Electronic Letters on Computer Vision and Image Analysis 0(0):1-7, 2000

Real –Time Hand Tracking and Gesture Recognition for Human-Computer Interaction

Cristina Manresa, Javier Varona, Ramon Mas and Francisco J. Perales

Unidad de Gráficos y Visión por Computador
Departamento de Matemáticas e Informática
Universitat de les Illes Balears
Edificio Anselm Turmeda, Crta. Valldemossa km 7.5
07122 – Palma de Mallorca - España

Received 1 January 2000; revised 1 January 2000; accepted 1 January 2000

Abstract

The proposed work is part of a project that aims for the control of a videogame based on hand gesture recognition. This goal implies the restriction of real-time response and unconstrained environments. In this paper we present a real-time algorithm to track and recognise hand gestures for interacting with the videogame. This algorithm is based on three main steps: hand segmentation, hand tracking and gesture recognition from hand features. For the hand segmentation step we use the colour cue due to the characteristic colour values of human skin, its invariant properties and its computational simplicity. To prevent errors from hand segmentation we add a second step, hand tracking. Tracking is performed assuming a constant velocity model and using a pixel labeling approach. From the tracking process we extract several hand features that are fed to a finite state classifier which identifies the hand configuration. The hand can be classified into one of the four gesture classes or one of the four different movement directions. Finally, using the system's performance evaluation results we show the usability of the algorithm in a videogame environment.

Key Words: Hand Tracking, Gesture Recognition, Human-Computer Interaction, Perceptual User Interfaces.

1 Introduction

Nowadays, the majority of the human-computer interaction (HCI) is based on mechanical devices such as keyboards, mouses, joysticks or gamepads. In recent years there has been a growing interest in a class of methods based on computational vision due to its ability to recognise human gestures in a natural way [1]. These methods use as input the images acquired from a camera or from a stereo pair of cameras. The main goal of these algorithms is to measure the hand configuration in each time instant.

To facilitate this process many gesture recognition applications resort to the use of uniquely coloured gloves or markers on hands or fingers [2]. In addition, using a controlled background makes it possible to localize the hand efficiently and even in real-time [3]. These two conditions impose restrictions on the user and on the interface setup. We have specifically avoided solutions that require coloured gloves or markers

Correspondence to: cristina.manresa@uib.es

Recommended for acceptance by <name>
ELCVIA ISSN: 1577-5097
Published by Computer Vision Center / Universitat Autonoma de Barcelona, Barcelona, Spain

Figure 1: Interactive game application workspace diagram.

and a controlled background because of the initial requirements of our application. It must work for different people, without any complement on them and for unpredictable backgrounds.

Our application uses images from a low-cost web camera placed in front of the work area, see Fig. 1, where the recognised gestures act as the input for a computer 3D videogame. Thus, the players, rather than pressing buttons, must use different gestures that our application should recognise. This fact, adds the complexity that the response time must be very fast. Users should not appreciate a significant delay between the instant they perform a gesture or motion and the instant the computer responds. Therefore, the algorithm must provide real-time performance for a conventional processor. Most of the known hand tracking and recognition algorithms do not meet this requirement and are inappropiate for visual interface. For instance, particle filtering-based algorithms can maintain multiple hypotheses at the same time to robustly track the hands but they need high computational demands [4]. Recently, several works have been presented for reducing the complexity of particle filters, for example, using a deterministic process to help the random search [5]. However, these algorithms only work in real-time for a reduced size hand and in our application, the hand holds most of the image.

In this paper we propose a real-time non-invasive hand tracking and gesture recognition system. In the next sections we explain our method divided in three main steps. First step is hand segmentation where the image region that contains the hand has to be located. In order to make this process it is possible to use shapes, but they vary greatly during the natural motion of hand [6]. Therefore, we choose skin-colour as the hand feature. The skin-colour is a distinctive cue of hands and it is invariant to scale and rotation. The next step is to track the position and orientation of the hand to prevent errors in the segmentation phase. We use a pixel-based tracking for the temporal update of the hand state. In the last step we use the estimated hand state to extract several hand features to define a deterministic process of gesture recognition. Finally, we present the system's performance evaluation results that prove that our method works well in unconstrained environments and for several users.

2 Hand Segmentation

The hand must be localized in the image and segmented from the background before recognition. Colour is the selected cue because of its computational simplicity, its invariant properties regarding to the hand shape configurations and due to the human skin-colour characteristic values. Also, the assumption that colour can be used as a cue to detect faces and hands has been proved in several publications [7][8]. For our application, the hand segmentation has been carried out using a low computational cost method that performs well in real time. The method is based in a probabilistic model of the skin-colour pixels distribution. Then, it is necessary to model the skin-colour of the user's hand. The user places part of his hand in a learning square as shown in Fig. 2. The pixels restricted in this area will be used for learning the model. Next, the selected pixels are transformed from the RGB-space to the HSL-space for taking the chroma information: hue and saturation.

We have encountered two problems in this step that have been solved in a pre-processing phase. The first one is that human skin hue values are very near to red colour, that is, their value is very close to 2π radians, so it is difficult to learn the distribution due to the hue angular nature that can produce samples on both limits. To solve this inconvenience the hue values are rotated π radians. The second problem in using HSL-

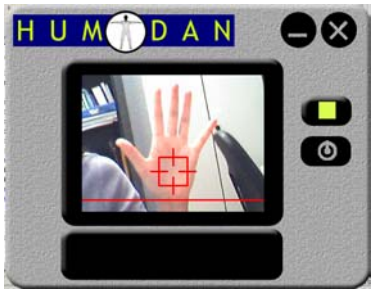

Figure 2: Application interface and skin-colour learning square.

space is when the saturation values are close to 0, because then the hue is unstable and can cause false detections. This can be avoided discarding saturation values near 0.

Once pre-processing phase has finished, the hue and saturation values for each selected pixel are use to infer the model, that is, $\vec{x} = (\vec{x}_1,...,\vec{x}_n)$, where *n* is the number of samples and a sample is $\vec{x}_i = (h_i, s_i)$. As a result of a testing and comparing phase with several statistical models such as mixture of gaussians or discrete histograms, the best results have been obtained using a Gaussian model. The values for the parameters of the Gaussian model (mean, $\bar{x}$ **,** and covariance matrix, Σ) are computed from the sample set using standard maximum likelihood methods [9]. Once they are found, the probability that a new pixel, $\vec{x} = (h, s)$, is skin can be calculated as

$$P(\vec{x} \text{ is skin}) = \frac{1}{\sqrt{(2\pi)^2 |\boldsymbol{\Sigma}|}} e^{(-\frac{1}{2}(\vec{x}-\bar{x})\boldsymbol{\Sigma}^{-1}(\vec{x}-\bar{x})^{\mathbf{T}})} . \quad (1)$$

Finally, we obtain the blob representation of the hand applying a connected components algorithm to the probability image, which groups pixels into the same blob.

3 Tracking

USB cameras are known for the low quality images they produce. This fact can cause errors in the hand segmentation process. In order to make the application robust to these segmentation errors we add a tracking algorithm. This algorithm tries to maintain and propagate the hand state over time.

We represent the hand state in time t, $\vec{s}(t)$, by means of a vector, $\vec{s}(t) = (\vec{p}(t), \vec{w}(t), \alpha(t))$, where $\vec{p} = (p_x, p_y)$ is the hand position in the 2D image, $\vec{w} = (w, h)$, is the size of the hand in pixels, and α is the hand's angle in the 2D image plane. First, from the hand state in time t we built an hypothesis of the hand state, $\vec{h} = (\vec{p}(t+1), \vec{w}(t), \alpha(t))$, for time $t+1$ applying a simple second-order autoregressive process to the position component

$$\vec{p}(t+1) - \vec{p}(t) = \vec{p}(t) - \vec{p}(t-1). \quad (2)$$

Equation (2) expresses a dynamical model of constant velocity. Next, if we assume that at time *t*, M blobs have been detected, $B = \{b_1, \ldots, b_j, \ldots, b_M\}$, where each blob b_j correspond to a set of connected skin-colour pixels, the tracking process has to set the relation between the hand hypothesis, $\vec{h}$, and the observations, b_j, over time.

In order to cope with this problem, we define an approximation to the distance from the image pixel, $\vec{x} = (x, y)$, to the hypothesis $\vec{h}$. First, we normalize the image pixel coordinates

$$\vec{n} = R \cdot \left(\vec{x} - \vec{p}(t+1)\right), \tag{3}$$

where R is a standard 2D rotation matrix about the origin, α is the rotation angle, and $\vec{n} = (n_x, n_y)$ are the normalized pixel coordinates. Then, we can find the crossing point, $\vec{c} = (c_x, c_y)$, between the hand hypothesis ellipse and the normalized image pixel as follows

$$\begin{aligned} c_x &= w \cdot \cos\vartheta \\ c_y &= h \cdot \sin\vartheta \end{aligned}, \tag{4}$$

where ϑ is the angle between the normalized image pixel and the hand hypothesis. Finally, the distance from an image pixel to the hand hypothesis is

$$d(\vec{x}, h) = \|\vec{n}\| - \|\vec{c}\|. \tag{5}$$

This distance can be seen as the approximation of the distance from a point in the 2D space to a normalized ellipse (normalized means centered in origin and not rotated). From the distance definition of (5) it turns out that its value is equal or less than 0 if $\vec{x}$ is inside the hypothesis h, and greater than 0 if it is outside. Therefore, considering the hand hypothesis h and a point $\vec{x}$ belonging to a blob b, if the distance is equal or less than 0, we conclude that the blob b supports the existence of the hypothesis h and it is selected to represent the new hand state. This tracking process could also detect the presence or the absence of the hand in the image [10].

4 Gesture Recognition

Our gesture alphabet consists in four hand gestures and four hand directions in order to fulfil the application's requirements. The hand gestures correspond to a fully opened hand (with separated fingers), an opened hand with fingers together, a fist and the last gesture appears when the hand is not visible, in part or completely, in the camera's field of view. These gestures are defined as *Start*, *Move*, *Stop* and the *No-Hand* gesture respectively. Also, when the user is in the *Move* gesture, he can carry out a *Left*, *Right*, *Front* and *Back* movements. For the *Left* and *Right* movements, the user will rotate his wrist to the left or right. For the *Front* and *Back* movements, the hand will get closer to or further of the camera. Finally, the valid hand gesture transitions that the user can carry out are defined in Fig. 3.

The process of gesture recognition starts when the hand's user is placed in front of the camera field of view and the hand is in the *Start* gesture, that is, the hand fully opened with separated fingers. For avoiding fast hand gesture changes that were not intended, every change should be kept fixed for *5* frames, if not the hand gesture does not change from the previous recognised gesture.

For achieving this gesture recognition, we use the hand state estimated in the tracking process, that is, $\vec{s} = (\vec{p}, \vec{w}, \alpha)$. This state can be viewed as an ellipse approximation of the hand where $\vec{p} = (p_x, p_y)$ is the ellipse center and $\vec{w} = (w, h)$ is the size of the ellipse in pixels. To facilitate the process we define the major axis lenght as M and the minor axis lenght as m.

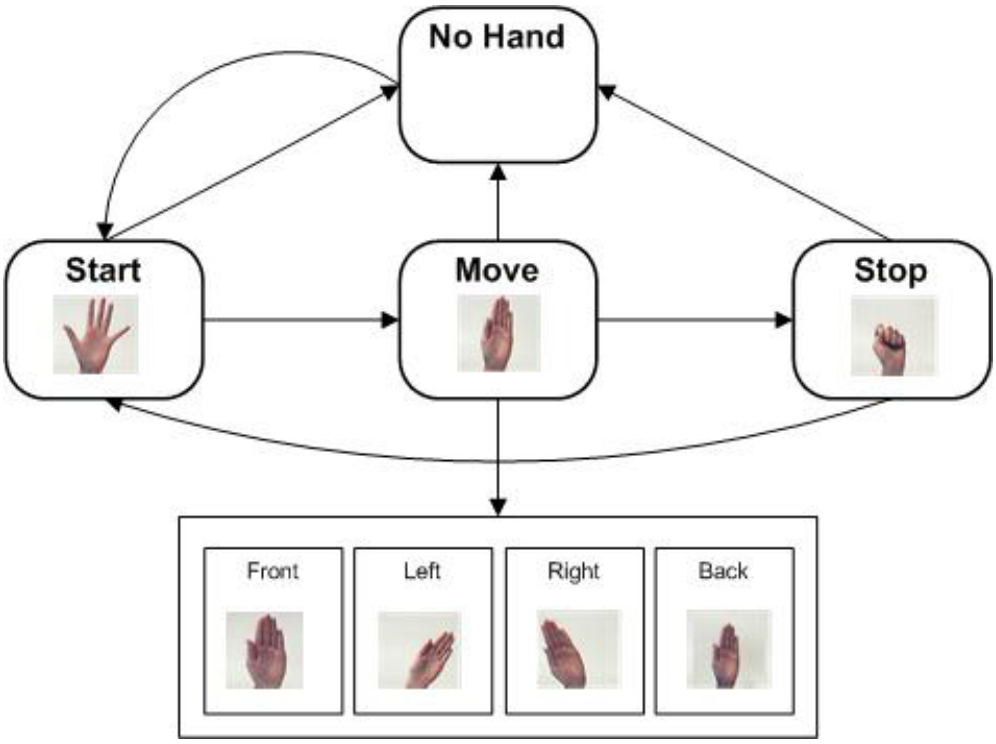

Figure 3: Gesture alphabet and valid gesture transitions.

In addition, from the hand's contour and the hand's convex hull we can calculate a sequence of contour points between two consecutive convex hull vertices. This sequence forms the so-called convexity defect and it is possible to compute the depth of the ith-convexity defect, d_i. From these depths some useful characteristics for the hand shape can be derived like the depth average, $\bar{d}$,

$$\bar{d} = 1/n \sum_{i=0..n} d_i \tag{6}$$

where n is the total number of convexity defects in the hand's contour, see Fig. 4.

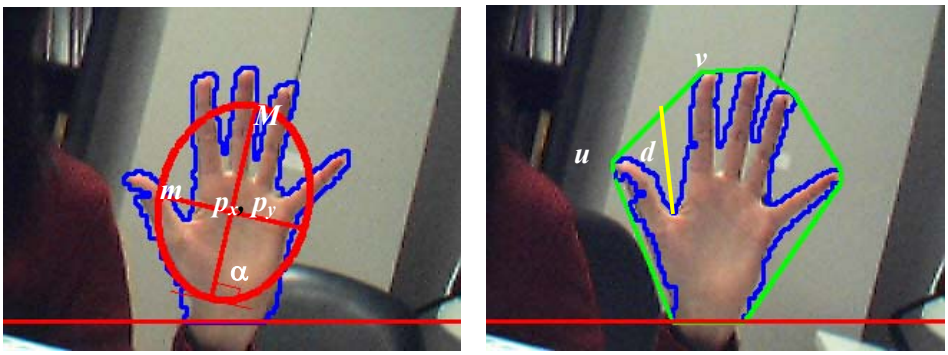

Figure 4: Extracted features for the hand gesture recognition. In the right image, u_i and v_i indicate the start and the end points of the ith-convexity defect, the depth, d_i, is the distance from the farthermost point of the convensity defect to the convex hull segment.

The first step of the gesture recognition process is to model the *Start* gesture. The average of the depths of the convexity defects of an opened hand with separated fingers is larger than in an open hand with no separated fingers or in a fist. This characteristic is used for differentiating the next hand gesture transitions: from *Stop* to *Start*; from *Start* to *Move*; and from *No-Hand* to *Start*. However, first it is necessary to compute the *Start* gesture characteristic, T_{start}. Once the user is correctly placed in the camera field of view with the hand widely opened for learning his skin-colour, the system also computes the *Start* gesture characteristic for the *n* first frames

$$T_{\text{start}} = \frac{1/n \sum_{t=0..n} d(t)}{2}. \tag{7}$$

After the recognition of the *Start* gesture, the most possible valid gesture change is the *Move* gesture. Then, if the current hand depth is less than T_{start} the system goes to the *Move* hand gesture. If the current hand gesture is *Move* the hand directions will be enabled: *Front*, *Back*, *Left* and *Right*.

If the user does not want to move to any direction, he should set his hand in Move state. For the first time that the *Move* gesture appears, the system computes the *Move* gesture characteristic, T_{move}, that is an average of the approximated area of the hand for *n* consecutive frames

$$T_{\text{move}} = 1/n \sum_{t=0..n} M(t) \cdot m(t). \tag{8}$$

For recognising the *Left* and *Right* directions, the calculated angle of the fitted ellipse is used. To prevent non desired jitter effects in orientation, we define a predefined constant T_{jitter}. Then, if the angle of the ellipse that circumscribes the hand, α, satisfies $\alpha > T_{\text{jitter}}$, *Left* orientation will be set. If the angle of the ellipse that circumscribes the hand, α, satisfies $\alpha < -T_{\text{jitter}}$, *Right* orientation will be set.

For controlling the *Front* and *Back* orientations and for returning to the *Move* gesture the hand must not be rotated and the *Move* gesture characteristic is used for differentiating these movements. If $T_{\text{move}} \cdot C_{\text{front}} < M \cdot m$ succeeds the hand orientation will be *Front*. The *Back* orientation will be achieved if $C_{\text{back}} > m/M$.

The *Stop* gesture will be recognised using the ellipse's axis. When the hand is in a fist, the fitted ellipse is almost like a circle and *m* and *M* are practically the same, that is, when $M - m < C_{\text{stop}}$. $C_{\text{front}}, C_{\text{back}}$ and C_{stop} are predefined constants established during the algorithm performance evaluation. Finally, the *No-Hand* state will appear when the system does not detect the hand, the size of the detected hand is not large enough or when the hand is in the limits of the camera field of view. The next possible hand state will be the *Start* gesture and it will be detected using the transition procedure from *Stop* to *Start* explained former on.

Some examples of gesture transitions and the recognised gesture results can be seen in Fig. 5. It is very important a correct learning of the skin-colour. If not, some problems with the detection and the gesture recognition can be encountered. One of the main problems with the use of the application is the hand control for maintaining it in the camera's field of view and without touching the limits of the capture area. This problem has been shown to disappear with user training.

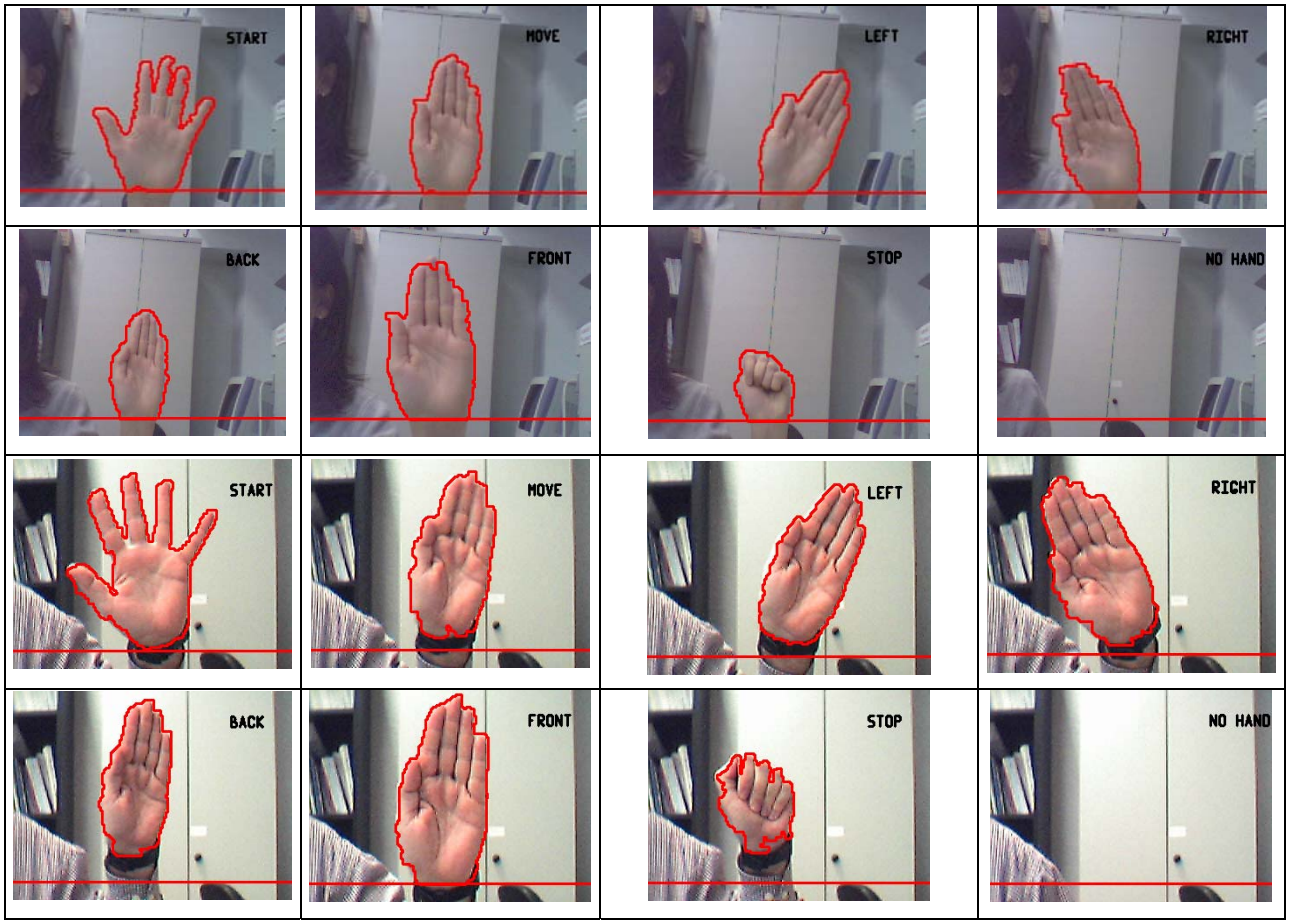

Figure 5: Hand tracking and gesture recognition examples.

5 System's performance evaluation

In this section we describe the accuracy of our hand tracking and gesture recognition algorithm. The application has been implemented in Visual C++ using the OpenCV libraries [11]. The application has been tested on a Pentium IV running at 1.8 GHz. The images have been captured using a Logitech Messenger WebCam with USB connection. The camera provides 320x240 images at a capture and processing rate of 30 frames per second.

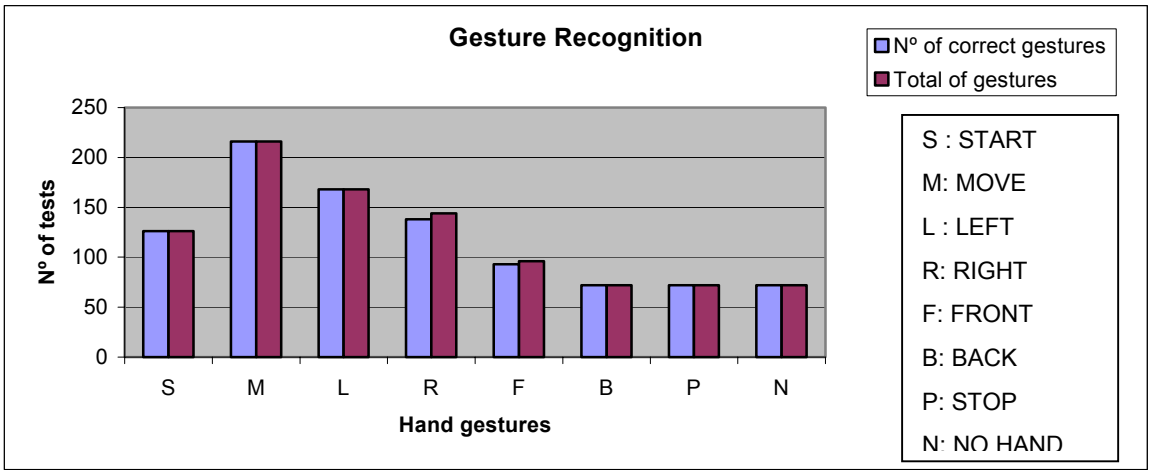

Figure 6: System's performance evaluation results.

For the performance evaluation of the hand tracking and gesture recognition, the system has been tested on a set of 24 users. Each user has performed a predefined set of 40 movements and therefore we have 960 gestures to evaluate the application results. It is natural to think that the system's accuracy will be measured controlling the performance of the desired user movements for managing the videogame. This sequence included all the application possible states and transitions. Figure 6 shows the performance evaluation results. These results are represented using a bidimensional matrix with the application states as columns and the number of appearances of the gesture as rows. The columns are paired for each gesture: the first column is the number of tests of the gesture that has been correctly identified; the second column is the total number of times that the gesture has been carried out. As it can be seen in Fig. 6, the hand recognition gesture works fine for a 99% of the cases.

6 Conclusions

In this paper we have presented a real-time algorithm to track and recognise hand gestures for human-computer interaction within the context of videogames. We have proposed an algorithm based on hand segmentation, hand tracking and gesture recognition from extracted hand features. The system's performance evaluation results have shown that this low-cost interface can be used by the users to substitute traditional interaction metaphors. The experiments have confirmed that continuous training of the users results in higher skills and, thus, better performances.

Acknowledgements

The projects TIC2003-0931 and TIC2002-10743-E of MCYT Spanish Government and the European Project HUMODAN 2001-32202 from UE V Program-IST have subsidized this work. J.Varona acknowledges the support of a Ramon y Cajal fellowship from the spanish MEC.

References

[1] V.I. Pavlovic, R. Sharma, T.S Huang, "Visual interpretation of hand gestures for human-computer interaction: a review", *IEEE Pattern Analysis and Machine Intelligence*, 19(7): 677 – 695, 1997 .

[2] R. Bowden, D. Windridge, T. Kadir, A. Zisserman, M. Brady, "A Linguistic Feature Vector for the Visual Interpretation of Sign Language", in Tomas Pajdla, Jiri Matas (Eds), *Proc. European Conference on Computer Vision, ECCV04*, v. 1: 391-401, LNCS3022, Springer-Verlag, 2004.

[3] J. Segen, S. Kumar, "Shadow gestures: 3D hand pose estimation using a single camera", *Proc. of the Computer Vision and Pattern Recognition Conference, CVPR99*, v. 1: 485, 1999.

[4] M. Isard, A. Blake, "ICONDENSATION: Unifying low-level and high-level tracking in a stochastic framework", *Proc. European Conference on Computer Vision, ECCV98*, pp. 893-908, 1998.

[5] C. Shan, Y. Wei, T. Tan, F.Ojardias , "Real time hand tracking by combining particle filtering and mean shift", *Proc. Sixth IEEE Automatic Face and Gesture Recognition, FG04*, pp: 229-674, 2004.

[6] T. Heap, D. Hogg, "Wormholes in shape space: tracking through discontinuous changes in shape", *Proc. Sixth International Conference on Computer Vision, ICCV98*, pp. 344-349, 1998.

[7] G.R. Bradski, "Computer video face tracking for use in a percep-tual user interface," *Intel Technology Journal*, Q2'98, 1998.

[8] D. Comaniciu, V. Ramesh, "Robust detection and tracking of human faces with an active camera" *Proc. of the Third IEEE International Workshop on Visual Surveillance*, pp: 11-18, 2000.

[9] C.M. Bishop, *Neural Networks for Pattern Recognition*. Clarendon Press, 1995.

[10] J. Varona, J.M. Buades, F.J. Perales, "Hands and face tracking for VR applications", *Computer & Graphics*, 29(2), 2005.

[11] G.R. Bradski, V. Pisarevsky, "Intel's Computer Vision Library", *Proc of IEEE Conference on Computer Vision and Pattern Recognition, CVPR00,* v. 2: 796-797, 2000.