Perceptive User Interface, A Generic Approach*

Michael Van den Bergh[1,3], Ward Servaes[2], Geert Caenen[1], Stefaan De Roeck[1], and Luc Van Gool[1,3]

[1] ESAT-PSI, University of Leuven, Belgium,
{mvandenb,caeneng,sderoeck}@esat.kuleuven.be
[2] PHILIPS, ward.servaes@philips.com
[3] Computer Vision Group (BIWI), ETH Zuerich, Switzerland,
vangool@vision.ee.ethz.ch

Abstract. This paper describes the development of a real-time perceptive user interface. Two cameras are used to detect a user's head, eyes, hand, fingers and gestures. These cues are interpreted to control a user interface on a large screen. The result is a fully functional integrated system that processes roughly 7.5 frames per second on a Pentium IV system. The calibration of this setup is carried out through a few simple and intuitive routines, making the system adaptive and accessible to non-expert users. The minimal hardware requirements are two web-cams and a computer. The paper will describe how the user is observed (head, eye, hand and finger detection, gesture recognition), the 3D geometry involved, and the calibration steps necessary to set up the system.

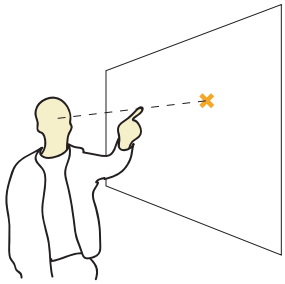


Fig. 1. User pointing at a point on screen.

1 Introduction

With the abundance of multimedia information and the increasing ubiquity of computing power, the number of applications for perceptive user interfaces is growing. Examples are a security control room, browsing a multimedia database as seen in the motion picture *Minority Report*, controlling a home entertainment system, the gaming industry... Vision allows us to detect a number of interesting cues like location, identity, a user asking for attention, expression, emotion, a user pointing at objects or a user gesturing. This paper summarizes the development of a perceptive user interface that allows a user to point at objects and

* This work was supported by GOA/2004/05, Research Fund K.U.Leuven and the ETH project Blue C II.

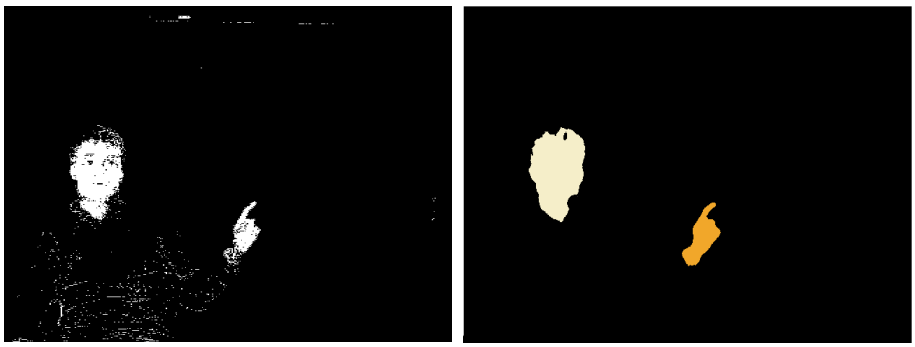
manipulate them with hand gestures.
There are a few existing non-commercial systems with similar objectives. One example is *PFinder*, developed by C. Wren, A. Azarbayejani and A. Pentland at the Massachusetts Institute of Technology (MIT), USA [1]. This system analyzes color and shape to detect the head and the hands of the user. A body contour is extracted by comparing the current image with a stored background image. Subsequently a number of coordinates in the image are selected with high probability of being a head or a hand. The method is very fast, but an interface as proposed in this paper requires a more detailed human model. Other systems attempt to build a more articulated and dynamic body model, e.g. the system developed by R. Plänkers and P. Fua at the École Polytechnique Fédérale de Lausanne EPFL, Switzerland [2]. They built a body model from *soft objects*. The model is iteratively fit to the camera images. This approach yields a high level of detail, but is too complex for a real-time implementation.
In this paper we describe a system that is both fast enough to implement a real-time perceptive user interface, and sufficiently detailed to allow a user to point at objects on a screen and to recognize the user's hand gestures. The rest of this paper is organized as follows. First the user's head and hands are extracted through skin color analysis (section 2.1). Next, the user's fingers and eyes are located and the user's gestures are processed (sections 2.2-2.4). If the user is pointing, an imaginary line is constructed between the user's eyes and his finger tip (figure 1) and intersected with the screen plane (section 3). The system is designed to be robust and adaptive to new environments, which requires a number of calibration steps (section 4). Finally, we describe the demonstration GUI and the integrated setup (section 5). Considering the real-time nature of the system, all algorithms need to be selected with speed of execution in mind.

2 Observing The User

2.1 Skin Color Analysis

This section addresses the detection of skin pixels in a camera image. Skin and non-skin pixels are modeled as distributions in the rg-color space. Based on these distributions a MAP rule is derived for the probability of a color pixel being skin or not. The results are further improved in a post-processing step.
First the image is converted to the rg-color space, which is an intensity normalized color space ($r = R/(R+G+B)$ and $g = G/(R+G+B)$). The benefits of this transformation are a lighting invariant, compact representation of skin colors and a speedy computation. To model skin and non-skin pixels in this color space, a *gaussian mixture model* (GMM) is implemented similar to [3]. The result are two distributions, $p(\mathbf{c}|skin)$ and $p(\mathbf{c}|non\text{-}skin)$. The model for the probability that a given color is skin is constructed with Bayes' rule,

$$P(skin|\mathbf{c}) = \frac{p(\mathbf{c}|skin)}{p(\mathbf{c}|skin) + p(\mathbf{c}|non\text{-}skin)} \tag{1}$$



(a) (b)

Fig. 2. (a) Result of skin color analysis without post-processing. (b) Result of skin color analysis after post-processing.

where the prior $P(skin)$ is discarded for the time being. Figure 2a shows the result of this detection model, after thresholding. To speed up the detection, the results of (1) are computed offline and are stored in a *lookup table* (LUT). To further speed up the detection, it is applied to a decimated image first, and then detailed detection is applied only to a region of interest (ROI).
The resulting detection still suffers from a lot of noise and holes. To improve the quality, we take another look at the prior that was discarded in (1). It is obvious that a pixel has a higher probability of being a skin pixel if it is surrounded by skin pixels. This information can be included in the prior. [4] explains how such a prior can be modeled with median filtering, requiring very few CPU cycles. Next, a connected components analysis is applied to select only sufficiently large objects (hand, face). Through background segmentation [5] objects from the background can be discarded. The result of these post-processing steps is shown in figure (2b). Note the level of detail of the hand segmentation. This quality will be very important for the robustness of the finger detection and gesture recognition in the following subsections.
Our current implementation uses a fast area-based measure to distinguish between head and hand objects (figure 2b). Though functional, future revisions might be improved with a basic face detector and a more detailed arm model.

2.2 Finger Detection

This subsection will describe the detection of fingers on the detected hand, and the localization of the fingertip of the pointing finger. With speed in mind a simple yet very effective technique from [6] was implemented. The first step is to perform erosion on the hand object until all fingers are removed, and to dilate it back to get a good approximation of the palm of the hand. Subtracting this object from the original observed object leaves only the fingers (figure 3). Using connected component analysis we can remove noise and count the number of observed fingers. For a hand with up to three extended fingers the accuracy of

(a) detection (b) erosion (c) dilation (d) subtract

Fig. 3. Finger detection

the method exceeds 95%. This means the method is robust enough to decide whether the user is pointing at an object (one finger) or gesturing (zero or multiple fingers). If exactly one finger is counted, the finger tip will be located as the furthest point from the gravity center of the palm of the hand.

2.3 Gesture Recognition

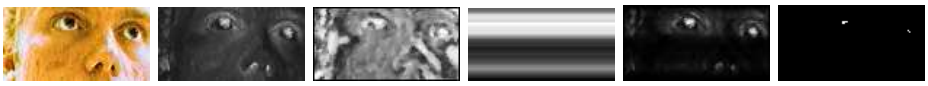
If zero or multiple fingers are detected, gesture recognition is used to understand which action the user is trying to invoke. The starting point is a hand object as segmented in figure 3a. The object is normalized for rotation and size through eigen analysis. Next, the outline shape is extracted with a 3×3 edge filter. The similarity between two edge maps is measured by the Hausdorff-distance [7].
In our case the gesture recognition subsystem can use two camera images. Therefore we can train two edge maps with each gesture, one from each camera. During detection the sum of both Hausdorff-distances is minimized to achieve an effective stereo detection. If there is doubt in one camera image, this ambiguity can be resolved through the second one. The accuracy depends entirely on the chosen gestures. In our experiments we achieved 95% accuracy distinguishing 4 gestures (pointing, scissors, gun and open hand).

2.4 Eye Localization

Localization of the eyes consists of two steps. First the results of the skin color analysis are used to estimate a region of interest (ROI). Using eigen analysis an ellipse is fit to the detected face object. Given this ellipse, a small rectangle can be cut out from the upper part of the face and normalized (figure 4). The second step is to perform a precise detection only on this region. To detect the eyes a number of features are extracted and based on each feature a pseudo-probability is calculated. A combination of these probabilities leads to a more robust detection (figure 4). The first probability is constructed based on luminance, as the eyes (pupils) are distinctly darker spots in the image,

$$p_L(\mathbf{p}) = \exp\left(\frac{-I(\mathbf{p})}{s}\right) \tag{2}$$

where $I(\mathbf{p})$ is the luminance of point p and s is a scaling factor. Experiments under various lighting conditions and using different cameras have shown 100 to



(a) (b) (c) (d) (e) (f)

Fig. 4. (a) region of interest, (b) probability based on luminance (p_L), (c) probability based on color (p_S), (d) probability based on integrated luminance (p_I), (e) product of all probabilities, (f) detected eye components.

be an appropriate value for s. The second probability is based on color, as the eyes are distinctly not skin-colored,

$$p_S(\mathbf{p}) = p(\textit{non-skin}|c_{\mathbf{p}}) \tag{3}$$

Finally a third probability can be extracted from the horizontally integrated luminance. Both the eyes and the shadow region around them present a much area region inside the ROI. These differences can be enhanced through horizontal integration,

$$S_I(\mathbf{p}) = \int_{-w}^{+w} I(t, y)\, dt \tag{4}$$

where p_I is obtained as a normalized version of S_I. To improve the detection an additional probability p_R is introduced to eliminate non-skin colored objects at the sides of the ROI (e.g. hairs, ears). Combining the probabilities yields,

$$p(\mathbf{p}) = p_L(\mathbf{p}) \cdot p_S(\mathbf{p}) \cdot p_I(\mathbf{p}) \cdot p_R(\mathbf{p}) \tag{5}$$

Finally, the eyes are located using connected component analysis, looking for the two components with highest probability. Unlikely detections are discarded after two geometric tests. The distance between the eyes needs to be likely with respect to the size of the face, and the eyes should be horizontal with respect to the rotation of the face. The method's accuracy exceeds 95% during frontal observation. It remains robust with respect to head rotations around the vertical axis until approximately 30° of rotation.

3 3D Extraction

The goal is to figure out which point on the screen the user is pointing at, which should be robust and unambiguous. We chose to draw an imaginary line from the user's eyes, through his fingertip, onto the screen (figure 1). In what follows the 3D coordinates of the eyes and fingertip are written as $\mathbf{X}^o$ and $\mathbf{X}^v$ respectively. The pointing direction can thus be written as $\mathbf{X}^v - \mathbf{X}^o$. The camera matrices P and P' and the screen plane $\mathbf{U}$ are obtained in a calibration step which is described in section 4.

First, we want to translate the couples of 2D coordinates in the camera images

into 3D coordinates in space. The relation between a 2D point $\mathbf{x}$ and corresponding 3D point $\mathbf{X}$ in homogeneous coordinates is [8]:

$$\mathbf{x} = P\mathbf{X} \tag{6}$$

Applying (6) to both cameras (P and P' respectively) we can construct a homogeneous system of four equations,

$$\begin{bmatrix} wP_1 - xP_3 \\ wP_1 - yP_2 \\ w'P'_1 - x'P'_3 \\ w'P'_1 - y'P'_2 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = \mathbf{0} \tag{7}$$

which we can use to calculate the points $\mathbf{X}^o$ and $\mathbf{X}^v$. Finally, the intersection of the line through these two points and the screen plane needs to be calculated. The equation of this line is

$$L \leftrightarrow \mathbf{X} = \mathbf{X}^o + k(\mathbf{X}^v - \mathbf{X}^o) \tag{8}$$

Together with the screen plane (11) we can find

$$\begin{bmatrix} k \\ x \\ y \end{bmatrix} = \begin{bmatrix} \mathbf{X}^v - \mathbf{X}^o & -\mathbf{U}_x & -\mathbf{U}_y \end{bmatrix}^{-1} (\mathbf{U}_0 - \mathbf{X}^o) \tag{9}$$

This equation can be used to compute the 2D pixel coordinate (x, y) on screen directly from the original 2D eye and finger coordinates.

4 Calibration

A number of calibration steps are necessary to get the system running. These steps are designed to be simple and fully automatic, as described in the following subsections.

4.1 Camera Calibration

The *internal camera parameters* can be assumed to be constant for fixed camera settings. The parameters are calibrated once (off line) using a semi-automatic calibration program with calibration object [9, 10]. The internal camera parameters are stored in the calibration matrices K and K' for both cameras respectively. The calibration of the *external camera parameters* (position, viewing direction) should be easy enough to be performed by the end user. To achieve this the calibration only requires the user to point his finger randomly in front of the cameras. The calibration data is recorded by registering correspondences in both camera images as the system tracks the user's fingertip. The relation between a set of correspondences $\mathbf{x}$ en $\mathbf{x}'$ and the camera pair's fundamental matrix F in homogeneous coordinates can be written as

$$\mathbf{x}'^t F\mathbf{x} = 0 \tag{10}$$

F can be determined using several correspondences. A robust method to estimate F is the Ransac method [11]. The camera matrices P and P' can then be derived from F via the essential matrix E and the calibration matrices K and K' as described in [12].

4.2 Screen Calibration

To calibrate the position of the screen, the system will display a sequence of points on the screen one by one, and ask the user to point his finger at them as they appear. Note that this procedure can also be used for the calibration in section 4.1. The 2D coordinates of the displayed points are denoted as (λ_i, μ_i). The 3D positions of the corresponding finger and eye positions are denoted as $\mathbf{X}_i^v$ and $\mathbf{X}_i^o$ respectively. The screen plane α is described by,

$$\alpha \leftrightarrow \mathbf{X} = \mathbf{U}_0 + x(\mathbf{U}_x) + y(\mathbf{U}_y) \tag{11}$$

where $\mathbf{U}_0$ is the upper left corner of the screen and $\mathbf{U}_x$ and $\mathbf{U}_y$ are the horizontal and vertical axes of the screen. The line projected through the 3D finger and eye coordinates is denoted as L_i,

$$L_i \leftrightarrow A_i\mathbf{X} + \mathbf{b}_i = \mathbf{0} \tag{12}$$

The rows of A_i form a basis for the orthogonal complement of the subspaces reproduced by the vector $\mathbf{X}_i^v - \mathbf{X}_i^o$. If an orthonormal basis is chosen, the following equation describes the perpendicular distance of any $\mathbf{X}$ to L_i,

$$d^2(\mathbf{X}, L_i) = (A_i\mathbf{X} + \mathbf{b}_i)^t(A_i\mathbf{X} + \mathbf{b}_i) \tag{13}$$

Every projected point (λ_i, μ_i) in the screen plane corresponds with a line L_i. In order for this point to be equal to the point the user is pointing at, L_i should intersect the screen plane α in that point. Thus, for every i,

$$A_i\left(\mathbf{U}_0 + \lambda_i(\mathbf{U}_x) + \mu_i(\mathbf{U}_y)\right) + \mathbf{b}_i = \mathbf{0} \tag{14}$$

Given a number of correspondences $(\lambda_i, \mu_i) \leftrightarrow L_i$, it is possible to determine α. In order to deal with unavoidable measurement errors, a least squares solution will be computed based on N correspondences. Using (13) we look for the plane α that puts the screen points $\mathbf{u}(\lambda_i, \mu_i) \in \alpha$ as closely as possible to their corresponding L_i.

$$\sum_i (A_i\mathbf{u}(\lambda_i, \mu_i) + \mathbf{b}_i)^t(A_i\mathbf{u}(\lambda_i, \mu_i) + \mathbf{b}_i) \tag{15}$$

which can be rewritten,

$$\sum_i \left(A_i\Lambda_i\mathbf{u} - \mathbf{b}_i\right)^t \left(A_i\Lambda_i\mathbf{u} - \mathbf{b}_i\right) \text{ where } \mathbf{u} = \begin{bmatrix}\mathbf{U}_0\\ \mathbf{U}_x\\ \mathbf{U}_y\end{bmatrix} \text{ and } \Lambda_i = \left[\, I_3 \middle| \lambda_i I_3 \middle| \mu_i I_3 \,\right] \tag{16}$$

Minimization with respect to $\mathbf{u}$ yields,

$$\underbrace{\sum_i (A_i \Lambda_i)^t (A_i \Lambda_i)}_{S} \mathbf{u} + \underbrace{\sum_i (A_i \Lambda_i)^t \mathbf{b}_i}_{\mathbf{g}} = \mathbf{0} \Leftrightarrow S\mathbf{u} = -\mathbf{g} \tag{17}$$

The solution to this equation is not unique. The screen can still be shifted away from the user. This leaves one free parameter for the screen plane. As we only need to determine the 2D screen coordinates based on the 3D finger and eye coordinates (9), this free parameter is irrelevant and can be discarded.

4.3 Color Calibration

The quality of the skin color analysis in section 2.1 depends a great deal on the white balance parameters of the cameras and the chosen threshold for (1). To improve the robustness and usability of the system we have developed an automatic system to adapt these parameters to the given camera and lighting conditions. We present a simple setup where the system imitates a mirror with a small rectangle overlaid in the middle of the picture. The user needs to sit and position his head exactly inside this rectangle before calibration begins. The following measure of skin color analysis quality is proposed,

$$q = n_+ - n_- \tag{18}$$

where n_+ denotes the number correctly classified pixels, the number of skin pixels inside the rectangle and the number of non-skin pixels outside, and n_- the number of skin pixels detected outside of the rectangle. The white balance is varied until a maximum is found for q. Given this white balance, the same quality measure is used to determine the optimal threshold. In most cases the color calibration will yield a natural white balance and a neutral threshold. Only with very strange backgrounds or lighting conditions the calibration will result in a compensating white balance.

5 Integrated System

5.1 Graphical User Interface

Until now we have described methods to detect skin, eyes, hands, fingers and gestures. Based on the coordinates of the eyes and fingertip we can determine which point on the screen the user is pointing at. Now it's important to illustrate the functionality in a clear and useful interface (figure 5). The goal is to build an environment where the user can point at objects (icons) to highlight them, grab them, drag them and activate them. Each icon has three states: normal, highlighted and grabbed. When the icons appear on the screen they are in the normal state. When the user points at an icon it will grow and become highlighted. The user can now gesture to grab the icon. The icon will go to the

Fig. 5. Using the interface.

grabbed state and its highlight will change color. Finally the user can gesture to activate the icon. In our implementation activating an icon opens a hyperlink to a new interface with new icons. All gestures are trained during a training procedure in advance and they can be chosen freely by the user. The only gesture that's fixed is the pointing gesture, where the user only extends one finger. The icons, their location on screen and their hyperlinks are defined in an XML file.

5.2 Integrated Setup

Our test setup consists of a large screen (3 m by 2 m) with two Sony DFW-VL500 cameras placed alongside it. The cameras are positioned near the border of the screen, in the middle of the vertical sides. The cameras are pointed directly at the user. The cameras could in fact be positioned anywhere, as long as the eyes and the hands are visible inside the picture. Both cameras continuously grab images. These images are passed through the skin color analysis. The grabbed images and their binary skin images are fed to the subsystems described in section 2. Depending on the state (pointing or gesturing) the system applies finger tip detection or gesture recognition. The gesture recognition system analyzes the stereo images and returns the ID of the recognized gesture and the hand's center of gravity. The eye detector locates the coordinates of both eyes and returns the middle point. Based on two coordinates (either fingertip and eyes, or hand center and eyes) an imaginary 3D line is constructed. The intersection of this line with the 3D screen plane gives us the 2D coordinate on the screen. The interface processes this 2D screen coordinate along with the ID of the recognized gesture. If the user points at an icon this icon will be highlighted. The gestures are interpreted to perform actions on the highlighted icon.
The result of the integrated setup is illustrated in figure 5. The figure shows a user actively demoing the user interface. Our system is implemented in C++ and achieves approximately 7.5 frames per second on a Pentium IV. The average pixel deviation is 15 pixels on a resolution of 1024x768, which boils down to a pixel deviation of about 2%. The icons in our system are 168x168 pixels in size, comfortably bigger than the pixel error.

6 Conclusion

We have selected several state-of-the-art techniques, keeping the speed of execution in mind. The skin color analysis was improved drastically with a fast post processing step. We implemented a robust stereo gesture recognition system. A fast and efficient eye detection system was implemented. We proposed a set of simple and intuitive calibration routines to calibrate the cameras, the screen and the skin color analysis, making the system adaptive and accessible to non-expert users. Each component was tested for quality and robustness by setting a ground truth and comparing the detection results. As described in the previous sections each component exceeded a 95% detection accuracy in our experiments. Finally we combined all these components to build a working prototype. Several movies demonstrating the work presented in this paper can be downloaded at `http://www.esat.kuleuven.ac.be/~mvandenb`. The functionality of the various subsystems from section 2 are demonstrated in these movies. The final movie shows the working prototype and is a clear proof of concept of our perceptive user interface.

References

1. Wren, C.R., Azerbayejani, A., Darell, T., Pentland, A.P.: Pfinder: Real-time tracking of the human body. IEEE Transactions on Pattern Analysis and Machine Intelligence **19** (1997) 780–785
2. Plänkers, R., Fua, P.: Articulated soft objects for video-based body modeling. Proceedings 8th International Conference on Computer Vision (2001)
3. Jones, M.J., Rehg, J.M.: Statistical color models with application to skin detection. Cambridge Research Laboratory Technical Report Series CRL98/11 (1998)
4. Jedynak, B., Zheng, H., Daoudi, M., Barret, D.: Maximum entropy models for skin detection. Technical Report publication IRMA **57** (2002)
5. Mester, R., Aach, T., Dümbgen, L.: Illumination-invariant change detection using a statistical colinearity criterion. Pattern Recognition: Proceedings 23rd DAGM Symposium (2001) 170–177
6. Hung, Y.P., Yang, Y.S., Chen, Y.S., Hsieh, I.B., Fuh, C.S.: Free-hand pointer by use of an active stereo vision system. Proceedings of Third Asian Conference on Computer Vision **1** (1998) 632–639
7. Sánchez-Nielsen, E., Antón-Canalís, L., Hernández-Tejera, M.: Hand gesture recognition for human-machine interaction. Journal of WSCG **12** (2004)
8. Gool, L.V.: Beeldinterpretatie en Computer Vision II. Visics, KULeuven (2002-2003)
9. Koninckx, T., Griesser, A., Van Gool, L.: Real-time range scanning of deformable surfaces by adaptively coded structured light. Fourth International Conference on 3-D Digital Imaging and Modeling (3DIM03), S. Kawada, ed. (2003) 293–302
10. Koninckx, T., Van Gool, L.: High-speed active 3d acquisition based on a pattern-specific mesh. SPIE's 15th annual symposium on electronic imaging - videometrics VII **5013** (2003) 26–37
11. Fischler, M., Bolles, R.: Random sampling consensus: a paradigm for model fitting with application to image analysis and automated cartography. Commun. Assoc. Comp. Mach. **24** (1981) 381–395
12. Hartley, R.I., Zisserman, A.: Multiple view geometry in computer vision. (2000) 239–240