A Fast and Stable Facial Interface Using Integral Projections[1]

G. García-Mateos and S. Fructuoso-Muñoz

Dept. de Informática y Sistemas, Universidad de Murcia, 30100 Espinardo, Murcia, Spain
e-mail: ginesgm@um.es; sergiofr@ono.com

Abstract—Integral projections are a useful technique in many computer vision problems. In this paper, we present a perceptual interface which allows us to navigate through a virtual 3D world by using the movements of the face of human users. The system applies advanced computer vision techniques to detect, track, and estimate the pose of the user's head. The core of the proposed approach consists of a face tracker, which is based on the computation, alignment, and analysis of integral projections. This technique provides a robust, accurate, and stable 2D location of the face in each frame of the input video. Then, 3D location and orientation of the head are estimated, using some predefined heuristics. Finally, the resulting 3D pose is transformed into control signals for the navigation in the virtual world. The proposed approach has been implemented and tested in a prototype, which is publicly available. Some experimental results are shown, proving the feasibility of the method. The perceptual interface is fast, stable, and robust to facial expression and illumination conditions.

DOI: 10.1134/S1054661807040049

1. INTRODUCTION AND RELATED RESEARCH

In the last few years, there has been increasing interest in problems related to the so-called "looking-at-people" area [1], like those associated with biometrics, perceptual interfaces, and video surveillance. In particular, the emerging field of perceptual human interfaces aims at the development of nonintrusive and more natural ways of man/machine interaction [1]. Most research has been done involving different human features as input. Human faces [2, 3, 4, 9, 13] and heads [14, 15, 17] are among the most frequently used features. But we can also find methods that are based on gaze tracking [8], mouth and lip reading [10, 12], and body tracking [15]. To be of practical usage, such interfaces should be very efficient, stable, nonintrusive, robust, and inexpensive. Accuracy is also desirable, although it is not essential in many applications such as entertainment software.

Potential applications of perceptual interfaces in the future are far beyond their present uses, which are reduced owing to practical limitations. In this sense, face-based interfaces are amongst the most promising new ways of man/machine interaction. The human face constitutes a natural and feasible mode of input and can provide computers with a wide range of information about the user: 3D location of the face, orientation, movement, and even the facial expression of the human user.

Facial interfaces benefit from the existing techniques for face detection and tracking. Many different approaches can be found in the literature to tackle both problems [6]. Most of them are based on color [2, 3, 9, 13], neural networks [5], eigenspaces [16], a combination of simple classifiers [11], and others. In some cases, color is used together with other methods, because it allows a fast and robust—but inaccurate—location of the face. The perceptual interface described in this paper is based on the technique of integral projections [3, 4]. The results of the interface are applied to navigation in a virtual 3D environment, specifically created for this purpose. In this environment, the user adopts the role of a character that can move freely through the virtual world, in a similar way to a first-person game. A prototype is publicly available, as we will refer to later. Figure 1 shows a global view of the system.

A camera connected to the computer is situated in front of the user. The system detects and tracks the location of the face in the video sequence, extracts information concerning the movement of the face and its 3D pose, and finally translates this data into a movement in the virtual world. This process is carried out in real time on an off-the-shelf PC, using a low-cost camera for the acquisition.

In the following sections, we will describe the facial interface, centering our discussion on the perceptual part of the problem. In Section 2, we briefly present the face detection and tracking processes using integral projections. Section 3 explains how the 3D pose is estimated using the results of the face tracker. Some simple but effective heuristics are defined for this purpose. In Section 4, we describe the virtual world and detail the translation of the 3D pose into control signals for the movement in this world. Some experimental results are shown in Section 5. Finally, Section 6 summarizes the main points of the proposed method and extracts relevant conclusions.

[1] The text was submitted by the authors in English.

Received April 21, 2005

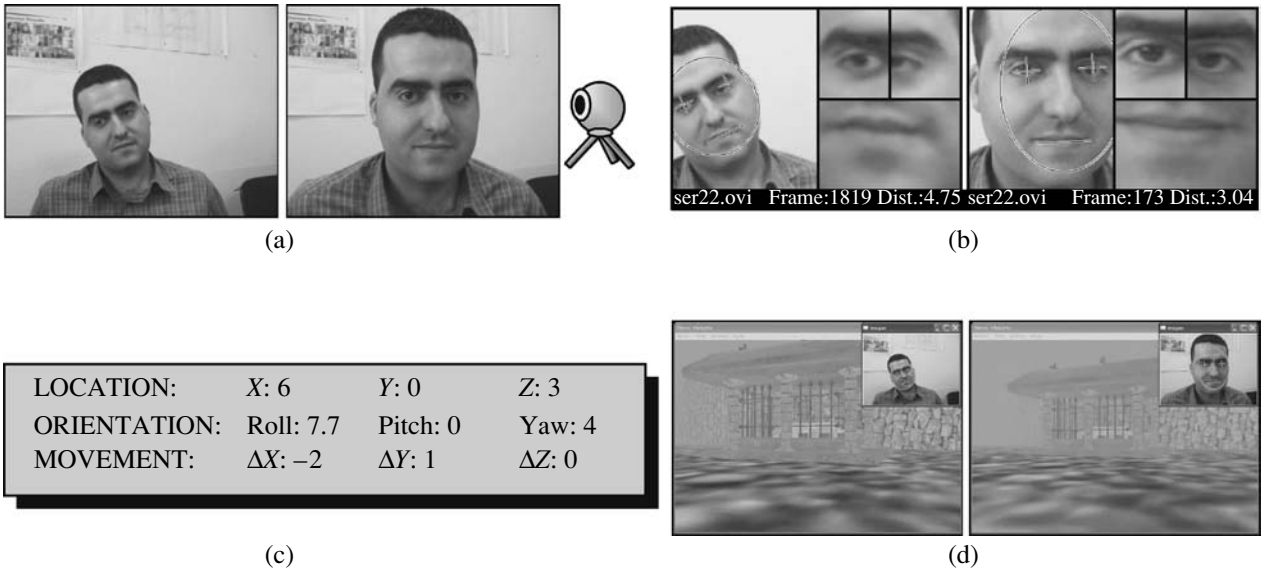

Fig. 1. Overview of the perceptual interface controlled by the user's face. (a) Input images captured from the camera (in this case, a FireWire camera). (b) Faces detected and facial features located by the tracker. (c) Estimated 3D pose parameters. (d) Movement translated into the virtual 3D environment.

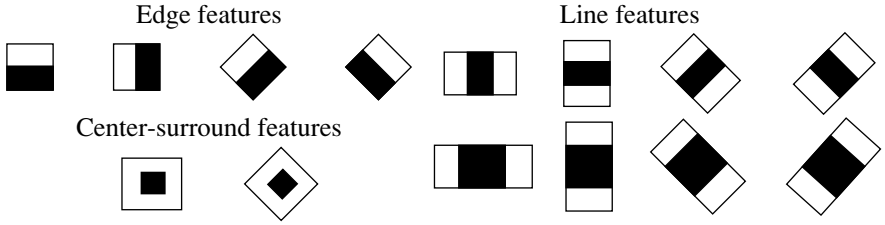

Fig. 2. Haar-like features used by the face detector [11].

2. FACE DETECTION AND TRACKING WITH INTEGRAL PROJECTIONS

Obviously, face detection and tracking are previous and essential problems in the design and construction of a perceptual interface based on human faces. Before any facial information can be extracted from the images, a face has to be properly detected and located in each frame of the input sequence. Many face detectors have been proposed in the literature [6], as described in Section 1. We have used the method presented in [11], which is called "cascade of boosted classifiers working with Haar-like features." This method is based on a set of elemental classifiers, taking as input the results of simple feature detector filters. These simple feature—the Haar-like features—are like those shown in Fig. 2. In a second step, the classifiers are combined using classification trees, trained with the well-known AdaBoost algorithm [11]. We will not describe this face detector in more detail, because it is beyond the scope of this paper.

Face detection is a rather costly process, but fortunately, it only has to be performed in the first frame of the sequence or whenever the face is lost. In the following frames, a tracker is responsible of updating the location of the face in the images.

Integral projections have proved to be a very useful technique to tackle the problem of face tracking, as discussed in [3, 4]. In addition, they provide some valuable additional information that can be used in pose estimation, as we will see in Section 3.

Intuitively, the integral projection of an image is the average of gray levels along a row or column of pixels. Let R be a region of interest in an image i. The vertical integral projection of R, denoted by $P_{V,R}$, is given by

$$P_{V,R}(y) = \text{Average}_{\forall(x,y)\in R}(i(x,y)). \quad (1)$$

Similarly, the horizontal integral projection of R, denoted by $P_{H,R}$, is given by

$$P_{H,R}(x) = \text{Average}_{\forall(x,y)\in R}(i(x,y)). \quad (2)$$

When integral projections are applied on human faces, typical patterns of dark and bright regions appear. Moreover, these patterns of projections are stable and robust for the same individual along a video sequence [4]. We can make use of this property to construct a face tracking system. The key problem now is the alignment of projections: shift and scale two projec-

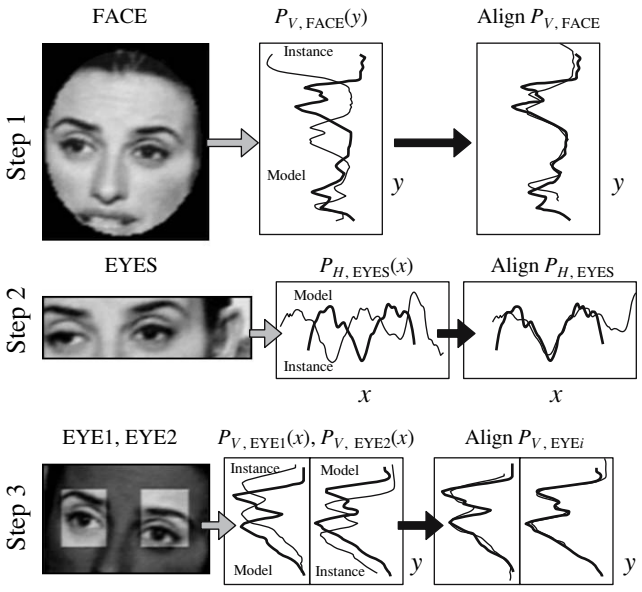

Fig. 3. A three-step face tracking process using integral projections. Left: input regions for each step. Center: projection models (thick green lines) and instances (thin red lines) before alignment. Right: the same projections after alignment.

tions so that they have a maximum overlap. An fast and stable alignment algorithm and the tracking process that uses it are described in detail in [4]. Here, we will give just a brief overview of this method.

Face tracking starts from the location of the face in the previous frame. An elliptical region of interest, named FACE, is taken in the current frame using that location. The process is then decomposed into three main steps:

1. Vertical alignment. First, the vertical integral projection of the expected location of the face is computed, i.e., $P_{V,\,FACE}$. This projection is aligned with respect to a model of vertical projection of the face, $M_{V,\,FACE}$, obtaining the movement and scale in the vertical direction. The model is computed from the first frame of the sequence.

2. Horizontal alignment. After step 1, the region with the eyes, *EYES*, can be located and segmented. Its horizontal projection is computed, i.e., $P_{H,\,EYES}$. Then, it is aligned with an adequate model, computed from the first frame. Thus, we obtain the horizontal movement and scale of the face.

3. Orientation alignment. Here orientation is considered in a planar sense, i.e., with respect to the plane of the image. In this step, the alignment is carried out by independently estimating the height of both eyes. First, a region roughly corresponding to each eye is selected, *EYE1* and *EYE2*. Then, we compute $P_{V,\,EYE1}$ and $P_{V,\,EYE2}$ and align them with the corresponding models.

The structure of the process is depicted and summarized in Fig. 3. After face tracking, we know the location of the left and right eyes in the image, $(x_{\mathrm{eye1}}, y_{\mathrm{eye1}})$ and $(x_{\mathrm{eye2}}, y_{\mathrm{eye2}})$, respectively, and the mouth $(x_{\mathrm{mouth}}, y_{\mathrm{mouth}})$. We also have aligned versions of $P_{V,\,FACE}$ and $P_{H,\,EYES}$ that will be used in the following stages.

3. 3D POSE ESTIMATION

As we have described in Section 2, the tracking process provides the location of the face and its main facial features (eyes and mouth) along a video sequence. Using this information, the 3D pose and movement of the user's head are computed. The generic problem of pose estimation—i.e., obtaining values for 3D location (x, y, z) and angle (*roll*, *pitch*, *yaw*) of an arbitrary object—is computationally complex. However, we are not interested in a precise pose calculation, but in an estimation suitable for perceptual interface usage.

For example, a simple but effective method to estimate the x and y components of the 3D location is just to take the average of the location of the eyes and mouth. That is,

$$\begin{aligned} x &= (x_{\mathrm{eye1}} + x_{\mathrm{eye2}} + x_{\mathrm{mouth}})/3, \\ y &= (y_{\mathrm{eye1}} + y_{\mathrm{eye2}} + y_{\mathrm{mouth}})/3. \end{aligned} \tag{3}$$

Thus, some heuristics have been defined to obtain a fast approximation of all the pose parameters. The rest of the values, z, *roll*, *pitch*, and *yaw*, are explained in the following subsections.

3.1. Depth Estimation

The depth, z, or distance of the user's head to the camera, is inversely proportional to the size of the head in the image. This holds true as long as the face is considered a rigid object, which might not always be exact (for example, because of facial expressions). The size of the head is approximated with the distance from the left to right eye.

Let d_{inic} denote the distance from the left to right eye in the first frame of the sequence. The depth is given by

$$z = \frac{d_{\mathrm{inic}}}{\sqrt{(x_{\mathrm{eye1}} - x_{\mathrm{eye2}})^2 + (y_{\mathrm{eye1}} - y_{\mathrm{eye2}})^2}}. \tag{4}$$

Some samples of depth estimation are shown in Fig. 4. Observe that these distances are relative values (relative to the initial depth).

3.2. Roll Estimation

The estimation of roll is straightforward. Since roll involves a rotation along the image plane, it can be computed via the perceived angle of the face. In particular, we have used the angle of the line passing through

both eyes with respect to the horizontal axis. It can be computed as follows:

$$roll = \arctan\frac{y_{eye2} - y_{eye1}}{x_{eye2} - x_{eye1}}. \quad (5)$$

The values obtained with Eq. (5) are correct as long as the user is facing the camera, which should be the most common situation. However, when this rotation is combined with other kinds of rotation (pitch or yaw), the values given by Eq. (5) are not very reliable. Nevertheless, in terms of our application, this effect is usually negligible, as we have verified in the experiments. Figure 5 contains some samples of roll estimation with the formula given in (5).

3.3. Pitch Estimation

Pitch cannot be precisely estimated just using the location of the eyes and mouth. Instead, we use the aligned projections to extract some additional information. When the user raises or lowers his/her head, the vertical projection of the whole face, $P_{V,FACE}$, changes smoothly, as we can observe in Fig. 6. Across this change, some parts of the projection increase and some others decrease.

We have defined the following ad hoc rule: the estimated pitch is proportional to the value, v, of projection $P_{V,FACE}$ at a certain point between the maximum corresponding to the eyes and the minimum of the nose. We denote this fixed point by fp. It is marked with a line in Fig. 6. Let v_{inic} be the value of v in the first frame. The pitch is computed as

$$pitch = P_{V,FACE}(fp) - v_{inic}. \quad (6)$$

This simple heuristic measure has a viable explanation: when the face is looking upward, the nostrils are more and more visible, making projections darker at that point. The effect of illumination conditions which could bias the measure is reduced if projection $P_{V,FACE}$ has been correctly aligned.

3.4. Yaw Estimation

As before, yaw is also approximated with a fast heuristic method which might not be applicable in a generic problem. In this case, we consider the horizontal projection of the region of the eyes, $P_{H,EYES}$. This projection usually contains two local maxima corresponding to each eye and a local minimum between them. When the user looks to the left or to the right, the horizontal projection changes accordingly. This effect is shown in Fig. 7.

Therefore, the yaw can be estimated by searching for the relative position of the minimum between the eyes in these projections, i.e., the location of the arrows in Fig. 7. Let m_{inic} be the initial location of this minimum in the first frame; then, the yaw is given by

$$yaw = \arg\min_{\forall x \in \{min,\, max\}} P_{H,EYES}(x) - m_{inic}. \quad (7)$$

The values obtained from Eq. (7) are given in pixels. They could be divided by the eye-to-eye distance to get relative values.

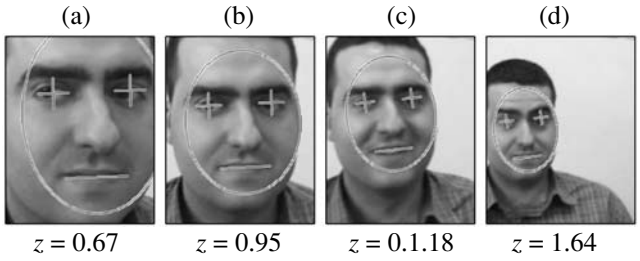

Fig. 4. Depth computation. Estimated values (relative to the initial distance of the face to the camera) are shown below the pictures.

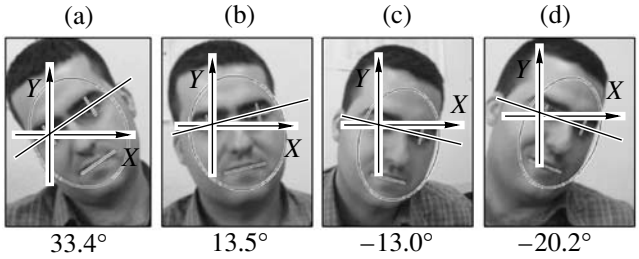

Fig. 5. Roll computation. Estimated angles are shown below the pictures.

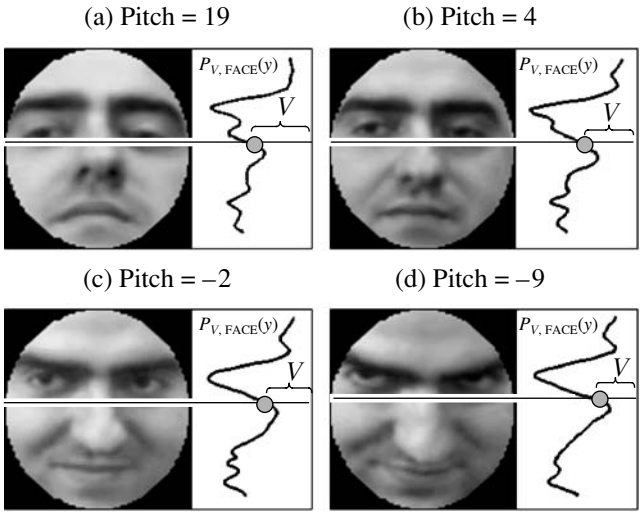

Fig. 6. Pitch computation. Vertical integral projections of the faces are shown on the right of the pictures, and estimated values below them.

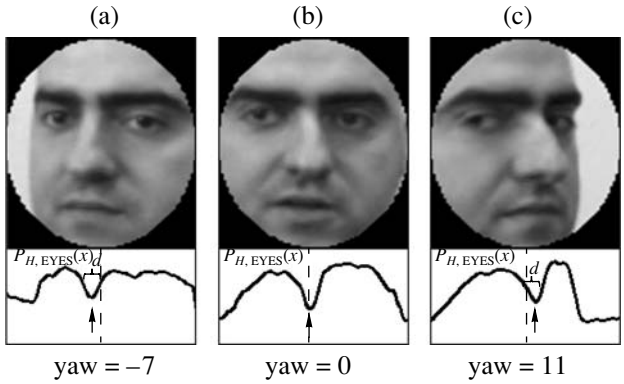

Fig. 7. Yaw computation. Horizontal integral projections of the eyes, $P_{H,\,EYES}$, and estimated yaw values are shown below the pictures.

As we have mentioned before, it is clear that this method, designed specifically for human faces, will not work correctly with other kinds of objects.

4. VIRTUAL ENVIRONMENT MOVEMENT

The perceptual interface described in Sections 2 and 3 has been integrated into a virtual 3D environment, specifically constructed for this purpose. The action in the program is carried out in a first-person view, in a similar way to a 3D video game. In this way, when the player moves his/her head, the system moves accordingly. The virtual environment consists of a number of cells, or rooms, where the user can walk freely. An overview of the world and some of its rooms are shown in Fig. 8.

Concretely, the user can move forward, backward, left, right, up and down, rotate left and right, and look up and down. These movements are controlled by means of the estimated pose parameters in the following ways:

Move forward/backward. Estimated z is used to determine the direction and magnitude of this movement. Recall that this value is relative to the initial depth, so $z \approx 1$ involves no movement. If $z < 1$, we move forward, and if $z > 1$, backward.

Move left/right. The horizontal location of the face, x, with respect to the image center, x_{center}, is used here. To avoid an excessive sensitivity of the system, we define a certain region $(x_{\text{center}} - k, x_{\text{center}} + k)$ where no movement is done. If $x < x_{\text{center}} - k$, we move left, and if $x > x_{\text{center}} + k$, we move right.

Move up/down. The virtual world allows a small variation in the vertical position of the camera that would correspond to walk raised or crawling. Parameter y controls this mode. Again, two thresholds, $y_{\min}$ and $y_{\max}$, are defined. When $y < y_{\min}$, we walk raised, and when $y > y_{\max}$, we crawl. Otherwise, an intermediate mode is used.

Rotate left/right. In this case, two parameters are taken into account: roll and yaw. Both contribute to determine the quantity of this rotation. In addition, roll has more priority than yaw. Thus, *roll* > 0 means rotate left, and *roll* < 0 means rotate right. Otherwise, if *roll* is 0 (or near 0), and *yaw* < 0 or *yaw* > 0, left or right rotation is done, respectively.

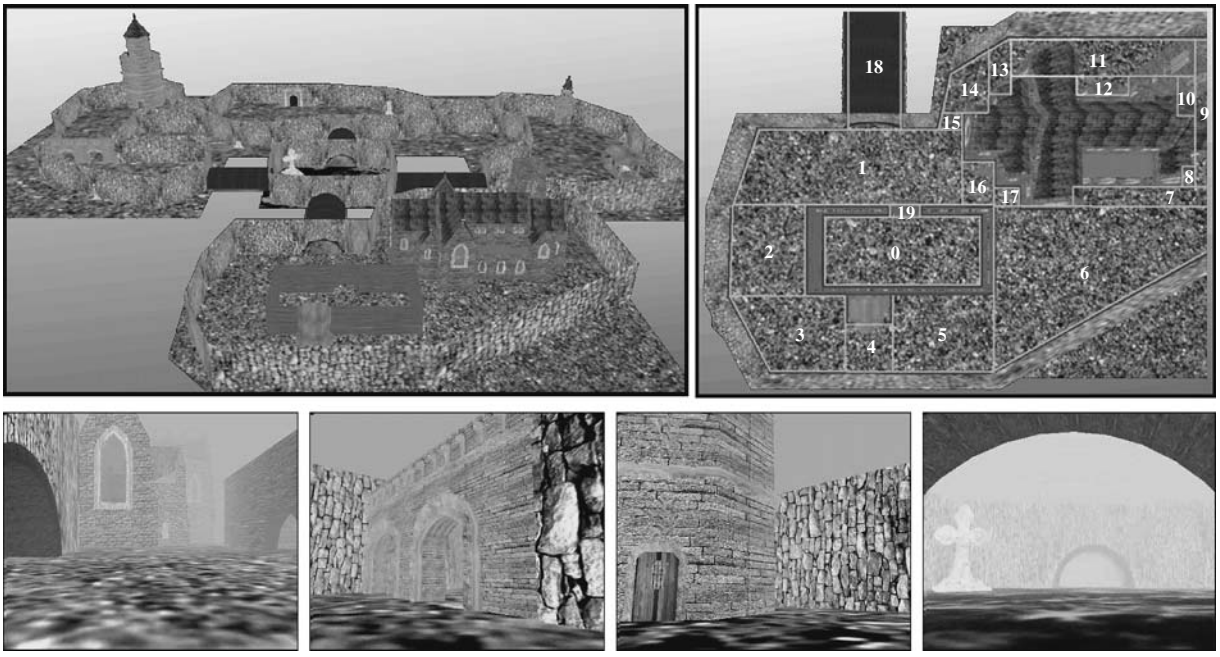

Fig. 8. Overview of the virtual 3D environment. Top: a bird's-eye view of the world (left) and a top view of some cells in the south (right). Bottom: samples of first-person views.

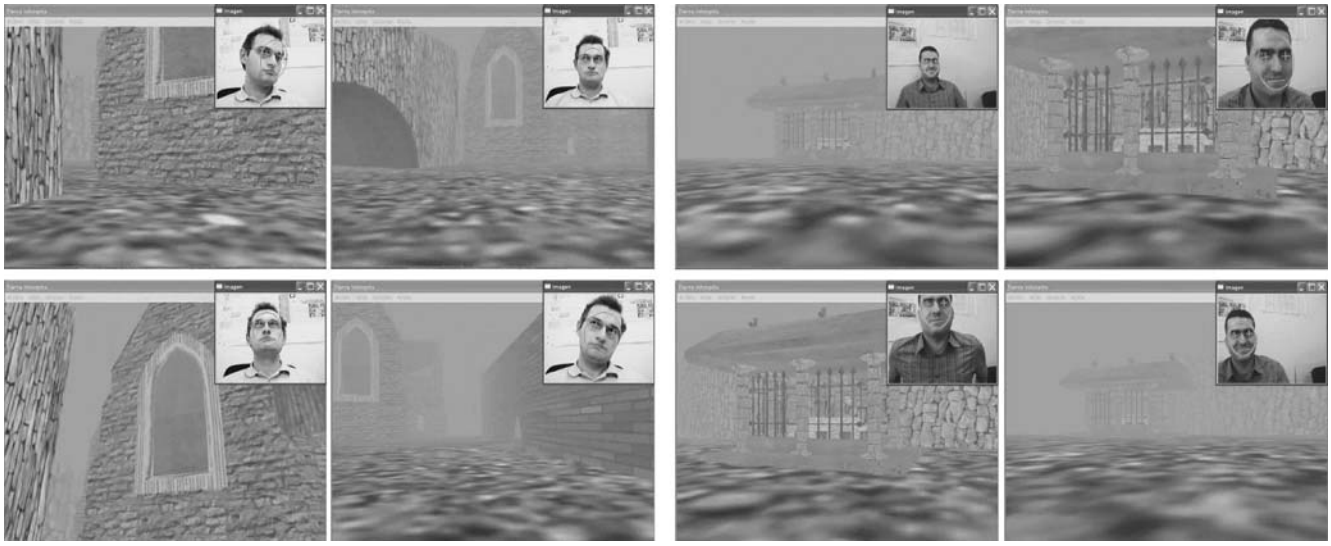

Fig. 9. Some execution samples of the prototype. The input video is shown in a window in the upper right corner.

Look up/down. The estimated pitch is applied in this control signal. *pitch* > 0 involves look up, and *pitch* < 0 involves look down. For a better stability, when *pitch* is 0 (or near 0), the program smoothly tends to a straight look.

To make the system a bit more stable, a small temporal smoothing is introduced in the estimated pose values. This causes a certain delay, but avoids trembling owing to small errors in the estimations. In the experiments, we have used a temporal smoothing of five frames. Working at an average frame rate of 10 fps, this means a reduced delay of 0.5 s.

On the other hand, as we have mentioned before, not all of the estimated pose parameters are independent. For example, a rotation along one direction can result in a bad estimation of some other parameters. Therefore, the movements have been prioritized to avoid this interdependence. In our face interface, the movement with the most priority is pitch. If a relevant pitch value is detected, all the other movements are omitted.

5. EXPERIMENTAL RESULTS

A prototype of the perceptual interface presented in this paper has been implemented using Intel® IPL and OpenCV libraries for image analysis and computer vision [5]. The program runs under MS Windows®, and the rendering engine uses DirectX® 9.0b, thus enabling hardware optimizations. The prototype program, named Tierra Inhospita, is publicly available[2] and can be downloaded free at the following site:

`http://dis.um.es/~ginesgm/th`

This web page contains some video samples and additional information about the perceptual interface and the virtual environment.

The program has been tested mainly on a Pentium IV at 2.6 GHz and 512 Mbytes of RAM, and also in some other systems. For the acquisition we have used two cameras: a QuickCam® Pro through USB and a Sony® DFW500 through FireWire. The program can also take as input a recorded AVI file.[3]

Figure 9 shows some execution samples of the program for two different users.

In the experiments reported here, we have not taken quantitative measures of the accuracy and stability of the system (for a detailed study of the face tracker accuracy, see [4]). However, we can point out some interesting results:

The program performs in real time, with an average of about 30 fps in rendering. Using the USB camera, image processing is done at 10 fps. We have to note that this is only limited by the speed of the acquisition system. With the file recorded from the FireWire camera, images can be processed at 30 fps.

In general, the responsiveness and stability of the system are very good. The face tracker is able to work with typical speeds in the user's movement and under a wide range of rotations. However, profile faces are not recognized owing to the structure of the tracker. When the face is lost, detection is applied until the face is found again.

[2] Regrettably, there is only a Spanish version of the program available. However, that should not be an impediment to using the program, as it is very easy to use (just install and execute). Some instructions and system requirements are indicated in the web page.

[3] In fact, the Sony DFW500 camera is used through recorded AVI files, as far as the current version of OpenCV does not support FireWire cameras.

The method is also very robust to facial expressions, like speaking, blinking, etc. A normal variation in this sense does not affect the estimated pose parameters. On the other hand, the method is sensitive to illumination conditions. For example, a strong lateral source of light could produce incorrect results or even impossibility of detecting and tracking the face.

Some parameters are more reliable than others. In this way, only a small rotation is admitted in *yaw* (recall that profile faces are not tracked). A wider variation is allowed in *pitch*, but estimated pitch values are not very reliable for large angles. All the other parameters, *x*, *y*, *z*, and *roll*, are computed with a fairly good accuracy.

6. CONCLUSIONS

Integral projections have been successfully applied to deal with the problems of human face detection and tracking. They provide a fast, robust, and stable way to track and locate human faces along a video sequence.

The main purpose of the work described in this paper was to show that integral projections can also be used to obtain 3D information of the face position and orientation. Using some simple heuristics, the tracking technique is extended to produce these estimations for the 3D pose of the user's face.

The values resulting from pose estimation are used in a perceptual interface, which is seamlessly integrated in a virtual 3D environment. This is done by translating face pose and movement into control signals in the virtual world. Although the values obtained with our method are not very adequate if a precise 3D tracking is needed (in terms of angles and distances in the real world), they prove to be very useful and satisfactory in applications such as perceptual interfaces. Moreover, our simple method avoids the unnecessary complexity of the generic problem of 3D reconstruction.

ACKNOWLEDGMENTS

This work has been supported in part by the Spanish MCYT under grant DPI-2001-0469-C03-01.

REFERENCES

1. A. Pentland, "Looking at People: Sensing for Ubiquitous and Wearable Computing," IEEE Trans. on PAMI **22**, No. 1, 107–119 (2000).
2. G. R. Bradski, "Computer Vision Face Tracking For Use in a Perceptual User Interface," Intel Technology Journal Q2'98 (1998).
3. K. Sobottka and I. Pitas, "Segmentation and Tracking of Faces in Color Images," in *Proceedings of 2nd Intl. Conf. on Automatic Face and Gesture Recognition* (1996), pp. 236–241.
4. G. Garcia-Mateos, "Refining Face Tracking with Integral Projections," in *Proceedings of AVBPA'2003*, (LNCS 2086, 2003), pp. 222–229.
5. H. A. Rowley, S. Baluja, and T. Kanade, "Neural Network-Based Face Detection," IEEE Trans. on Pattern Analysis and Machine Intelligence **20**, No. 1, 23–38 (1998).
6. M. H. Yang, N. Ahuja, and D. Kriegman, "A Survey on Face Detection Methods," IEEE Trans. on Pattern Analysis and Machine Intelligence (2002).
7. Intel® OpenCV: Open Source Computer Vision Library: http://www.sourceforge.net/projects/opencvlibrary
8. R. Stiefelhagen, J. Yang, and A. Waibel, "A Model-Based Gaze Tracking System," in *Proceedings of IEEE Intl. Symposia on Intelligence and Systems* (1996), pp. 304–310.
9. K. Schwerdt and J. L. Crowley, "Robust Face Tracking Using Color," in *Proceedings of 4th Intl. Conf. on Aut. Face and Gesture Recognition* (Grenoble, France, 2000), pp. 90–95.
10. V. Pahor and S. Carrato, "A Fuzzy Approach to Mouth Corner Detection," in *Proceedings of ICIP-99* (Kobe, Japan 1999), pp. I-667–I-671.
11. R. Lienhart, "Cascade of Boosted Classifiers Working with Haar-like Features," in *Proceedings of 2nd Intl. Conf. on Automatic Face and Gesture Recognition* (2002), pp. 236–241.
12. R. Kaucic and A. Blake, "Accurate, Real-Time, Unadorned Lip Tracking," in *Proceedings of 6th Intl. Conference on Computer Vision* (1998), pp. 370–375.
13. S. Spors and R. Rabenstein, "A Real-Time Face Tracker for Color Video," *Conference on Acoustics, Speech, and Signal Processing* (IEEE Intl. Utah, USA 2001).
14. M. Isard and A. Blake, "Contour Tracking by Stochastic Propagation of Conditional Density," in *Proceedings 4th European Conf. on Computer Vision* (Cambridge, UK 1996), pp. 343—356.
15. C. Vieren, F. Cabestaing, and J. Postaire, "Catching Moving Objects with Snakes for Motion Tracking," Pattern Recognition Letters **16**, 679–685 (1995).
16. A. Pentland, B. Moghaddam, and T. Starner, "View-Based and Modular Eigenspaces for Face Recognition," in *Proceedings CVPR'94* (Seattle, Washington, USA 1994), pp. 84–91.
17. M. La Cascia, S. Sclaroff, and V. Athitsos, "Fast, Reliable Head Tracking Under Varying Illumination: An Approach Based on Registration of Texture-mapped 3D Models," IEEE PAMI **22** (4), 322–336 (2000).

Ginés García-Mateos received his MS degree in computer science from the Universidad de Murcia, Spain, in 1998. He joined the Computer Science and Systems Department, from the Universidad de Murcia, in 1998, where he is currently an assistant professor of algorithms, data structures, and image processing. His research interests include computer vision, pattern recognition, and image analysis, particularly human face processing, perceptual interfaces, and biometrics, where he is currently developing his PhD.

Sergio Fructuoso-Muñoz received his MS degree in computer science from the Universidad de Murcia, Spain, in 2004. He developed his MS project in the Computer Science and Systems Department, in the subject area of perceptual interfaces. His research interests include computer graphics, virtual environments, image analysis, pattern recognition, and entertainment software.