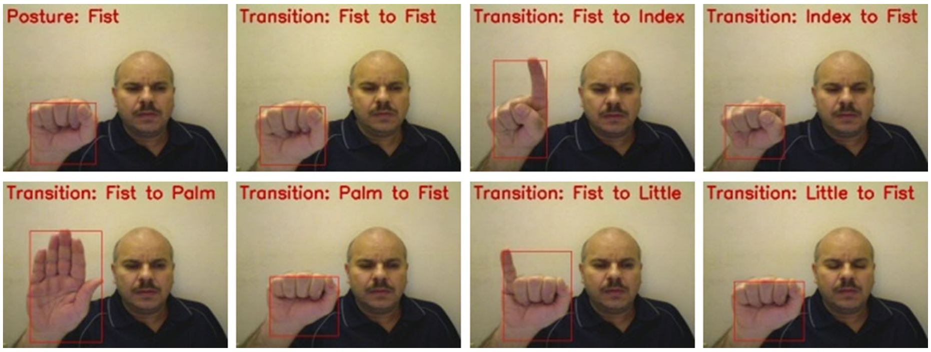

Posture: Fist
Transition: Fist to Fist
Transition: Fist to Index
Transition: Index to Fist
Transition: Fist to Palm
Transition: Palm to Fist
Transition: Fist to Little
Transition: Little to Fist

Fig. 8 Transitions from fist to other postures

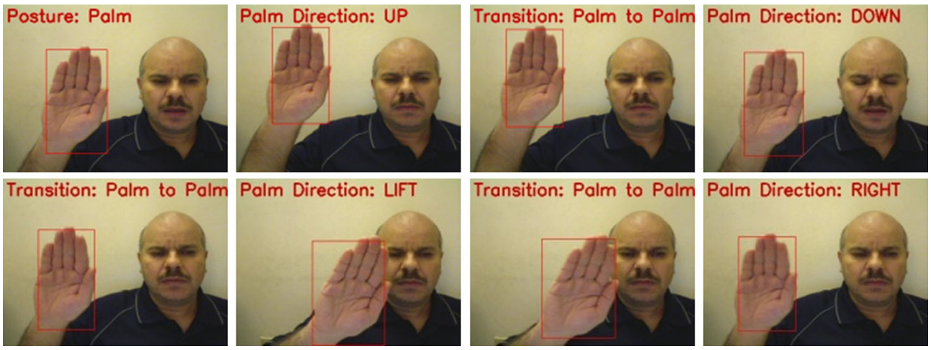


Fig. 9 Movement direction cases of palm posture

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 and larger than 1 cm, then the posture gets closer to the camera (zoom in) and if the difference is negative and larger than 1 cm, then the posture gets away from the camera (zoom out). Figure 10 shows all the zoom cases of little posture.

The size of the height of the rectangle, which captures the detected hand posture only, is inversely proportional to distance from the camera. The threshold for detecting hand movement away or close to the camera depends on the rectangle height difference or distance difference between detected hand postures from camera for every two successive frames that have the same posture, which is 1 cm, regardless of the speed of motion. If we make threshold less than 1 cm, the system will consider left/right/up/down change of direction or zoom In/Out for any very small hand vibration in any dimension (x, y, z).

6 Interaction with a 3D exertion game using hand gestures

With the purpose of testing our hand gesture recognition algorithm in the context of an interactive application, we have integrated the recognition engine into a target shooting exertion game. Exertion games are computer video games that require the player to be physically active while playing the game in order to achieve her/his goal. Given the physicality of this type of games, players are faced with the difficulty of interacting with the game using traditional input controllers like gamepads. Hence, exertion games present a good opportunity for applying body movement tracking techniques as an alternative to more traditional input controllers. It is in this context that we have decided to use our hand gesture recognition algorithm.

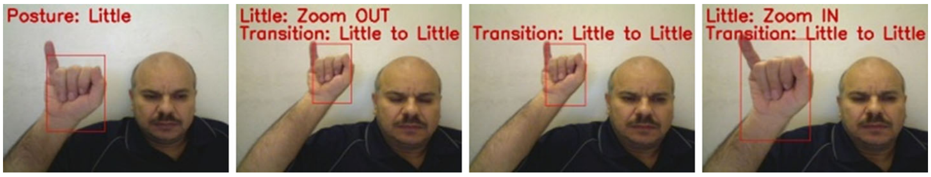

Posture: Little
Little: Zoom OUT
Transition: Little to Little
Transition: Little to Little
Little: Zoom IN
Transition: Little to Little

Fig. 10 Zoom cases of little posture

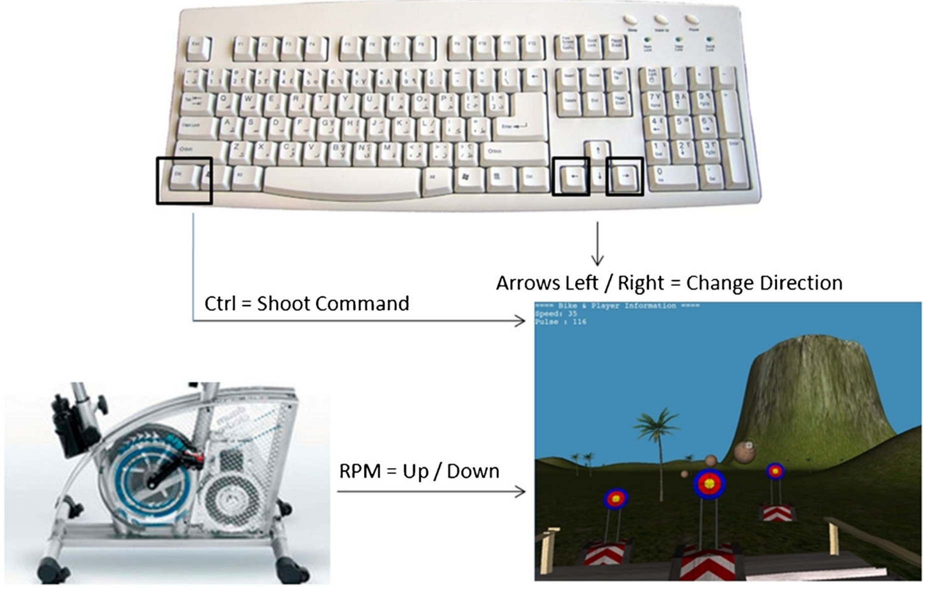
6.1 3D Exertion game

The exertion game that we have used is based on a typical target-shooting scenario and is fully described in previous work [25]. The User Interface shows a 3D island environment where the camera viewport has a first person shooter perspective showing the direction where the player is looking at. The player's character is standing on a platform facing a field, and in her/his line of view there are three targets located at different distances which the player must knock down by throwing coconuts at them as shown in Fig. 11. The player plays the game while exercising on a stationary bicycle which acts as the input controller for the game.

The game engine handles the rules of the game as follows: to gain points, the player must knock down targets by hitting them with a coconut. Bonus points are gained when all 3 targets are knocked down within a 10 s time frame. After 10 s of being knocked down, a target stands up again by itself. Targets are located at various distances from the player. The player gets feedback about the general direction of the throw by seeing a crosshairs icon on the screen as shown in the screenshot of Fig. 11.

The vertical movement of the crosshairs is controlled by a stationary bicycle used as an input controller. The RPM (revolutions per minute) reading from the stationary bike's pedals is mapped into the Y coordinates that drive the vertical displacement of the crosshairs, which affects the trajectory and distance (depth) to which a coconut may be thrown by the player. The higher the RPM reading is, the higher the throw and vice versa. As the game is played, the player must adjust slightly the speed of his/her pedaling to reach closer or farther targets.

The player controls the horizontal movement of the crosshairs by using the left and right arrows of the keyboard. The control is limited to changing the direction on which the crosshairs displaces (left or right), but the speed and movement of the crosshairs is preset. This adds a level of difficulty and exertion to the game, and requires the player to make frequent changes of direction. Throwing the coconuts is triggered by pressing the 'control' key on the keyboard. The player may shoot as often as she/he wants.



```
==== Bike & Player Information ====
Speed: 35
Pulse : 116
```

Fig. 11 Game input methods (without hand gesture control)

Figure 11 depicts the 3 input methods to control the game. Shooting command and changing direction left and right are controlled by the keyboard while changing direction up and down is controlled by the bike. This is an excellent example of a game that requires a novel way of interaction, since handling the keyboard while exercising on the bicycle is an awkward challenge. Therefore, in this work, as is described in the next section, the functionality of the keyboard is replaced with a hand gesture control, to provide a more natural and entertaining mode of interaction for the player.

6.2 The hand gesture interface

To address the problem of moving the crosshairs left and right and throwing coconuts while pedaling the bike, we used three hand gestures that generate commands, two of which allow the player to change direction (left/right) and a third one to shoot a coconut. In our implementation we wanted to minimize the requirements for modifying the codebase of the videogame to interface with the hand-gesture recognition system. For this purpose we decided to have both applications (Game and gesture recognition process) run independently but at the same time on the same host computer. Since the game is already coded to respond to keyboard events for left/right and shoot actions coming from the keyboard, we have made the gesture recognition process make a hardware interruption to the underlying operating system to set those keyboard events manually. In this fashion, for each in-game action, we mapped a specific hand-gesture to the related keyboard input event as we will describe in the next section. This technique allows the integration of any third party existing video games with the hand gesture recognition system. With the hand gesture recognition replacing the functionality of the keyboard, we aimed at having the player more engaged in the gaming experience using a more natural interface. To implement this functionality, we mapped these events sent to keyboard by integrating the transition between recognized hand postures and hand motions directions for every two consecutive frames for the palm posture case according to the FSM presented in Fig. 7. The derived interaction commands for the game are presented in Table 2.

Figure 12 depicts the input methods to control the game with the hand gesture interface while the player on the bike making hand gestures and playing the game.

For a host computer, we used a PC running Windows XP operating system. The configuration of the system was a 2.0 GHz processor and 2 GB RAM. The exercise bike was an Ergo-bike 8008 TRS 3 model connected to the host computer by a serial cable. The webcam was a low-cost Logitech QuickCam 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. The webcam is connected to the host computer with a USB port. The game was developed using the Unity Engine and programmed using javascript and C#. No significant modifications were made to the game to be compatible with the gesture interface as it simply responded to the keyboard input events received from the operating system.

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 in C/C++.

Table 2 Interaction commands performed by hand gestures

Current state	Next state	Direction	Send key	Event
Palm	Palm	Right	{RIGHT}	Move Right
Palm	Palm	Left	{LEFT}	Move Left
Palm	Fist	Hold	^	Shoot

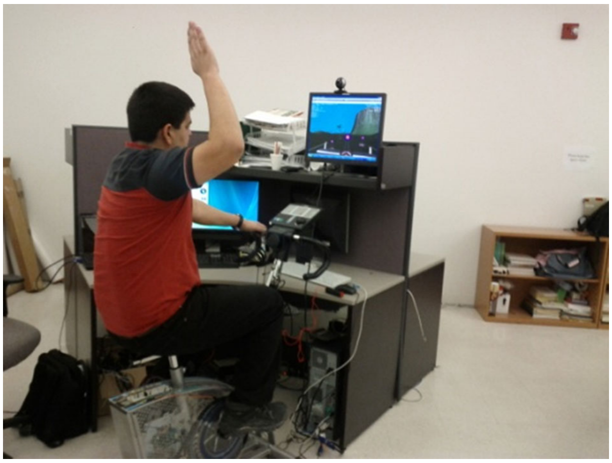


Fig. 12 Game input methods with a hand gesture recognition

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.

6.2.1 Shooting action using hand gesture

The first custom gesture represents a "fire button" action. When this gesture is detected by the gesture recognition process, it generates a Key-pressed event with the "Ctrl" key as an identifier. This event is then picked up by the game, which responds by throwing a coconut. This gesture can be observed in Fig. 13 when the player changes her/his hand posture from palm posture to fist posture to shoot the target by throwing a coconut. The palm posture of the player is shown in the small screenshot in the top left corner of Fig. 13a while in the next frame, the player changes her/his hand posture to fist posture to shoot the target as shown in the small screenshot on the top left corner of Fig. 13b.

6.2.2 Changing direction action using hand gesture

The other two gestures represent the action of moving the crosshairs left or right. Again, when either of these two gestures is detected, a key-pressed event is generated with the "Left arrow" or "Right arrow" identifiers, accordingly. These gestures can be observed in Fig. 14 when the player moves her/his hand palm posture left or right.

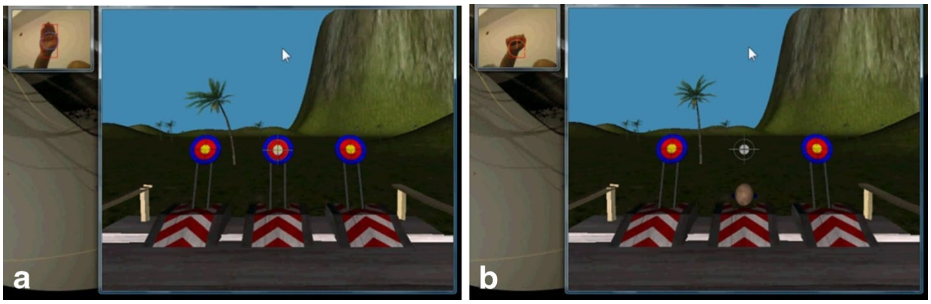


Fig. 13 "Shooting" action using hand gesture for the exertion game. On the top left: (**a**) Player with palm posture (**b**) Player changes to fist posture for shooting

7 User study

With the purpose of getting feedback from actual users about our interface implementation, we conducted a user study, where 15 participants played the game using the hand gesture interface. At this stage of our research we wanted to assess the level of technology acceptance from users to this particular interface implementation. For this, the scope of our present user study is to gather the opinions and impressions from users in a structured way by using the Technology Acceptance Model [8]. At the end of the game sessions participants answered a questionnaire and were interviewed to collect their impressions and their suggestions regarding how natural was for them to play using hand gestures to control the actions in the game.

There were 12 male and 3 female participants. All of them were students in the computer science department at the University of Ottawa at the undergraduate level, and none of them worked in the same laboratory than the authors. Their ages ranged from 21–25. None of

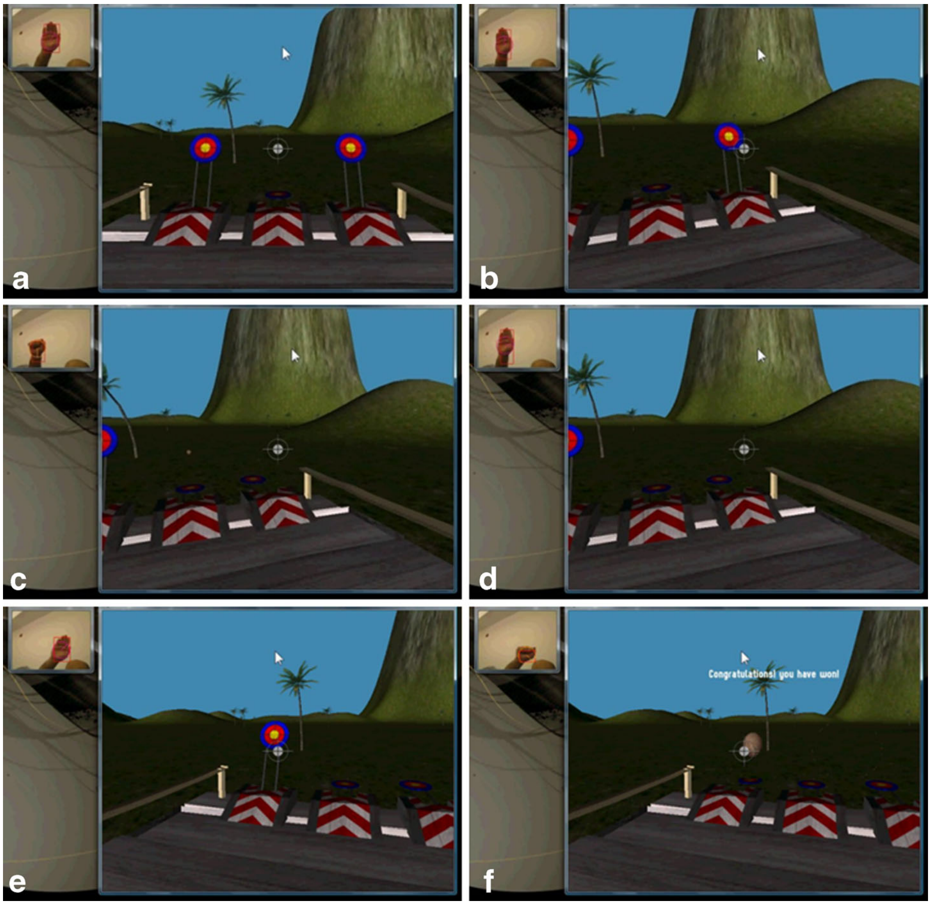


Fig. 14 "Left/Right" movement actions with shooting using hand gesture in the exertion game. On the top left: (**a**) Player with palm posture (**b**) Player moves her/his palm posture right to move crosshairs right (**c**) Player changes to fist posture for shooting (**d**) Player changes to palm posture (**e**) Player moves her/his palm posture left to move crosshairs left (**f**) Player changes to fist posture for shooting

them reported having any physical or mental disabilities and although all of them reported having played video games at some point, none of them were particularly avid gamers.

The sessions were carried out with one participant at a time. At the beginning of the session, the participant was introduced to the overall procedure of the experiment, and a demonstration of the game and the general instructions for playing were presented. Then the participant was given 5 min to do some warm up in the bicycle alone without having the game on. Once the warm-up period ended the game was started and the player was left to play the game for 5 min. At the end of the 5 min, the game would be over and the participant was allowed to cool down pedaling on the bike.

After the gaming session participants were asked to answer a questionnaire. The questionnaire consisted of a 5-point Likert scale with 10 items including reversal statements [20]. Likert scales are typically used to measure psychometric response to assess the attitude that users have towards some topic or experience, and are applied often as the items in a questionnaire. It has been suggested that Likert scales are among the most common methods to estimate usability [9].

In our study, Likert items were concerned with the player perception of ease of use, accuracy and overall performance of the exergame's hand gesture interface. Particularly we wanted to assess how natural the interactions were to them and if they perceived any performance issues such as difficulty to perform changes of direction or perceived delays in those changes. Figure 15 shows a sample question from the full questionnaire (Appendix 1).

The interviews were guided by open questions that would cover closely the same topics of the questionnaire except that they were meant to encourage the participant to elaborate verbally about their overall experience.

8 Results and discussion

In this section, we present the results of the questionnaire and the interviews carried out during the evaluation.

Responses to the individual items of the questionnaire are considered as *ordinal data*, which is a common practice with Likert scales. Hence, we present the results in terms of descriptive statistics that when analyzed can help to draw a picture of the perception that the users got for each topic covered by the items of the questionnaire. Tables 3 and 4 show such descriptive statistics of the data collected.

The items of the questionnaire were divided into four main areas. Questions 1 to 3 were all related to the 'Left/Right' hand gestures and each one addressed a specific topic: ease of use, perception of delay, and accuracy to perform the task. Questions 4 to 6 similarly addressed the same topics but for the "Shoot" hand gesture. Questions 7 and 8 were in general about the whole hand gesture interface to the game, in order to assess if it was comfortable and felt natural and intuitive. Finally questions 9 and 10 were aimed at assessing the willingness of the participants to use a system like such in their daily lives. The structure of the questionnaire was defined following the precepts of the Technology Acceptance Model (TAM) [8].

Fig. 15 Sample Likert item from the questionnaire

8) Playing the game using hand gestures felt natural and I was able to focus on playing the game

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
[ ]	[ ]	[ ]	[ ]	[ ]

Table 3 Median Mode and Range for each Likert item

Question:	Median	Mode	Range
1.- Left/Right Gesture – Easy	4	4	3
2.- Left/Right Gesture – No Delay	3	3	2
3.- Left/Right Gesture – Accurate	4	4	3
4.- Shoot Gesture – Easy	5	5	2
5.- Shoot Gesture – No Delay	3	3	3
6.- Shoot Gesture – Accurate	4	4	2
7.- Hand Gestures - Comfortable	3	2	2
8.- Hand Gestures -Natural	5	5	3
9.- Use Intention	4	5	2
10.- Use Intention (reversed)	5	5	1

Table 3 shows a summary of the Median, Mode and Range on the data set pertaining to each of the items of the questionnaire.

From this data summary it can be seen that the lowest value of the median across all questions was 3 ('neutral') while the highest was 5 ('strongly agree') with a similar tendency for the most repeated responses (mode) with values between 2 and 5. The range of the data sets was never beyond 3 points, which shows a cohesive response pattern.

Table 4 shows an aggregation of responses for each of the Likert levels in each question. For each question we see the tally of responses and a second row shows the same values expressed in percentage.

Table 4 Aggregation of responses of each Likert level across all 10 questions

Question:	Strongly disagree [10]	Disagree [1]	Neutral [22]	Agree [32]	Strongly agree [2]
1.- Left/right gesture - easy	0	1	4	5	5
		7 %	27 %	33 %	33 %
2.- Left/right gesture - no delay	0	5	8	2	0
		33 %	53 %	14 %	
3.- Left/right gesture - accurate	0	4	0	10	1
		27 %		67 %	6 %
4.- Shoot gesture - easy	0	0	3	4	8
			20 %	27 %	53 %
5.- Shoot gesture - no delay	1	4	6	4	0
	6 %	27 %	40 %	27 %	
6.- Shoot gesture - accurate	0	0	3	9	3
			20 %	60 %	20 %
7.- Hand gestures - comfortable	0	7	6	2	0
		47 %	40 %	13 %	
8.- Hand gestures -natural	0	1	2	4	8
		7 %	13 %	27 %	53 %
9.- Use intention	0	0	1	7	7
			6 %	47 %	47 %
10.- Use intention (reversed)	0	0	0	7	8
				47 %	53 %

From the details of the distribution of responses in Tables 3 and 4, it can be seen that both of the modalities of hand gestures ('Left/Right' and 'Shoot') had a relatively low rating (3 'neutral') in response to the statement of 'not perceived delay'. According to the responses users did not generally agree, and presented a 'neutral' perspective to the statement that said that there was no-delay. By observation we could measure an occasional noticeable delay between the execution of the hand gesture by the player and the recognition of it by the system. When it occurred, this delay could vary in a range from 0.5–1.0 s. However, players soon adjusted their game strategy to cope with the delay in those cases and were able to perform well in the game. There were no significant false positive detections, and when the system would interpret the natural movement of the hand (resulting from exercising on the bike) as a 'move left/right' command, players would simply correct the action to aim for the desired target. This adjustment from the subjects is consistent with their responses in the questionnaire, where users did not seem to be affected in terms of their perceived accuracy to knock down the targets (the goal of the game). In this case, they rated with values mostly on or above the 'agree' level with 73 % of the responses to the 'left/right' gesture and with 80 % of the responses for the 'shoot' gesture.

In terms of ease of use, both types of hand gestures were rated generally in the same order, between 'agree' and 'strongly agree'. This trend was reinforced during the interviews with the users, who expressed that the method of interaction was simple and intuitive.

When judging if the use of the hand gestures was comfortable for directing the actions on the exertion game, users expressed some reserves. The median value was at 3 with a mode of 2, and 47 % of the responses expressing a 'disagree' position and 40 % in 'neutral'. During the interviews users expressed that, even for an exertion game, they think that keeping the hand raised at all times for a longer game session would be quite uncomfortable, which is consistent with their responses on the questionnaire. Still, in terms of how natural the interaction was to them, the median and mode values were at 5 with a low range of 1, which shows that the mode of interaction was intuitive.

Overall from the interviews we could see that users were highly motivated to use the hand gesture control and this can also be seen on the answers to the last two items of the questionnaire, which show an intention of use with a rating of 4 and 5 in the Likert scale.

The proposed system demonstrated acceptable interaction with 15 users in terms of perceived accuracy and speed because the features of hand gestures were extracted in real-time for using the SIFT algorithm, and were invariant to scale and orientation. Future work could include an extended evaluation where different techniques are compared with the subjects within the context of this particular game implementation.

9 Conclusion

We proposed a bare hand gesture recognition interface that generates commands to control objects directly in an exertion videogame that makes use of a stationary bicycle as one of the main inputs for game playing. Our proposed hand gesture recognition system utilizes both Bag-of-features and Support Vector Machine (SVM) to realize natural, user-independent, user-friendly, and real-time interaction between human and computers. With this interface, the user can control and direct left-right movement and shooting action by a set of hand gesture commands.

The preliminary results of a user study to evaluate our implementation show the high level of interest from users to make use of multimedia systems that implement natural ways of interaction as the one presented here. Although some concerns in terms of comfort, users had a positive experience using our exertion game and they expressed their positive intention to use a system like this in their daily lives.

Appendix 1. Questionnaire

For each one of the following statements, please mark the option below that closest reflects your level of agreement with the statement.

1) Moving the crosshairs left and right with hand gestures was easy to accomplish.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
[ ]	[ ]	[ ]	[ ]	[ ]

2) I did not perceive any delay when moving the crosshairs left and right using hand gestures

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
[ ]	[ ]	[ ]	[ ]	[ ]

3) I had difficulty hitting the targets because the application would not respond all the time to my left/right hand gestures.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
[ ]	[ ]	[ ]	[ ]	[ ]

4) Shooting in the game by using hand gestures was easy

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
[ ]	[ ]	[ ]	[ ]	[ ]

5) I did not perceived delays when shooting using the hand gesture

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
[ ]	[ ]	[ ]	[ ]	[ ]

6) I had difficulty hitting the targets because the application would not respond all the time to my 'shoot' hand gesture

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
[ ]	[ ]	[ ]	[ ]	[ ]

7) I felt comfortable interacting with the game by means of hand gestures

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
[ ]	[ ]	[ ]	[ ]	[ ]

8) Playing the game using hand gestures felt natural and I was able to focus on playing the game

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
[ ]	[ ]	[ ]	[ ]	[ ]

9) If I wanted to use an exercise bike on a regular basis I would rather use this exergame with the hand gestures, instead of a traditional bike

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
[ ]	[ ]	[ ]	[ ]	[ ]

10) The whole experience with the exergame and hand gesture interface was awkward and I would generally avoid using it for my exercising

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
[ ]	[ ]	[ ]	[ ]	[ ]

References

1. Baklouti M, Monacelli E, Guitteny V, Couvet S (2008) Intelligent assistive exoskeleton with vision based interface. In: Proceedings of the 6th international conference on Smart Homes and Health Telematics, vol. 5120, pp 123–135
2. Brown E, Cairns P (2004) A grounded investigation of game immersion. In: CHI'04 extended abstracts on human factors in computing systems. ACM, New York, pp 1297–1300
3. Chen Q, Georganas N, Petriu E (2008) Hand gesture recognition using Haar-like features and a stochastic context-free grammar. IEEE Transactions on Instrumentation and Measurement 57(8):1562–1571
4. Chen S, Pan Z, Zhang M, Shen H (2011) A case study of user immersion-based systematic design for serious heritage games. Multimedia Tools and Applications
5. Dardas N, Alhaj M (2011) Hand gesture interaction with a 3D virtual environment. The International Journal of ACM JORDAN 2(3):186–194
6. Dardas N, Chen Q, Georganas N, Petriu E (2010) 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, Phoenix, AZ, USA
7. Dardas N, Georganas N (2011) Real time hand gesture detection and recognition using bag-of-features and support vector machine techniques. IEEE Transactions on Instrumentation and Measurement 60 (11):3592–3607
8. Davis FD (1986) A technology acceptance model for empirically testing new end-user information systems: theory and results. Massachusetts Institute of Technology. Available at: http://dspace.mit.edu/handle/1721.1/15192
9. Dumas J (1998) Usability testing methods: subjective measures: part II - measuring attitudes and opinions. October issue of Common Ground, The newsletter of the Usability Professionals' Association, pp 4–8
10. Fang G, Gao W, Zhao D (2007) Large-vocabulary continuous sign language recognition based on transition-movement models. IEEE Trans Syst Man Cybern, Part A: Systems and Humans 37(1):1–9
11. Goshorn R, Goshorn D, Kolsch M (2008) The enhancement of low-level classifications for ambient assisted living. Proceedings of the 2nd Workshop on Behaviour Monitoring and Interpretation, BMI'08
12. Hsu C-W, Lin C-J (2002) A comparison of methods for multi-class support vector machines. IEEE 373 Trans Neural Network 13:415–425
13. Hurst W, Wezel C (2012) Gesture-based interaction via finger tracking for mobile augmented reality. Multimedia Tools and Applications
14. Jiang Y, Ngo C, Yang J (2007) Towards optimal bag-of-features for object categorization and semantic video retrieval. In ACM Int'l Conf. on Image and Video Retrieval
15. Just A, Rodriguez Y, Marcel S (2006) Hand posture classification and recognition using the modified census transform. In: Proceedings of the Seventh IEEE international conference on automatic face and gesture recognition (FG'06), University of Southampton, UK, pp 351–356
16. Kostomaj M, Boh B (2011) Design and evaluation of user's physical experience in an ambient interactive storybook and full body interaction games. Multimed Tool Appl 54(2):499–525
17. Krishnan N, Subban R, Selvakumar R et al (2007) Skin detection using color pixel classification with application to face detection: a comparative study. IEEE Int Conf on Computational Intelligence and Multimedia Applications 3:436–441
18. Lazebnik S, Schmid C, Ponce J (2006) Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition
19. Li K, Du Y, Fu Z (2011) TreeHeaven: a table game using vision-based gesture recognition. In: Proceedings of the 2011 ACM symposium on the role of design in UbiComp research & practice (RDURP '11). ACM, New York, NY, USA, pp 11–14
20. Likert R (1932) A technique for the measurement of attitudes. Arch Psychol 140(55)
21. Lowe DG (2004) Distinctive image features from scale-invariant keypoints. Int J Comput Vision 60 (2):91–110
22. Nickel K, Stiefelhagen R (2007) Visual recognition of pointing gestures for human-robot interaction. Image and Vision Computing, Elsevier 25(12):1875–1884
23. Ntalianis K, Doulamis A, Tsapatsoulis N, Doulamis N (2010) Human action annotation, modeling and analysis based on implicit user interaction. Multimed Tool Appl 50(1):199–225
24. Pan Z, Li Y, Zhang M, Sun C, Guo K, Tang X, Zhou S (2010) A real-time multi-cue hand tracking algorithm based on computer vision. In: Virtual reality conference (VR). IEEE, Piscataway, pp 219–222

25. Silva J, El Saddik A (2011) An adaptive game-based exercising framework. Virtual environments, human-computer interfaces and measurement systems (VECIMS)
26. Song Y, Tang S, Zheng Y, Chua T, Zhang Y (2012) Exploring probabilistic localized video representation for human action recognition. Multimed Tools and Appl 58(3):663–685
27. Sparacino F (2008) Natural interaction in intelligent spaces: designing for architecture and entertainment. Multimedia Tools and Applications
28. Takahashi M, Fujii M, Naemura M, Satoh S (2011) Human gesture recognition system for TV viewing using time-of-flight camera. Multimedia Tools and Applications
29. Varona J, Jaume-i-Capo A, Gonzalez J, Perales F (2009) Toward natural interaction through visual recognition of body gestures in real-time. Interact Comput 21:3–10
30. Viola P, Jones M (2004) Robust real-time object detection. International Journal of Computer Vision 57 (2):137–154
31. Weng C, Li Y, Zhang M, Guo K, Tang X, Pan Z (2010) Robust hand posture recognition integrating multi-cue hand tracking. In: Proceedings of the entertainment for education, and 5th international conference on E-learning and games. Springer, New York, pp 497–508
32. Wu Y, Huang TS (1999) Human hand modeling, analysis and animation in the context of HC1. In: IEEE International Conference Image Processing. Kobe, Japan
33. Zarit B, Super B, Quek F (1999) 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

Nasser Hasan Dardas received the B.Sc. degree in Electrical Engineering from Jordan University of Science and Technology, Irbid, Jordan, in 1988, the M.Sc. degree in Electrical Engineering – Communication from University of Jordan, Amman, Jordan, in 1999 and the M.A.Sc. degree in Electrical Engineering – Computer Networks from Carleton University, Ottawa, ON, Canada, in 2008. He is currently working toward the Ph.D. degree in Electrical and Computer Engineering at University of Ottawa, Ottawa, ON, Canada. His research interests include visual object detection and tracking, and image processing. His current research topic is focused on vision-based hand gesture detection, tracking, and recognition in real time.

Juan M. Silva received the B. Eng. degree in Computer Science and Engineering from La Paz Institute of Technology, Mexico, and the M.C.S. degree in Computer Science from the CICESE Research Center, Ensenada, Mexico, in 2006. He is currently working toward the Ph.D. degree in Computer Science with the Multimedia Communications Research Laboratory (MCRLab), School of Electrical Engineering and Computer Science, University of Ottawa. After completing his masters he worked for various Information Technology companies in Mexico and has collaborated in some start-up companies in Canada. His research interests include adaptive exergaming, ambient intelligence, pervasive computing, web technologies and multimedia communications.

Abdulmotaleb El Saddik has been named a 2010 Distinguished Scientist by the Association for Computing Machinery (ACM), the first ever at SITE and at the Faculty of Engineering. Professor El Saddik's current research, which provides an excellent bridge between computer science and engineering, is in the area of ambient intelligence and multimedia communications. In particular, he focuses on the analysis, design and development of haptics audio visual algorithms and of collaborative protocols and applications. He is also a Fellow of IEEE, FEIC and FCAE. The Distinguished Scientist grade recognizes ACM members who have at least 15 years of professional experience and 5 years of continuous professional membership, and whose accomplishments in and impact on the computing field have been significant.