Automatic Hand or Head Gesture Interface for Individuals with Motor Impairments, Senior Citizens and Young Children

Mário Saleiro[1,2], João Rodrigues[1,2], J.M.H. du Buf[1]
[1] Vision Laboratory – Institute for Systems and Robotics (ISR), University of the Algarve, Faro, Portugal
[2] Instituto Superior de Engenharia, University of the Algarve, Faro, Portugal
Phone: +351 289800100, Fax: +351 289888405, e-mail: mariosaleiro@yahoo.com.br, {jrodrig, dubuf}@ualg.pt

Abstract — **Senior citizens, very young children and users with different kinds of impairments are often prevented from enjoying the benefits of latest technologies for assistance, accessibility and usability, often due to unfriendly or expensive interfaces. We present a friendly and inexpensive system that allows one to interact with a computer, using only a few distinct but intuitive gestures which are translated into mouse actions. The gestures can be carried out by the head or by the hand, which are automatically selected, without any kind of prior calibration or special environment. For each kind of user a different frontend is presented, adapted to specific needs, nevertheless access to all the functionalities of the operating systems can be given if requested.**

I. Introduction

During the past years many interface systems based on tracking head and hand gestures have already been developed, providing simple and easy ways for people to interact with computers. However, most methods are based on morphological points of the member that will be used to make the gestures. In addition, most are very specific such that only a limited range of users can truly benefit from them. The ones based on head tracking require in most cases that the user can at least move his head freely, which unfortunately cannot be done by people with more severe motor impairments [1, 2]. For other users which do not have such impairments, the use of a hand to control the computer can be more comfortable and easy, but then a specific method for hand recognition is required [3, 4]. There also are systems which combine head and hand gestures [5, 6], the number of users being more restricted. Other major problems may be the overall cost of the system, a need for prior calibration, and hardware and software installation by a specialized technician.

Apart from people with motor impairments, there are two more groups, senior citizens and children, especially very young ones, who just find computers hard to use either because they cannot read or they cannot understand how to use them. Using the computer by making hand gestures for navigating through a simple graphical interface will surely help them to take advantage of computers' functionalities, and in the case of children it can even be fun to play with the system.

In this paper we describe a non-intrusive, innovative and intuitive system for enhancing computer accessibility for a wide range of individuals, from those with different kinds of impairments to elderly and children. As usual, the system consists of two interfaces. The Human Machine Interface (HMI) is based on a cheap webcam and a normal computer, with automatic selection of very few (5) head or hand movements, which do not need any kind of extra configuration or calibration. The gestures of the head or hand are converted into mouse actions, allowing the user to move the cursor around an attractive, easy to use and personally adaptable Graphical User Interface (GUI), which provides basic but important computer utilities. For more advanced users, full access to the operating system can be given.

In Section 2 we present the implementation of the Human Machine Interface, in Section 3 the Graphical User Interface, and we conclude with a final discussion and lines for future work in Section 4.

II. The Human Machine Interface

As already mentioned above, for the HMI we propose a system with no prior calibration, based on vision and a few head or hand movements. The system has to be fast enough to work in real-time on any computer with any webcam, and it must be able to automatically switch between head and hand without additional intervention of the user. We therefore propose an algorithm which treats head and hand equally, with the movements of both being similar and intuitive.

The system consists of five steps: (a) frame capture and color normalization, for capturing the frames and their normalization in RGB color space in order to minimize the illumination influence of the environment. (b) Then a skin color detection method is applied to each frame, resulting in a binarized frame with some blobs, head/hand, but also with some background noise. (c) Noise filters are applied to create a new binarized frame with well-defined blobs and no noise. After this (d) blob detection is performed and

Acknowledgments: This research was supported by the Portuguese Foundation for Science and Technology (FCT), through the pluriannual funding of the Institute for Systems and Robotics (ISR-Lisbon/IST) by the POS_Conhecimento Program which includes FEDER funds, and by the FCT project SmartVision: active vision for the blind (PTDC/EIA/73633/2006).

data from all recognized blobs are collected for (e) gesture recognition.

A. Frame Capture and Color Normalization

To capture the frames from the webcam we use the Intel Open Source Computer Vision Library (OpenCV, available at http://sourceforge.net/projects/opencvlibrary/). In our case the frames are captured with a size of 640×480. However, in order to cope with different acquisition sizes and also to reduce processing time, every frame denoted by $I(x, y)$ is resampled to a pre-defined size of $M \times N$ (320×240), with $x = \{1, \dots, M\}$ and $y = \{1, \dots, N\}$. Each pixel P_i is defined as (R_i, G_i, B_i) in RGB color space of I, $i = \{1, \dots, M \times N\}$. Figure 1 (top row) shows examples of two resampled frames.

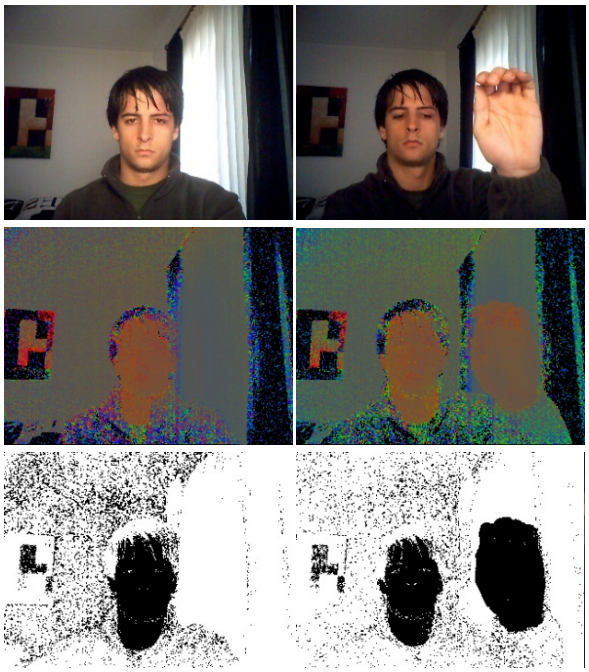

Fig. 1. Resampled 320×240 images (top row), after color normalization (middle row), and the binarized images with skin-colored regions in black (bottom row).

Color normalization [7] is necessary to make the system more robust against changes in the environment illumination or camera errors. Each individual pixel of I is corrected for illuminant geometry independency (i.e., chromaticity), by

$$P_i = \left(\frac{R_i}{R_i + G_i + B_i}, \frac{G_i}{R_i + G_i + B_i}, \frac{B_i}{R_i + G_i + B_i}\right), \quad (1)$$

and the number of colors in the corrected frames (I_c) is kept the same (each component of the new RGB space is multiplied by 255) to cope with the skin color filter as described in the next step. Different color normalization schemes can be used, see e.g. [8]; they will improve the results, but they are more time consuming.

Figure 1 (middle row) shows examples of the normalization process.

B. Skin Color Detection

A few criteria are used to detect pixel clusters with skin color in RGB color space. This method is extremely fast and yields good results for this type of application, because the same criteria have been widely used in other vision systems which require skin detection [7, 9]. Nevertheless, more precise skin detection models can be used, see e.g. [7, 10], but they require more CPU time.

If I_s is the binary frame in which each pixel P_i at position (x, y) in I_c is classified as skin with value 0 (black) or as non-skin with value 255 (white), then

$$I_s(x, y) = \begin{cases} 0, & \varphi[I_c(x, y)] = 1 \\ 255, & \text{else,} \end{cases} \quad (2)$$

where $\varphi = \{R > 95 \wedge G > 40 \wedge B > 20 \wedge (\max\{R, G, B\} - \min\{R, G, B\}) > 15 \wedge |R - G| > 15 \wedge R > G \text{ and } R > B\}$ with $(R, G, B) \in [0,255]$.

Figure 1 (bottom row) shows the results. The major face and hand regions, i.e., the regions of interest (ROIs), have been detected, but also many small regions. The latter can be eliminated by filtering in the next processing step.

C. Noise Filtering

To reduce noise and maintain the ROIs, two different filters are applied. The first one counts the number of black pixels in a region Ω of size $p \times q$, centered at (x, y). If this number is bigger than a threshold ϑ_1, the center pixel turns black:

$$N_b = \sum_{(k,l) \in \Omega} \{I_s(x - k, y - l) = 0\}, \quad (3)$$

$$I_{n1}(x, y) = \begin{cases} 0, & N_b > \vartheta_1 \\ I_s(x, y), & \text{else.} \end{cases} \quad (4)$$

Several environments and users (with glasses, bald, etc.) were tested and the best results were obtained with the size of Ω being 5×5 and a threshold $\vartheta_1 = 12$. As expected, using bigger regions Ω removes more noise, but accuracy of the boundaries of the ROIs will be lost.

The second filter is applied to I_{n1}. The frame is split into non-overlapping square regions $\Psi_{i,j}$ of size $m \times m$. In each square the number of white pixels is counted, and if there are more white pixels than a threshold ϑ_2, the whole square turns white; otherwise it will turn black:

$$N_w(\Psi_{i,j}) = \sum_{(k,l) \in \Psi_{i,j}} \{I_{n1}(x - k, y - l) = 255\}, \quad (5)$$

$$I_{n2}(x - k, y - l) = \begin{cases} 255, & N_w(\Psi_{i,j}) > \vartheta_2 \\ 0, & \text{other,} \end{cases} \quad (6)$$

with $(k, l) \in \Psi_{i,j}$.

The same environments and users were tested and the best values were $m = 4$ and $\vartheta_2 = 3$.

The goal of the second filter, in addition to removing more noise, is to turn the different image regions more homogenous, i.e. more suitable for the ROI detection algorithm, as can be seen in Fig. 2. In addition, since the image is split into square regions and all the pixels of a region will have the same value (Eq. 6), the number of pixels to be processed in the next steps is reduced: only one pixel for each square is needed; this will also speed up ROI detection (next section).

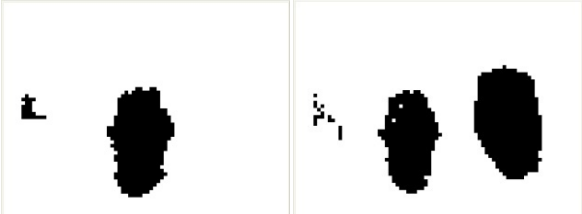

Fig. 2. Noise-filtered frames from Fig. 1.

D. Blob Detection

The next step is the detection of the ROI(s), i.e., blob(s) in the subsampled version of I_{n2}. This is a crucial step since most blob-detection algorithms require a lot of CPU time. Blob detection is done in four steps:

(a) Each frame is analyzed row by row, from top to bottom. If three adjacent black pixels are found in a row, the start of a line-blob is found. After a start has been found, if three adjacent white pixels are detected, the end of the line-blob is found and the minimum and maximum coordinate in x are stored together with the row's y coordinate.

(b) When a row has been processed, if a line-blob exists at a similar position in all of the previous 4 rows, that blob is expanded to include the new line-blob. If not, a new blob is created.

(c) After all blobs have been detected, their sizes are computed and they are stored in descending size order. Only the biggest blob in a frame will be selected.

(d) Before final blob selection, the array of blobs is filtered in order to eliminate small blobs which may exist inside big blobs.

Figure 3 shows the detected blobs in the frames presented in Fig. 1. As can be seen, only one blob was detected in the left frame and two in the right one. Due to steps (a) and (b) of the algorithm and a minimum blob size test (more than 25 connected elements), small blobs will be discarded automatically in the detection process; this was what occurred with the two small left blobs in Fig. 2. Also, if more than one blob have been detected, only the biggest one will be selected, in Fig. 3 represented by the green box. All other detected blobs are marked as red boxes, until they become the biggest (predominant) blob.

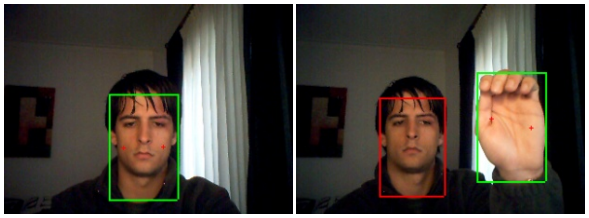

Fig. 3. Two frames with detected blobs. In the right image the head is also detected, but only the hand is selected (green box).

E. Gesture Recognition

Gesture recognition starts with an initialization step which is fundamental to the system. During initialization the user must keep his head or hand in the webcam's field of view, at a static position that will be the base position in using the interface (GUI), as shown in Fig. 3. After initialization, translations of the head or hand are permitted, but they must stay in the field of view of the camera.

The initialization step consists of computing the average dimensions of the biggest (initial) blob from frame 10 to 25 (about 1 second), applying the algorithm described in the previous section. The first 10 frames are discarded due to hardware initialization and calibration of the webcam (this process is fully automatic).

At the end of the initialization step the ratio P between the height h and the width w of the biggest blob box is computed,

$$P = \frac{h}{w}, \tag{7}$$

also the position of the center of the blob box (x, y). These two properties permit a very simple tracking of the biggest blobs in the subsequent frames.

For tracking, only the two biggest blobs in subsequent frames will be considered. The Euclidean distance D from the center of each blob in the present frame (x_{new}, y_{new}) to the center of each blob in the previous frame (x_{old}, y_{old}) is computed,

$$D = \sqrt{(x_{old} - x_{new})^2 + (y_{old} - y_{new})^2}. \tag{8}$$

The blob with the smallest distance is selected for tracking and this process is repeated every frame. This allows the system to track the head or the hand selected in the initialization step so that the user can move the head or hand (2D translations) with some flexibility while using the interface. Since box ratios P are also updated, the user can move back and forth, provided the blob remains the biggest; otherwise the other blob will be selected. In this case the system alerts the user by beeping, starting a new initialization (as the initial one, it takes about 1 second).

The goal of this HMI in this application, as explained in the Introduction, is to control the cursor. To this purpose, the following movements are defined: standby (base), up, down, left and right (the 5 basic

movements), plus select and help (two complementary movements). The movements of head or hand should be as natural as possible, and in the case of the head they should be possible without any other movement of the body.

Four movements are shown in Fig. 4, of the head (left frames) and of the hand (right frames) with, top to bottom: up, down, left and right. Taking into account that the face and hand shown in Fig. 3 are at base position (face straight forward; hand showing only palm), the gestures that correspond to the up (move head up; hand fully open) and down (move head down; hand closed) movement of the cursor are detected by comparing the blob's ratio P to an upper (U_{thr}) and a lower threshold (L_{thr}). The latter are determined from the base ratios P_b computed after the calibration step:

$$U_{thr} = 1.10 \times P_b; \quad (9)$$

$$L_{thr} = 0.9 \times P_b. \quad (10)$$

If the ratio of a tested blob P_t is larger than the upper threshold U_{thr}, the cursor moves up, and if it is smaller than the lower threshold L_{thr}, the cursor moves down. The constants presented in Eqns 9 and 10 were established experimentally, and the same values are applied to head and hand movements. Examples of these two gestures are shown in Fig. 4, the top two rows.

The left and right commands are activated when the user leans the head or hand to the left or to the right; see the bottom two rows in Fig. 4. To detect these gestures, two vertical lines, at distance $w/6$ and $5w/6$ inside the blob's box are considered. The average position of all black pixels on each of these lines is computed, and the two resulting positions are used to detect the "left" and "right" gestures: when the user leans his head or hand to one side, the average positions will go up and down relative to the middle of the box ($h/2$). Examples of those positions, indicated by a "+" symbol, can be seen in Fig. 4 and in more detail in 5.

A minimum vertical distance (*MVD*) between both positions was determined experimentally, such that small lateral movements are ignored, i.e.,

$$MVD = 0.2 \times h. \quad (9)$$

Left and right movements are detected when (a) the vertical distance between the two positions is larger than the *MVD*, and (b) one of the two positions is higher than $h/2$. The latter position determines the side of the movement: left or right.

Mouse selections ("clicks") are done by (a) holding the face or hand in the base position, (b) then quickly change to down position, and (c) return to base position. At most three frames with the down-gesture must be detected, and when the base position is detected again a mouse selection is done at the current position of the cursor.

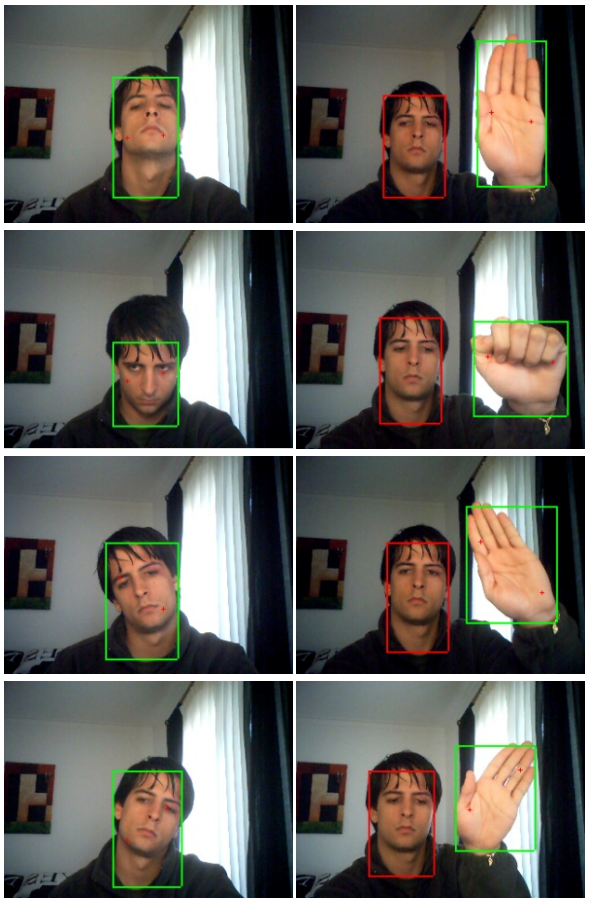

Fig. 4. Corresponding gestures of the head (at left) and hand (at right). Top to bottom: up, down, left and right gestures.

Finally, the help movement consists of leaning the head or hand to the left and right three times in any sequence. As in the selection gesture, a maximum of 3 frames per gesture is applied.

After detecting the movements (commands), no specific functions are required. We simply use the functions available in each operating system, for example the Windows API has SetCursorPos() and mouse_event().

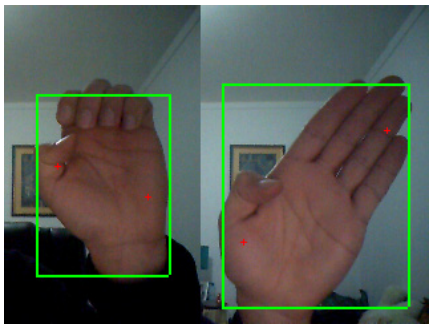

Fig. 5. The two reference points (+) used for detecting right and left movements. The left image shows the base position, the right one shows movement to the right.

III. Graphical User Interface

The Human Machine Interface presented above is completely independent of the Graphical User Interface, and it can be used directly by the operating system. Nevertheless, our goal is to create a frontend GUI which allows the user to interact with the computer in an easier way, especially users who cannot read or with various impairments.

The GUI was developed using the Adobe Flex 3 SDK but other frontend interfaces could be used, for instance Microsoft's Silverlight or Visual Studio.

In order to enhance accessibility, the application is composed by modules. These consist of different functions that the user may wish to use. Modules which are more useful for a specific user can be selected by configuring the frontend interface. Figure 6 shows some examples of implemented modules.

The configuration of the frontend interface is done by dragging-and-dropping of selected modules. This can be done either by someone who knows how to drag and drop items in a configuration menu using the mouse, or by the user himself by selecting the items in the menu. When the configuration is finished, from that moment on the computer will initialize the application automatically. Future changes of the configuration are allowed in order to improve accessibility and to adapt the frontend interface to new necessities of the user.

Some pre-defined frontends are also available; Fig. 7 shows, from top to bottom, frontends for elderly, for people with motor impairments, and for young children.

Fig. 6. Examples of available modules.

Examples of useful modules are:

- webcam chat – The user can have video conversations with other people helping him in social insertion, or calling friends or family when any help is needed;
- phone – Allows the user to call people by dialing the number or by clicking the number in a quick call list, as shown in Fig. 8;
- SOS – Can be used when the user needs urgent help. The user can also make the help gesture to automatically activate this module;
- shopping cart – The user can request essential products like food or hygiene products to be delivered to his home;

Fig. 7. Examples of pre-defined frontend GUIs for (top to bottom) elderly, people with motor impairments and young children.

- didactic games – Especially for children (or physiotherapy "games" for people with motor impairments) so that they can increase their knowledge and have some fun.
- books – People with motor impairments who cannot read a book because they cannot turn pages can use the "e-books" module.

Modules to navigate pre-defined websites or to grant full access to Internet Explorer can also be provided. Another important feature is the ability to gain full control over the operating system by clicking the respective module, or a virtual keyboard can be used, e.g., the one provided by Microsoft Windows. The symbols of all modules are easily recognized by any user and they are big in size, making navigation easier for people who cannot read or who have vision problems (at this point it is important to stress that the precision of the HMI for the selection of any icon is exactly the same as when using a normal mouse).

Figure 9 shows screenshots of the environment, with the option of the HMI window being shown on top of the active GUI, in this case the frontend for children. The upper screenshot shows the HMI with the hand in the base position, the lower with the hand in the "right" position. At the bottom-right corner of the

window, a small icon shows the actual hand and head gesture: a square and a right-arrow.

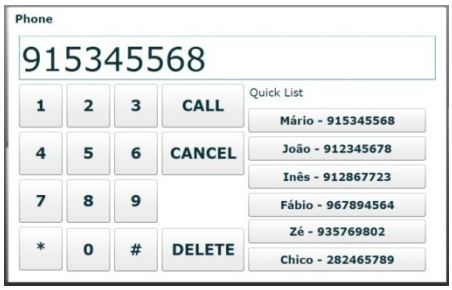

Fig. 8. The phone module.

IV. Conclusions and Future Work

We presented a new system which automatically selects between head and hand, and which uses the same movements and algorithms for both head and hand to detect and execute commands, in this particular case for substituting the mouse. It is truly inexpensive, since only a common computer with a cheap webcam is needed, with no need for any kind of environmental calibration.

The frontend interface is user-friendly, making it possible to anyone who does not know how to read to use the computer and to benefit from its basic capabilities and functionalities. However, as has been mentioned, the user is not limited to the basic capabilities provided by the frontend interface; full access to the operating system with all applications can be granted.

In terms of performance, a notebook with Intel Centrino processor running at 1.73 GHz with 2GB of RAM and a 1.3 Megapixel webcam can process about 15 frames per second, which is more than enough for the system working in real-time.

During tests the system worked well in most environments, but in some specific conditions such as a very low illumination level it proved hard or impossible to use. The use of a webcam implies that a minimum (but normal) illumination level is required. We also verified that the initialization step is fundamental for the system to work as pretended. A wrong initialization can make some movements hard or impossible to detect. Finally, if there is a big area with the same skin color, for instance a background made of wood with almost the same color, then the system will select this area as the main region.

All limitations discussed above have simple solutions, like increasing the illumination level or changing the background, but also more complex solutions are possible. To solve the problem of head/hand detection and their movements, Rodrigues and du Buf presented models for facial landmark detection (eyes, nose, mouth), face recognition, object recognition (we can consider in this application the hand as a specific object), also to select only salient areas of the frames by using Focus-of-Attention models [8, 11, 12]. However, such solutions were not used because the basic idea was to create a cheap system which work on any computer, and the use of such models implies a much more powerful computer.

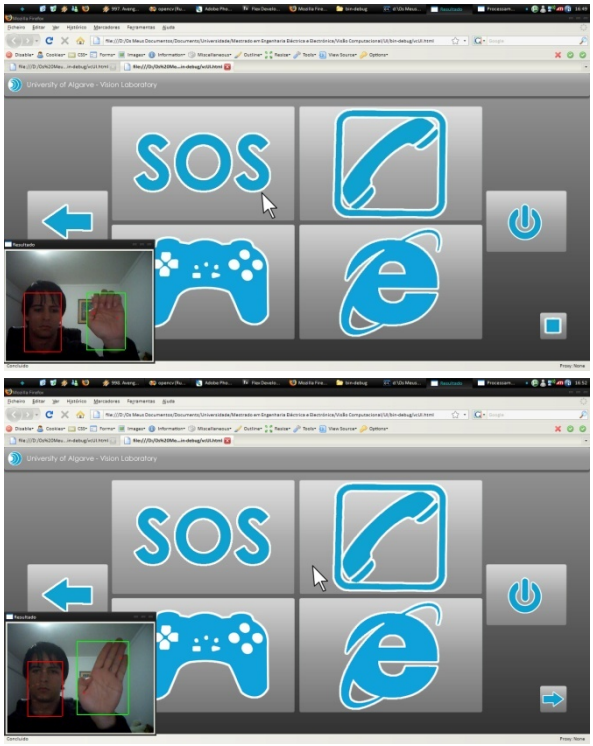

Fig. 9. Screenshots of the environment, with the option of showing the HMI window on top of the GUI. At the top the base position, at the bottom the hand gesture to move the cursor to the right.

The limitation due to the skin-color algorithm can be solved by using e.g. probabilistic models based on a large number of skin samples [13], or only samples of a specific user, but these will require more CPU time.

In future work[1], three points will be focused. A major priority is to port the system to mobile phones and PDAs, such that these devices can be used by anyone, providing access to this technology to people with impairments and to those who think that these devices are too complex or confusing. In this case we will focus not so much on moving the mouse, but on some combinations of movements that will give access to some specific features. The frontend interface can also be improved by the development of new modules which address other functionalities. Finally, the HMI can be interfaced with "The amateur painter" [14], a tool for converting photographs into artificial paintings, such that fine arts edutainment can become available to a much wider audience.

[1] A website with a demonstration video showing different kinds of users including people with different impairments will be available very soon.

REFERENCES

[1] V. Graveleau, N. Mirenkov, G. Nikishkov, "A head-controlled user interface," *Proc. 8th Int. Conf. on Humans and Computers*, Aizu-Wakamatsu, 31 Aug-2 Sept, pp. 306-311, 2005.

[2] R. Kumar, A. Kumar, "Black Pearl: An alternative for mouse and keyboard," *J. on Graphics, Vision and Image Processing*, Vol. 8 (III), pp. 1-6, 2008.

[3] J. Hannuksela, M. Barnard, P. Sangi, J. Heikkilä, "Adaptive motion-based gesture recognition interface for mobile phones," *Proc. Computer Vision Systems*, Springer LNCS 5008, pp. 271-280, 2008.

[4] S. Waldherr, S. Thrun, R. Romero, "A gesture-based interface for human-robot interaction," *J. Autonomous Robots*, Vol. 9 (2), pp. 151-173, 2000.

[5] M. Hasanuzzaman et al., "Face and gesture recognition using subspace method for human-robot interaction," *Proc. Advances in Multimedia Information Processing*, Springer LNCS 3331, pp. 369-376, 2005.

[6] H. Kang, M. Ju, "Face and gesture-based interaction for displaying comic books," *Proc. Advances in Image and Video Technology*, Springer LNCS 4872, pp. 702-714, 2007.

[7] V. Vezhnevets, V. Sazonov, A. Andreeva, "A survey on pixel-based skin color detection techniques," *Proc. Graphicon*, Moscow, Russia, pp. 85-92, 2003.

[8] J. Martins, J. Rodrigues, J.M.H. du Buf, "Focus of attention and region segregation by low-level geometry, *Proc. Int. Conf. on Comp. Vision Theory and Applications*, Lisbon, Portugal, February 5-8, Vol. 2, pp. 267-272, 2009.

[9] U. Ahlvers, R. Rajagopalan, U. Zölzer, "Model-free face detection and head tracking with morphological hole mapping," *Proc. 13th Europ. Signal Proc. Conf.*, September, pp. 1-4, 2005.

[10] J. Agbinya, B. Lok, Y. Sze Wong, S. da Silva, "Automatic online porn detection and tracking," *Proc. 12th IEEE Int. Conf. on Telecommunications*, May, pp. 1-5, 2005.

[11] J. Rodrigues, J.M.H. du Buf, "Multi-scale keypoints in V1 and beyond: object segregation, scale selection, saliency maps and face detection", *BioSystems*, Vol. 86, pp. 75-90, 2006, doi:10.1016/j.biosystems.2006.02.019.

[12] J. Rodrigues, J.M.H. du Buf, "Multi-scale lines and edges in V1 and beyond: brightness, object categorization and recognition, and consciousness", *BioSystems*, Vol. 95, pp. 206-226, 2009, doi:10.1016/j.biosys-tems.2008.10.006.

[13] A. Amine, S. Ghouzali, M. Rziza, "Face detection in still color images using skin color information", *IEEE Tr. PAMI*, vol. 24, pp. 696-706, 2002.

[14] D. Almeida, S. Nunes, J. Carvalho, V. Brito, J. Rodrigues, J.M.H. du Buf, "Fine arts edutainment: the amateur painter", *Proc. Int. Conf. Computer Graphics Theory and Applications*, Funchal, Portugal, January 22-25, pp. 262-267, 2008.