Model and implementation of body movement recognition using Support Vector Machines and Finite State Machines with Cartesian coordinates input for gesture-based interaction

Raphael W. de Bettio, André H. C. Silva, Tales Heimfarth and André P. Freire
Computer Science Department, Federal University of Lavras, Brazil
e-mails: {raphaelwb, andrecosta, tales, apfreire}@dcc.ufla.br

Alex G. C. de Sá
Computer Science Department, Federal University of Minas Gerais, Brazil
e-mail: alexgcsa@dcc.ufmg.br

ABSTRACT

The growth in the use of gesture-based interaction in video games has highlighted the potential for the use of such interaction method for a wide range of applications. This paper presents the implementation of an enhanced model for gesture recognition as input method for software applications. The model uses Support Vector Machines (SVM) and Finite State Machines (FSM) and the implementation was based on a Kinect® device. The model uses data input based on Cartesian coordinates. The use of Cartesian coordinates enables more flexibility to generalise the use of the model to different applications, when compared to related work encountered in the literature based on accelerometer devices for data input. The results showed that the use of SVM and FSM with Cartesian coordinates as input for gesture-based interaction is very promising. The success rate in gesture recognition was 98%, from a training corpus of 9 sets obtained by recording real users' gestures. A proof-of-concept implementation of the gesture recognition interaction was performed using the application Google Earth®. A preliminary acceptance evaluation with users indicated that the interaction with the system via the implementation reported was satisfactory.

Keywords: Gesture, SVM, FSM, Kinect, Model.

1. INTRODUCTION

Novel methods for more natural interaction have made the use of interactive technologies more ubiquitous and present in a wide range of computer applications used in every-day life. Advancements in the use of interaction based on geastsure recognition created several opportunities to enhance the interaction in a number of types of applications [1]. In particular, the use of gesture recognition in video games such as the Wii® and the XBox® have popularised the use of gesture-based interaction. However, the use of such interaction method is not limited to entertainment applications, but has a great potential to be used in a wide range of applications, as mentioned by Mitra and Acharya [1]: navigation and/or manipulation in virtual environments, enabling young-aged children to interact with computers, sign language recognition, enhancements to hearing impaired people, forensic identification, and others.

The development of more natural interfaces, including gesture-based interaction, was one of the main aspects mentioned by Weiser [2, 3] in his vision for the development of ubiquitous computing. In fact, many other research works have highlighted the importance and potential for the development of natural gesture-based interfaces in a number of different contexts [4, 5, 6, 7, 8].

The availability of generic easy-to-implement interface components is an essential aspect to allow developers to use gesture-based interaction in their applications. Making generic models for gesture-based interaction available to developers can simplify the implementation of more interactive interfaces, without the need to implement complex gesture recognition algorithms.

Several different approaches and algorithms have been used to perform automatic gesture recognition, such as the application of Hidden Markov Models (HMM) [9, 10, 11, 12], Finite State Machines (FSM) [13, 14, 15, 16], Fuzzy Logic [17] and Support Vector Machines (SVM) [16, 18].

Previous research studies have proposed the implementation of gesture and movement recognition using SVM and FSM. In the work performed by Matsunaga and Oshita [18], for example, this approach was applied for the recognition of movements with accelerometers used in the Wii® video game. The results of this implementation showed that the use of SVM and FSM for gesture recognition was very promising. However, the approach had technological limitations due to the use of input via accelerometers.

Research works reporting on the use of SVM and FSM for gesture recognition with users' postures represented using Cartesian coordinates have not been encountered in the literature. The use of Cartersian coordinates for data input for recognition provides more flexibility for the recognition of gestures. Unlike accelerometers, the use of Cartesian coordinates enables the recognition of a wider range of movements, once they can be used to represent any posture, and a movement can be defined as a set of postures in a given time-span.

This paper presents the definition and implementation of a computational model for gesture-based interaction in computer applications, by means of data input with Cartesian coordinates. The enhanced model was based on the model proposed by Matsunaga and Oshita [18], using SVM and FSM, with a better potential for generalisation of the types of movement that can be recognised by the use of Cartesian coordinates instead of accelerometers.

The implementation used a Kinect® device [19] for the acquisition of the postures. The training of the recognition algorithm was performed based on 9 sets of data obtained from real users' gestures, with a success rate of 98% in the recognition. A proof-of-concept implementation was performed to evaluate the adequacy of the gesture recognition implementation to be used as input method for applications. The implementation of a gesture-based interface for the applica-

tion Google Earth® was successfully accomplished, and the results from a preliminary acceptance evaluation indicated that the system worked satisfactorily.

This paper is organised as follows. Section 2 presents a review of the literature and related work. Section 3 presents the main concepts of the techniques and technologies used in this work. Section 4 details the description of the computational model used for the gesture recognition used in this research. Section 5 presents the implementation of a proof-of-concept of the model and the results from the preliminary acceptance evaluations. Finally, Section 6 presents the conclusions and proposed future work.

2. RELATED WORK

In this section we present the main studies that applied approaches related to thosed used in the present article. We also discuss the main gaps encountered in these works that were addressed in our approach.

Oshita and Matsunaga [16] proposed a method for automatic learning for gesture recognition. In their work, they combined two different techniques for pattern recognition: 1) Self-Organising Maps (SOM), used to divide training data with users' gestures into different phases to build a finite state machine (FSM) for each gesture and 2) Support Vector Machines (SVM), applied to learn the transition conditions for each state (gesture phase) of the state machine, considering that there was one SVM for each state. Each of those methods was more adequate to different parts of the process. SOMs provide good features to categorise data into groups. SVMs have good performance to partition characteristics spaces into different regions belonging to each class. Tests performed by Oshita and Matsunaga [16] involved both simple and complex gestures. Both types of gestures had satisfactory recognition success rates. The process to build state machines to define transitions in gesture recognition to generate movements in the work presented in this paper was based on Oshita and Matsunaga's [16] proposal.

In another related work, the same author [18] presented a method where the state machine is set manually. The main difference between the approach proposed by Matsunaga and Oshita [18] and the present work was the input data for the model. In the present work, the recognition of gestures was based on transitions from positions (Position X to Position Y, for example) and not on parts of movements detected by accelerometers (e.g. right leg up, right leg down) as used in Matsunaga and Oshita [18]. In the work performed by Matsunaga and Oshita [18], the gesture recognition was performed using a Wii® remote control, that is based on accelerometers. In the present work, a Kinect® device was used to recognise gestures. By using a Kinect® device, it was possible to perform the gesture recognition by processing digital images, in which the human body joints are recognised by a process that results in a set of Cartesian coordinates.

In another work, developed by Biswas and Basu [20], a Kinect® device was applied as a means to perform body recognition. However, their work was limited to posture recognition and did not recognise movements. Biswas and Basu [20] also used Computer Vision techniques that were based on images generated by the Kinect® device, whilst in the present work, pre-processed data obtained from the device were used to recognise the position of body joints. Another similar work was developed by Ren *et al.* [21], that used input data generated by the camera in a Kinect device for hand gesture recognition.

Song *et al.* [22] performed a study with the recognition of continuous movements of hands and body, and presented a computational model using SVMs to recognise body movements. However, similarly to the work performed by Biswas and Basu [20], the recognition was performed directly on the images generated by the camera, and not based on pre-processed data with more refined information about the position of body joints.

3. THEORETICAL BACKGROUND

Support Vector Machines

In the present work, Support Vector Machines (SVM) was used to perform the recognition of body movements. The main goal of this technique is to build a hyperplan separator to abstract a decision surface for each input. The decision process consists of distinguishing data input from two opposite classes. The decision model defines a maximum margin, which is the maximum distance between a separating hyperplan and the nearest point to the two classes in relation to this hyperplan [23]. Once the margin is calculated, two support vectors paralell to the hyperplan are added to the model. The distance of each vector to the hyperplan is defined as the value of the maximum margin. At the end of the process, a given data input is classified according to its proximity to each vector.

Another issue with the use of SVM a limitation when it is not possible to separate data linearly. When this property occurs in some of the input data, one of the ways to overcome this issue is to assign a flexible margin. This way, the modelling of a SVMs will allow for empirical errors while trying to minimise them [24], maximising the margin in relation to the separating hyperplan.

If the use of a flexible margin for classification is not adequate due to the complexity of the data, another possible approach is to change the search space in an algorithm, in order to make it separable. This can be done by using a Kernel, which is a function that transforms the dimension of the input data [25]. This is normally performed when the separating surface cannot be done by a straight line (simple hyperplan). Then, a plan with more dimensions is used to enable the data separation using a hyperplan.

However, such transformation can lead to the *Curse of dimensionality*, as the number of dimensions can become too large, whilst the quantity of example remains the same. This can make the solution to the problem considerably more complex.

There are a number of Kernel functions that can be used, with the most common being polinomial, gaussian and sigmoidal functions [26]. An analogy can be made between the choice of a kernel function and the choice of an activation funtion in an artificial neural network.

As pointed out by Hsu *et al.* [27], the configuration of a Support Vector Machine is less complex than an artificial neural network. In order to obtain a better generalisation capacity with artificial neural networks, it is necessary to obtain a certain sensibility (or experience) to define empirical data (supervised learning), such as the learning rate and *momentum*, for example.

Besides, according to Haykin [28], SVM used for machine learning can provide good performance in terms of generalisation in problems related to pattern classification and linear regression, despite the fact that it does not incorporate domain knowledge about the problem. This is a unique feature of SVMs.

Kinect

This section provides an overview of the Kinect® device, used in this work to obtain information about users' body movements and postures, in order to incorporate its structure into a framework to manipulate computer applications by means of gestures and body movements. All information was based from the technical manual [19] provided with the device API.

Kinect® is a device developed by Microsoft to be used as an input medium for the videogame XBOX® 360, enabling users to interact in a natural manner with games by body movements and gestures, in other words, with a natural user interface (NUI).

The programming interface in the Kinect® Software Development Kit (SDK) provides a sophisticated software library and tools to help programmers develop applications using the Kinect® device. The components available on the Windows Kinect SDK are: Kinect hardware, Microsoft Kinect drivers, NUI API, KinectAudio DMO and Windows 7 default APIs. Considering the hardware aspects, the Kinect SDK contains:

- **RGB camera sensor** - with the ability to recognise the users' faces;
- **Four-component microphone** - with the ability to recognise users' voices;
- **Depth sensor** - with the ability to detect users' body joints in a 3D space.

In the work presented in this paper, the Depth sensor and the NUI API were particularly important for the implementation of the model. The NUI API consists of a set of APIs that retrieve data from the image sensors and enables the control of other devices of applications from the Kinect®.

With the aid of the NUI Skeleton API (part of the NUI API), it was possible to obtain detailed information about the position of an individual traced by the Kinect® device. The NUI Skeleton API is able to recognise up to two individuals. However, in the present work, the model considered that only one individual would be interacting at one time.

Fig. 1 illustrates the 20 body joints that can be recognised using the Kinect® device.

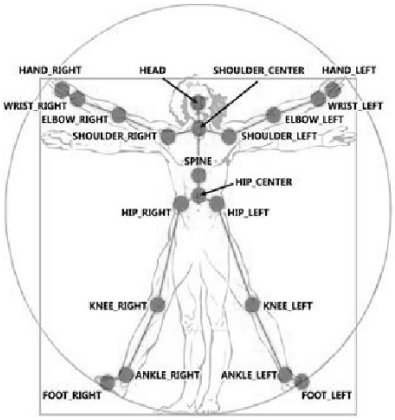

Fig. 1: **Skeleton positions available in the Kinect device relative to the human body [19].**

As mentioned previously, the NUI Skeleton API provides information about the position of an individual. This information is available as a vector containing the Cartesian positions of all body joints shown in Fig. 1. This vector was used as the input to the gesture recognition model described in the following section.

4. GESTURE RECOGNITION MODEL

This section presents the three steps used in the computational model for gesture recognition used in the present work. The model consists of the following steps: 1) data pre-processing, 2) posture classification and 3) movement classification.

Data pre-processing

The first version of the model implemented in this work considered initially only the right arm, using the following joints: right shoulder, right elbow and right hand. The right wrist was not considered due to its Cartersian coordinates in a 3D space being too close to the coordinates of the right hand.

During the pre-processing it is necessary that the Cartesian coordinates undergo a translation process, with the right shoulder as the reference point. The choice for the right shoulder was made in order to allow for more flexible movements. By having coordinates relative to the right shoulder, the movement would be independent of the users' lateral position in relation to the capture device.

The translation process is performed following the process illustrated at Fig. 2 and defined by the Eq. (1). Firstly, each black point illustrated in the figure was named as $P1$, $P2$ and $P3$, each being, respectively, the shoulder, elbow and hand. The main objective of this process is to translate the points $P1$, $P2$ and $P3$ from their original position, represented in Fig. 2-I to the final position, represented in Fig. 2-II. In order to perform this translation, point $P1$, as described previously, was translated as being the origin, and points $P2$ e $P3$ were translated having point $P1$ as reference.

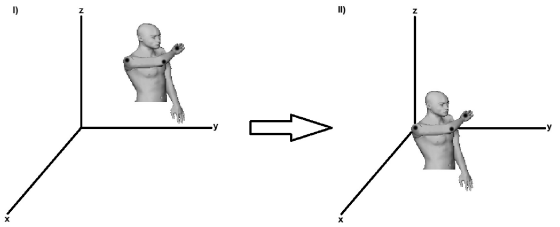

Fig. 2: **Graphical representation (in 3D space) of the applied translation process.**

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (1)$$

In the Eq. (1), d_x, d_y e d_z represent the translation vector; x, y e z are the Cartesian coordinates of the initial positions and x,y e z are the Cartesian coordinates of the final positions. The translation can also be represented in the following manner (Eq. (2)).

$$P' = T(d_x, d_y, d_z) \cdot P \quad (2)$$

In the representation of the Eq. (2), T is defined as a translation function; P is the initial position of point P and P is the final position of point P after the translation.

According to the mathematical definition of the problem, point $P1$ has the Cartesian coordinates x_1, y_1 e z_1, point $P2$ has coordinates x_2, y_2, and point $P3$ has x_3, y_3 e z_3, i.e., $P1 = (x_1, y_1, z_1)$, $P2 = (x_2, y_2, z_2)$, and $P3 = (x_3, y3, z_3)$. The translation of $P1$ to the origin position is performed according to the Eq. (3).

$$T(-x_1, -y_1, -z_1) = \begin{bmatrix} 1 & 0 & 0 & -x_1 \\ 0 & 1 & 0 & -y_1 \\ 0 & 0 & 1 & -z_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

Hence, after the translation of $P1$ to the origin, the following points are obtained. The equations Eq. (4), Eq. (5) and Eq. (6) show the resultant points.

$$P_1' = T(-x_1, -y_1, -z_1) \cdot P_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad (4)$$

$$P_2' = T(-x_1, -y_1, -z_1) \cdot P_2 = \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \\ 1 \end{bmatrix} \quad (5)$$

$$P_3' = T(-x_1, -y_1, -z_1) \cdot P_3 = \begin{bmatrix} x_3 - x_1 \\ y_3 - y_1 \\ z_3 - z_1 \\ 1 \end{bmatrix} \quad (6)$$

Posture classification

The second step in the model for gesture recognition is the classification of individual postures. Five different postures based on movements of the right arm were defined for this model. Those postures are illustrated in Fig. 3, and described as following.

- **a** - Right arm pointing upwards;
- **b** - Right arm pointing downwards;
- **c** - Right arm resting;
- **d** - Right arm pointing to the left;
- **e** - Right arm pointing to the right.

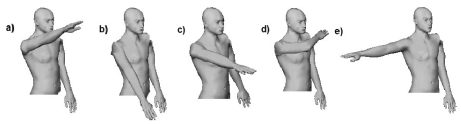

Fig. 3: **Five postures considered in the posture classification.**

As mentioned previously, the classification of those posturse considered three joints (shoulder, elbow and hand) with the Cartesian coordinates having been translated having the right shoulder as reference.

The next step in the classification was to use a Support Vector Machine that was able to recognise the defined postures. The parameterisation of the Support Vector Machines was performed according to the method defined by Hsu *et al.* [27]. The type of SVM used was a Support Vector Classification (C-SVC), specialised in classification problems. In order to do this, it was necessary to normalise collected sample data, making the coordinates x, y and z of each joint considered (elbow and hand) in the sample normalised in the range between -1 and 1.

The chosen kernel function was a Radial Basis Function (RBF). This function enables the solution of problems that are non-linearly separable and has the advantage of having only two empirical parameters - *cost* and *gamma*, represented by C and γ, respectively. The values used in this work for C and γ were 0.98 and 0.001, respectively. The Eq. (7) describing RBF is presented as following.

$$K(x, y) = e^{\gamma \|x-y\|^2} \quad (7)$$

The model for the SVM training was based on classes and attributes. A class consists of a number of attributes. After a system based on SVM is trained, this system is able to determine the probability that a set of attributes is part of a given class. This way, each of the five postures was defined as a class, with each of these classes being defined by 6 attributes. Three of those attributes are the coordinates x, y and z, corresponding to the elbow joint, and the other three are the x, y and z coordinates corresponding to the hand position. The third joint (the shoulder) was not considered, as it was translated as being the origin point (0, 0, 0). For this reason, after the pre-processing stage, this position will be always the same.

Movement classification

The third step in the model for gesture recognition is the movement classification. In this model, a movement consists of a sequence of postures.

Based on the model adopted by Oshita and Matsunaga [16], a Finite State Machine (FSM) was created to model movements. According to Louden [29], FSM are mathematical models to describe particular types of algorithms. In particular, FSM can be used to describe the process of pattern recognition in input strings and can also be used to build search systems.

For the research reported in this paper, finite state machines were built having only one initial state (posture c - arm resting), defining possible movements when it was possible to reach other states following transitions from the initial state. Each other state in the machine is represented by a posture a, b, d and e, being respectively states 1, 2, 3 and 4. There is only one final state. The state machine is illustrated in Fig. 4.

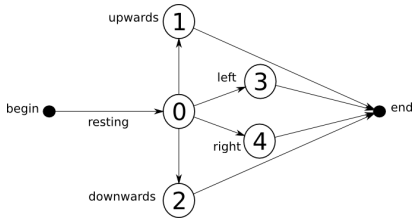

Fig. 4: **Representation of Finite State Machine for movement recognition.**

This way, it was possible to identify the following movements: arm resting to arm pointing upwards, arm resting to arm pointing downwards, arm resting to arm pointing to the right, and arm resting to arm pointing to the left. Fig 5 illustrates the first of those movements and how it is modelled

using a FSM. As soon as posture c (arm resting) is recognised, the machine is positioned at state 0. Following, states 1, 2, 3, or 4 can be reached if some movement from the resting state (0) is identified. If a non-recognised movement is identified, the machine returns to the initial state.

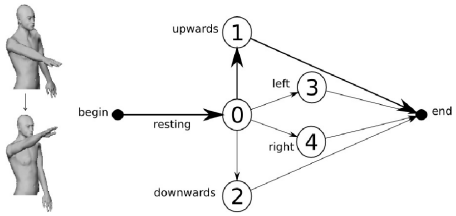

Fig. 5: **Representation of transition from state 0 (initial) to state 1 (arm pointing upwards).**

The next section describes how this model was implemented in an application as a proof of concept and preliminary acceptance tests performed with users.

5. PROOF-OF-CONCEPT IMPLEMENTATION AND PRELIMINARY ACCEPTANCE TESTS

This section describes the implementation of the gesture recognition model in an application as a proof of concept and preliminary acceptance tests performed with users.

As presented in previous sections, the Kinect® device, when integrated with the software library NUI Skeleton API is able to identify 20 body joints of up to two different individuals. However, this API does not provide resources to classify postures or movements, being restricted to making available vectors containing Cartesian coordinates that may be used as an input to build applications. The main goal of the proof-of-concept implementation described in this section was to show the suitability of the model for gesture recognition presented in the previous sections to be used as the input method of interactive applications. The chosen application for the first implementation was Google Earth®.

The prototype contains three main components: a posture classification (PC) module, a movement classification (MC) module and a keyboard simulator (KS). The architecture of this implementation is described in Fig. 6. The symbols used in the figure are described as following.

- **P1 e P2** - Positions 1 e 2;
- **M1** - Movement 1;
- **x1,y1,z1 - xn,yn,zn** - Vector of Cartesian coordinates that constains users' joint positions identified by the Kinect® device;
- **PC** - Posture classification module;
- **P1,P2 and PN** - Postures 1,2 and n;
- **MC** - Movement classification module;
- **KS** - Keyboard simulator;
- **A, CTRL+A** - Keys (or keystrokes) that can be related to a movement.

The first component, the Posture Classification (PC) module, is responsible for classifying postures enacted by users and recognised by a Kinect® device. Aiming to simplify the implementation, this component has been built based on api SVMLIB fully described in Chang and Lin [26]. By evaluating the examples of postures (P1 and P2), it is possible

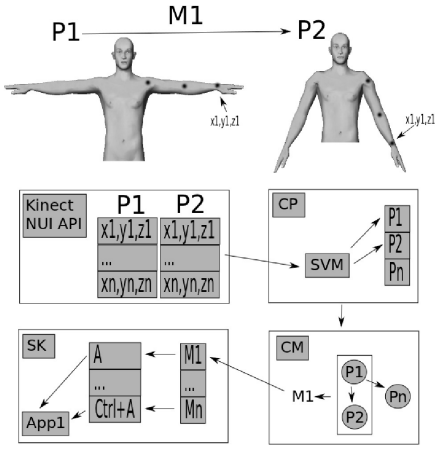

Fig. 6: **Architecture of the implemented prototype.**

to observe that the Cartesian coordinates identified as x_1, y_1 and z_1 represent different points in a 3D space, i.e., they can be used as landmarks to identify the two examples of postures. However, the main goal of the prototype is to be able to identify any posture. In order to do this, it is necessary that more joints be recognised. The SVM technique was used due to its ability to classify groups of coordinatse, and hence, classify any posture.

The second component, the movement classification (MC) module is based on the concept that a movement consists of a sequence of postures. Following the example in Fig. 6, movement M1 consists of the sequence composed by postures P1 and P2. In order to enable the mapping of any sequence of postures, state machines were designed, in which each state identifies one posture, and the transitions model changes in postures to identify movements.

The third component, the keyboard simulator (KS), is responsible for simulating the action of pressing a key or keystrokes when any movement is successfully identified by the movement classification module. By using this module, the prototype can use any movement as an input for applications that can be operated via keyboard.

Based on the postures and joints previously defined, data from users were collect to train the recognition algorithm from nine different participants. Each participant was positioned at a pre-defined distance from the Kinect® device (2 meters), and 100 samples of snapshots of postures were collected (20 of each posture) for each participant. This made for a total of 900 snapshots in the sample. Given the lower complexity of the movements considered in this version of the implementation, the sample of nine participants would be enought to capture the core features needed for the training of the SVM, as was later confirmed by the satisfactory sucess rate.

The training of the algorithm for this module was performed using two different methods, in order to compare the results. Both methods grouped snapshot samples into 9 sets, being 6 sets used for training and 3 sets used for tests. The main different between the two methods was the way the samples were grouped to generate the training and test sets - one method grouped samples based on the individuals and the other grouped samples randomly.

In the first method, in which samples were grouped by individuals, each of the 9 data sets contains samples for a single individual. Based on a combinatorial analysis, by having 9 sets being combined into 2 groups, being one group with 6 sets and the other with 3 sets, a total of 84 possible combina-

tions can be obtained. All those combinations were tested, with an average success rate of 82.920%, and standard deviation 11.381%.
With the second method, grouping samples randomly, 9 data sets were selected from the samples randomly in the proportion of 80/20 (80% for training and 20% for tests). The random selection was built in a way that all wamples were used only once, without repetition. Based on these sets, the same 84 possible combinations were tested, with a resulting average success rate of 98.531% and standard deviation 1.233%. Fig. 7 presents a graph comparing the results from the two different methods. It is possible to observe that the method with random grouping of samples had a considerably higher success rate than the method that grouped samples based on each individual.

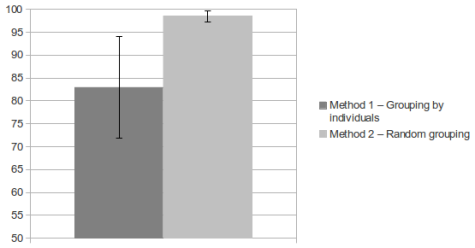

Fig. 7: **Success rates with different training methods.**

The effectiveness of pattern recognition tools (including the SVM) is directly proportional to the quality of the data used for training. The quality of the data is determined by how representative the training data is in relation to the data used in testing and in the final application. This way, based on the results found in this evaluation, the second method had a much better effectiveness. This is probably due to the fact that the training was based on samples from all individuals, differently from the first method, where training data were based on individuals being considered separately. The second approach was able to represent a wider range of Cartesian coordinates and body joints that represented a posture, yielding a better success rate in the recognition.
After training the recognition algorithm, the architecture described in Fig. 6 was integrated to the application Google Earth®, in order to allow it to be operated using gesture interaction. This application provides features to enable users to visualise satellite photos in a 3D space in the form of a globe. The reason why this application was chosen was because the interaction with the globe can be performed using simple keyboard operations (left-arrow key, right-arrow key, upward key and downward key). The keyboard commands were linked to specific movements detected by the movement classification module. Fig. 8 shows an illustration of the prototype and the keyboard simulation module.
After the implementation of the prototype with gesture recognition, a preliminary acceptance test was performed with 9 users. Users were asked to interact with Google Earth using movements with their right arm, after having been given instructions about the basic movements that can be used in the application. They were then asked to rate the difficulty to use the system in a 5-point Likert scale ranging from 1 - very difficult to 5 - very easy. The vast majority of answers were between 4 (easy) and 5 (very easy).

6. CONCLUSIONS

This paper presented the results of the implementation of a model for gesture recognition for interactive applications, based on input via Cartesian coordinates. The work presented improvements based on a model proposed by Matsunaga and Oshita [18] with Support Vector Machines and Finite State Machines. The improved model presented in this work can be generalised to a wider range of applications due to the use of Cartesian coordinates as input, as opposite to data entry based on accelerometers performed in previous works.
The results of the training algorithms with 9 sets obtained with real users were very satisfactory, with a success rate of 98% in the recognition of gestures.
The results obtained from this work show that the use of gesture recognition based on SVM and FSM is very promising. After the preliminary acceptance tests with users that showed that the use of basic operations was satisfactory, we intend to conduct future studies to further explore into more detail different issues related to the interaction based on gestures, and its relationship to the usability of applications.
As future work, we also intend to improve the implementation of the model by increasing the number of Cartesian coordinates considered in the gesture recognition, in order to be able to recognise other postures and other movements. We also intend to extend the model to recognise other types of movements, such as moving towards or away from the computer.

7. REFERENCES

[1] S. Mitra, and T. Acharya, Gesture Recognition: A Survey, *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, May, 2007, vol. 37, no. 3, pp. 311–324.

[2] M. Weiser, The Computer for the 21st Century, *Scientific American – Special Issue on Communications, Computers, and Networks*, September, 1991, vol. 265, no. 3, pp. 94–104.

[3] M. Weiser, Some Computer Science Issues in Ubiquitous Computing, *Communications of the ACM – Special Issue on Computer Augmented Environments: Back to the Real World*, July, 1993, vol. 36, no. 7, pp. 75–84.

[4] S. S. Rautaray, and A. Agrawal, Real Time Multiple Hand Gesture Recognition System for Human Computer Interaction, *International Journal of Intelligent Systems and Applications (IJISA)*, May, 2012, vol. 4, no. 5, pp. 56–64.

[5] K. A. Yuksel, and S. H. Adali, Prototyping Input Controller for Touch-less Interaction with Ubiquitous Environments, *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI'11)*, Stockholm, Sweeden, August-September, 2011, pp. 635–640.

[6] P. M. Yanik, J. Manganelli, J. Merino, A. L. Threatt, J. O. Brooks, K. E. Green, and I. D. Walker, Use of Kinect Depth Data and Growing Neural Gas for Gesture based Robot Control, *Proceedings of the 6th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth)*, San Diego, California, USA, 21-24 May, 2012, pp. 283–290.

[7] M. F. Shiratuddin, and K. W. Wong, Non-contact Multihand Gestures Interaction Techniques for Architectural Design in a Virtual Environment, *International Conference on Information Technology and Multimedia (ICIM)*, Kajang, Malaysia, 14-16 November, 2011, pp. 1–6.

[8] M. Roccetti, and G. Marfia, Recognizing Intuitive Predefined Gestures for Cultural Specific Interactions: An

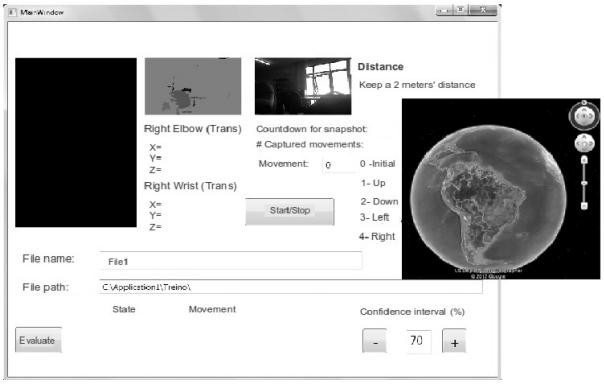

Fig. 8: **Illustration of the prototype with the keyboard simulation module.**

Image-based Approach, *Proceedings of the IEEE Consumer Communications and Networking Conference*, Las Vegas, Nevada, USA, 8-11 January, 2011, pp. 172–176.

[9] L. R. Rabiner, A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, *Proceedings of the IEEE*, February, 1989, vol. 77, no. 2, pp. 257–286.

[10] J. Yamato, J. Ohya, and K. Ishii, Recognizing Human Action in Time-sequential Images using Hidden Markov Model, *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Illinois, 15-18 June, 1992, pp. 379–385.

[11] F. Samaria, and S. Young, HMM-based Architecture for Face Identification, *Image and Vision Computing*, October, 1994, vol. 12, no. 8, pp. 537–543.

[12] F. Bevilacqua, B. Zamborlin, A. Sypniewski, N. Schnell, F. Guédy, and N. Rasamimanana, Continuous Realtime Gesture Following and Recognition, *Proceedings of the 8th International Conference on Gesture in Embodied Communication and Human-Computer Interaction*, Bielefeld, Germany, 2010, pp. 73–84.

[13] J. Davis, and M. Shah, Visual Gesture Recognition, *IEEE Proceedings - Vision, Image and Signal Processing*, April, 1994, vol. 141, no. 2, pp. 101–106.

[14] P. Hong, M. Turk, and T. S. Huang, Gesture Modeling and Recognition using Finite State Machines, *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition*: IEEE Computer Society, Grenoble, France, 28-30 March, 2000, pp. 410–415.

[15] A. F. Bobick, and A. D. Wilson, A State-based Approach to the Representation and Recognition of Gesture, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, December, 1997, vol. 19, no. 12, pp. 1325–1337.

[16] M. Oshita, and T. Matsunaga, Automatic Learning of Gesture Recognition Model using SOM and SVM, *Proceedings of the 6th International Conference on Advances in Visual Computin (ISVC'10) - Volume Part I* , Las Vegas, Nevada, USA, September, 2010, pp. 751–759.

[17] M. Lech, and B. Kostek, Gesture-based Computer Control System Applied to the Interactive Whiteboard, *Proceedings of the 2nd International Conference on Information Technology (ICIT)*, Gdansk, Poland, 28-30 June, 2010, pp. 75–78.

[18] T. Matsunaga, and M. Oshita, Recognition of Walking Motion Using Support Vector Machines, *Proceedings of the 1st International Symposium on Information and Computer Elements (ISICE)*, 2007, pp. 337–342.

[19] Microsoft Research, Programming Guide: Getting Started with the Kinect for Windows SDK Beta from Microsoft Research, *Microsoft Corporation*, Technical Report, 2011.

[20] K. K. Biswas, and S. K. Basu, Gesture Recognition using Microsoft Kinect, *Proceedings of the 5th International Conference on Automation, Robotics and Applications (ICARA)*, Wellington, New Zealand, 6-8 December, 2011, pp. 100–103.

[21] Z. Ren, J. Meng, J. Yuan, and Z. Zhang, Robust Hand Gesture Recognition with Kinect Sensor, *Proceedings of the 19th ACM International Conference on Multimedia*, Scottsdale, Arizona, USA, November-December, 2011, pp. 759–760.

[22] Y. Song, D. Demirdjian, and R. Davis, Continuous Body and Hand Gesture Recognition for Natural Human-Com-puter Interaction, *ACM Transactions on Interactive Intelligent Systems (TiiS) – Special Issue on Affective Interaction in Natural Environments*, March, 2012, vol. 2, no 1, pp.1–28.

[23] M. A. Hearst, Support Vector Machines, *IEEE Intelligent Systems and their Applications – Issue: Computing & Processing (Hardware/Software)*, July-August, 1998, vol. 18, no. 4, pp. 18–28.

[24] A. C. Lorena, and A. C. P. L. F. Carvalho, An Introduction to Support Vector Machines (in Portuguese), *Journal of Applied and Theoretical Informatics (RITA)*, 2007, vol. 14, no. 2, pp. 43–67.

[25] M. Behzad, K. Asghari, M. Eazi, and M. Palhang, Generalization Performance of Support Vector Machines and Neural Networks in Runoff Modeling, *Expert Systems with Applications: An International Journal*, May, 2009, vol. 36, no. 4, pp. 7624–7629.

[26] C. C. Chang, and C. J. Lin, LIBSVM: A Library for Support Vector Machines, 2012. Available in: *http://www.csie.ntu.edu.tw/ cjlin/libsvm*. Accessed in July, 2012.

[27] C. Hsu, C. C. Chang, and C. J. Lin, A Practical Guide to Support Vector Classification, Department of Computer Science National Taiwan University, Taipei 106, Taiwan, 2010. Available in: *http://www.csie.ntu.edu.tw/˜cjlin/papers/guide/guide.pdf*

[28] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Upper Saddle River, NJ, USA, Prentice Hall PTR, Second Edition, March, 1999.

[29] K. C. Louden, *Compiler Construction: Principles and Practice*, Boston, MA, USA: PWS Publishing Co., First Edition, January 1997.

Received: February 2013. Accepted: June 2013.