ELSEVIER

Robotics and Autonomous Systems 19 (1997) 359–368

Robotics and Autonomous Systems

Interacting through eyes

Carlo Colombo [a,*], Alberto Del Bimbo [b]

[a] *Dipartimento di Elettronica per l'Automazione Via Branze 38, 25123 Brescia, Italy*

[b] *Dipartimento di Sistemi e Informatica Via S. Marta 3, 50139 Firenze, Italy*

Abstract

Advanced interaction devices are needed to improve the communication between robots and humans. In this paper we regard computers as specialized robots, provided with a number of sensors for understanding the environment in which they operate (humans included), and *virtual actuators*, which cannot transport materials but can transport information in the form of sensory feedback. Given such a theoretical framework, we focus our attention on the design of a machine vision system for the understanding of gaze pointing actions in advanced interaction applications. Our approach, which is based on the combination of eye tracking in images and a simple drag and click semantics, allows the definition of an *ocular mouse* for interacting with on-screen scenarios. Experimental results obtained with our system are provided, which on the one hand demonstrate the validity and extensibility of our approach to different situations, and on the other point out the fundamental role played by sensory feedback and human adaptation characteristics in man–machine interaction.

1. Introduction

Today's robots interact with humans much more than in the recent past: in some cases, robots and humans even operate and cooperate in the same environment. No way in contradiction with the claim that advanced robots should be autonomous, such human – robot cooperation witnesses the fact that robotic systems of the 1990s have become robust and safe enough to accomplish complex goals, which require decisional autonomy, flexibility and adaptability to unstructured operating scenarios. Service robotics provides many examples of highly cooperative systems such as the robots which distribute meals in hospitals, take lifts and navigate along corridors avoiding patients and physicians, or the robots for home care of disabled and elderly people.

We can attempt to give an informal, recursive definition of advanced autonomous robots as *artificial systems which process and transport materials and information, and interact with the environment, which possibly includes humans and robots* [3]. The definition stresses the importance of maintaining, processing and exchanging information (not only materials, as with industrial pick-and-place manipulators), and implicitly emphasizes the role of perception as a key to communicate with the external environment. Fig. 1 shows a robot classification based on the relative importance of the perception and actuation aspects. Robot (a), which is not equipped with sensors at all, is the classical industrial manipulator for the repetitive execution of predefined tasks. Robot (b) is provided with both sensors to perceive the environment and actuators to modify it, and can execute complex tasks requiring exteroceptive control and adaptation to the operating context. Opposite to robot (a), robot (c) lacks of actuating devices, and is specialized in understanding the surrounding environment, even if without the possibility of altering it.

* Corresponding author. E-mail: columbus@bsing.ing.unibs.it.

PII S0921-8890(96)00062-0

(a) actuators only **(b) sensors and actuators** **(c) sensors and virtual actuators**

Fig. 1. Robot classification based on sensorimotor capabilities.

Computer systems, especially those provided with advanced devices for interaction with the user, can be regarded as a special instance of robots of class (c), as they cannot transfer materials but they do exchange information. Let us suppose that information is in the form of acoustic and visual signals; then the system will use microphones and video cameras for sensory input, and monitors and speakers for output. The latter devices can be regarded as *virtual actuators* which produce, in some sense, a modification of the external environment (for example, opening a new window of a computer screen determines a change of the user's visual environment).

In the last few years, a number of sensing devices – such as head mounted displays and datagloves, touch screen systems provided with heat or pressure sensors, body markers, etc. – have been proposed as advanced tools for extending the basic interaction facilities offered by keyboard and mouse [12]. A special class of advanced interaction tools is the one based on *gaze-input*, and operating according to the principle of What You Look At Is What You Get (WYLAIWYG), that is, system input information is encoded in the user's direction of gaze [8]. Most of the gaze-driven interaction tools are data-helmets with some embedded infrared system, provided with a transmitter and a receiver, which evaluates the current pupil position by exploting the differences in infrared reflectivities between the pupil and surrounding iris and sclera (an example is the ERICA system developed at the University of Virginia for handicapped people [7]). Other systems, used mainly for experiments in psychophysics, use either special light-emitting contact lenses or electrodes placed around the eye, which measure the contraction of the ocular muscles [2]. Advanced interaction devices can be rather expensive, as they are often some mechatronic assembly of miniaturized sensors and actuators, combined with special processing hardware. Another problem connected with many such devices is intrusivity, as the devices can be cumbersome to use.

Given a particular interaction device, the problem arises of how to couple raw sensor input with suitable data processing strategies in order to design effective systems for environment perception. Recently, the simultaneous exponential increase of computing power and decrease in the cost of hardware has encouraged to use computer vision as a non-intrusive technology for the design of advanced man–machine interaction systems [1]. This popularity is mainly due to the fact that a number of visual data processing techniques exist to locate and track user body parts in images without any dedicated hardware. These have already been successfully used for lip-reading, interpreting hand gestures and recognizing facial expressions [5,11].

In this paper, we present a vision-based, gaze-input interaction system to communicate with a computer through pupil shifts and an inexpensive camera. The proposed approach, contactless and non-intrusive, is suitable to provide all users, and specifically the disabled affected by severe motor pathologies, with a friendly modality of dialogue with the machine. The approach can be equally applied to different interaction scenarios, ranging from navigation in hypermedial environments and graphic interfaces to teleoperation and telepresence. Another use of our system is related to simply monitoring user actions, i.e. without turning eye-gaze information into command input: this has applications in market analysis and usability engineering (monitoring the ocular activity and satisfaction of possible purchasers and users of new products).

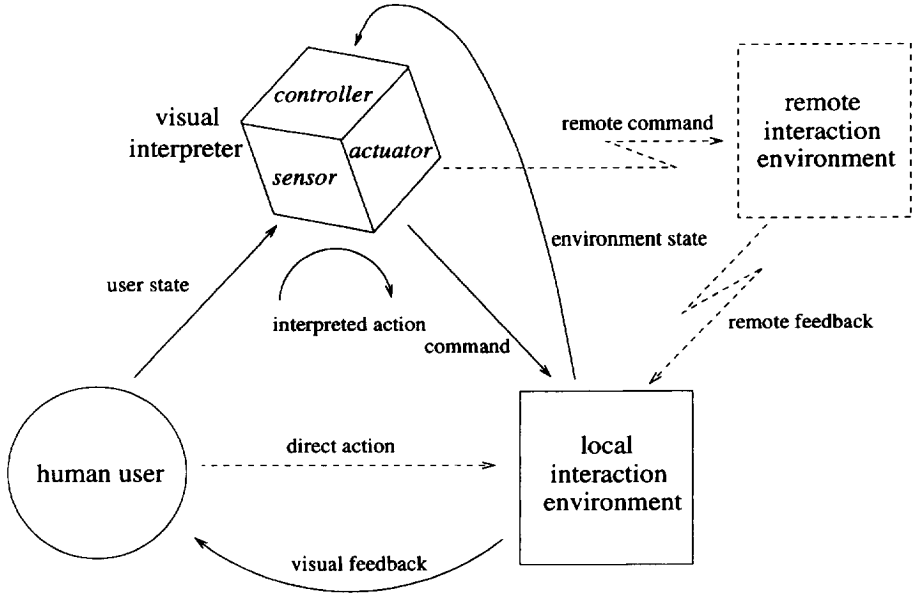


Fig. 2. Geometry of gaze-driven interaction.

In our approach, the user's eyes are continuously tracked by means of computer vision, and image information is used to infer user action. This leads us to define an *ocular mouse* for natural pointing in bidimensional (2D) interfaces which, intercepting the user's shifts of interest, can provide the screen location currently observed by the user. A simple semantics is associated to the ocular mouse, which allows us to replicate the basic functions of dragging and clicking with conventional mice, and accommodate pointing, navigation, and selection commands.

The approach presented here is derived in the context of a conceptual scheme of man–machine interaction mediated by computer vision, which regards interaction from the Cybernetics viewpoint of the communication between two processes – modeled after the man and the machine, respectively – sharing a common visual information channel and communication alphabet.

2. Interaction scheme

We configure the general scheme for vision-based interaction as a loop, whose basic units are the user, an on-screen interaction environment provided by the computer and a *visual interpreter* (Fig. 2). The user interacts with the environment by performing exploratory and/or manipulatory actions in it. The role of the visual interpreter is twofold: to analyze user actions by means of computer vision, and to map and execute them as changes of the environment.

The environment plays an active role in the interaction, in that it not only provides the user with the results of his action (feedback), but also makes explicit the rules that support the communication with the machine. In the general case, the environment of interaction does not need to be fully visual (it could be e.g. the combination of visual and acoustic information), nor fully the output of a computer (think for example to an augmented reality environment, in which real world and synthetic-projected features are blended together to form the environment of interaction [13]). In addition, the environment can locally display the content of a remote scene, thus allowing human-to-human communication, telepresence and teleoperation.

The visual interpreter is composed of three elements which, borrowing from the Cybernetics terminology, we refer to as actuator, sensor, and controller.

The *actuator* is any device capabable of changing the environment. Virtual actuators can modify the position and size of a 2D window, or displace a virtual

camera so as to determine a viewpoint change in a graphic environment. Real actuators can command either locally or remotely a mechatronic system, such as a pan-tilt camera unit.

The *sensor* is the front-end to the user. Its role is to visually track the user, and produce a set of visual parameters which reflect both the geometric and temporal characteristics of user action.

The *controller* can be regarded as a finite state machine which supervises the visual interpreter operations, being responsible of mapping visual parameters from the sensor onto commands for the actuator. Such a mapping is based on a predefined interaction semantics, and depends on the status of the interaction, as resulting from the previous history of both user actions and environment changes. Each new set of input parameters is handled by the controller as an event, sampled from either a continuous or a discrete space. The set of all distinct events handled by the controller makes up the alphabet of the communication.

The overall richness of the interaction alphabet depends on both the sensor and environment. That is, sensor and environment provide the raw information to be exchanged, while the controller provides such information with the proper semantics, based on the context. The more the user is comfortable with the semantics, the more he will forget the fact that his actions are actually interpreted by the interface, and pay attention to what is happening in the environment (user-friendly interface). Such a direct interaction is indicated in Fig. 2 by a dashed arrow. To use a rich semantics at the user and/or at the environment level is a design choice, which is affected, among other specific requirements, by the type and characteristics of the user actions being taken into account by the system.

3. A visual interpreter for gaze pointing actions

The above scheme is general enough to describe interaction for a wide class of user actions and visual features. In this section, we focus our interest on eye-gaze input, and show how to design a visual interpreter for monitoring and remapping gaze pointing actions onto the interaction environment. The key idea is to develop an eye-based pointer to know where the user is actually looking on the screen, and let the eye pupil be used as a 2D mouse.

3.1. Monitoring the user: the sensor

The sensor monitors the aspects of user's activity which are relevant for interaction, and encodes them into a number of parameters (user's state). In Section 3.1.1 we develop a geometric model of interaction, embedding a meaningful characterization of both user behavior and camera acquisition. We use this model in Section 3.1.2 to design the vision algorithms for extracting the user's state from raw visual data.

3.1.1. Models

Gaze pointing actions involve the user's visuomotor subsystem, in terms of fast eyeball displacements, or saccadic shifts, and slow compensation movements of the head [14]. Saccades are basically driven by attentional mechanisms, which are classified either as reflexive (or content-driven, as they depend on some characteristics of the scene such as color, motion, etc.) or *purposive* (or task-driven, e.g. when reading some text or trying to close an on-screen window). When the user is placed in front of a computer screen, it is reasonable to assume that his gaze activity corresponds mainly to saccadic scanpaths [10] accompanied by small oscillations of the head in a plane parallel to the screen. (To optimize interaction, the user naturally tends to keep the face parallel to the screen.)

Geometrically speaking, gaze pointing actions are commonly described in terms of gaze direction, and parametrized by the two degrees of freedom (DOF) expressing gaze direction w.r.t. some fixed reference frame (see e.g. [6]). However, for the purpose of using the eye as a mouse, it is most convenient to choose a different parametrization, and let a gaze pointing action correspond to the coordinates of the screen location pointed by the user, i.e. the intersection of the current gaze direction and the screen plane. It is clear that such parametrization will not only depend on the orientation of the user's visual axis, but also on the head's position.

Some important aspects of the interaction model can be assumed from Fig. 3, which shows the content of the image, screen and face planes as the user is sequentially inspecting the outline of a Z-shaped pattern displayed on the screen:

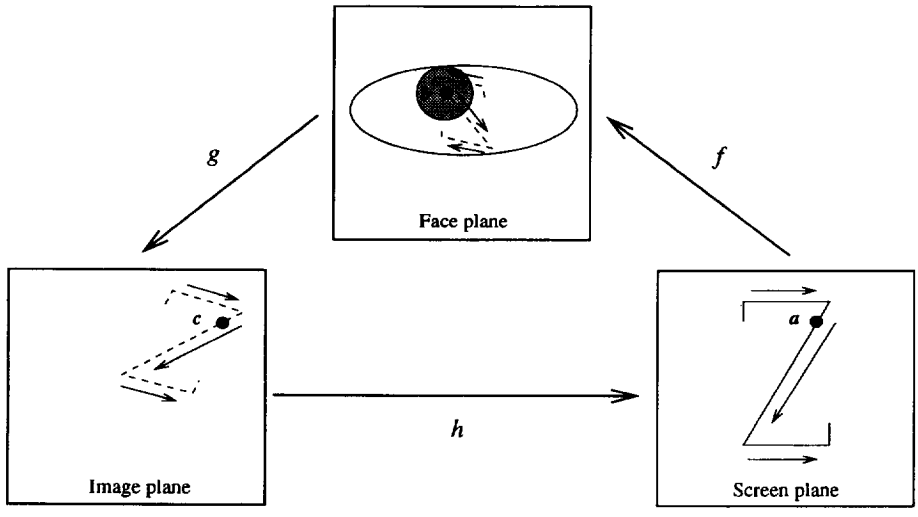


Fig. 3. Geometry of gaze-driven interaction.

- Modeling the user's face as a plane has the effect of regarding eyeball rotations, corresponding one–one to gaze shifts, as translations in this plane;
- From a pure geometric viewpoint, the face plane can be regarded as the image plane of *a camera with only one photo-sensitive element*, which is allocated to the pixel corresponding to the current pupil position (point $\boldsymbol{b}$ in the figure);
- We assume an affine camera approximation [9] for both the screen-to-eye and eye-to-camera projections $\boldsymbol{f}$ s.t. $\boldsymbol{b} = \boldsymbol{f}(\boldsymbol{a})$ and $\boldsymbol{g}$ s.t. $\boldsymbol{c} = \boldsymbol{g}(\boldsymbol{b})$. As a result, the map $\boldsymbol{h}$ which backprojects camera points $\boldsymbol{c}$ onto screen points $\boldsymbol{a}$ is also linear, and defined as a composition of linear maps:

$$\boldsymbol{h} = (\boldsymbol{g} \circ \boldsymbol{f})^{-1} \quad \text{s.t. } \boldsymbol{a} = \boldsymbol{h}(\boldsymbol{c}): \tag{1}$$

- The mappings $\boldsymbol{f}$, $\boldsymbol{g}$ and $\boldsymbol{h}$ produce an affine-deformed version of the original pattern, which can undergo scaling, mirroring, rotation or translation. The mappings are time-dependent, as long as the head is allowed to move freely, as they must depend on head position. Assuming that the external eye contour (shown in the figure as an ellipse) is fixed to the head, we can in principle monitor head movements and update the mappings by observing the time evolution of this contour in the image;
- While the iris can be partially occluded by the eyelids, the pupil is never occluded, and must always lie inside the external eye contour.

3.1.2. Measurements

Image measurements for head tracking and pupil detection are obtained by coupling an elastic template with the raw image data provided by the camera. The template models the external eye contour with elliptic arcs and the iris with an ellipse, whose center locates the pupil. At startup, the template is automatically initialized in the image by brightness gradient histogramming, and the approximate location of the eyes is found. At run-time, a visual tracker is started (Fig. 4, left), which after a few iterations attains and maintains lock onto the current eye visual appearance (Fig. 4, right). In the figure, the segments perpendicular to the external eye contour represent the search paths used by the tracker's predictor to find the next contour instance. The search range adaptively varies in size on the basis of a measure of the current reliability of edge point estimation. A two-step eye tracking approach is used: the next instance of the external eye contour is first found, thus providing the bound for the subsequent iris/pupil search. Besides the search bound, the current external eye location provides the time-varying information needed to measure and update the affine mapping $\boldsymbol{h}$. Head displacement information can also be derived in a simple and robust way by using a template modeling of both external eye contours [4].

To complete the measurement process, image pupil positions must be converted into the corresponding screen locations using an approximation of the affine

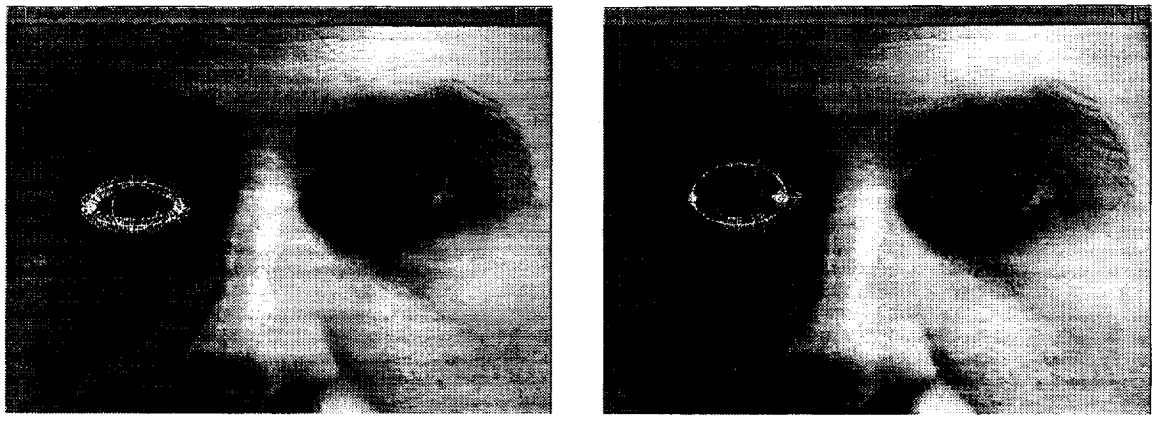


Fig. 4. The visual tracker. Left: Automatic eye localization and template initialization. Right: After locking onto the eye appearance.

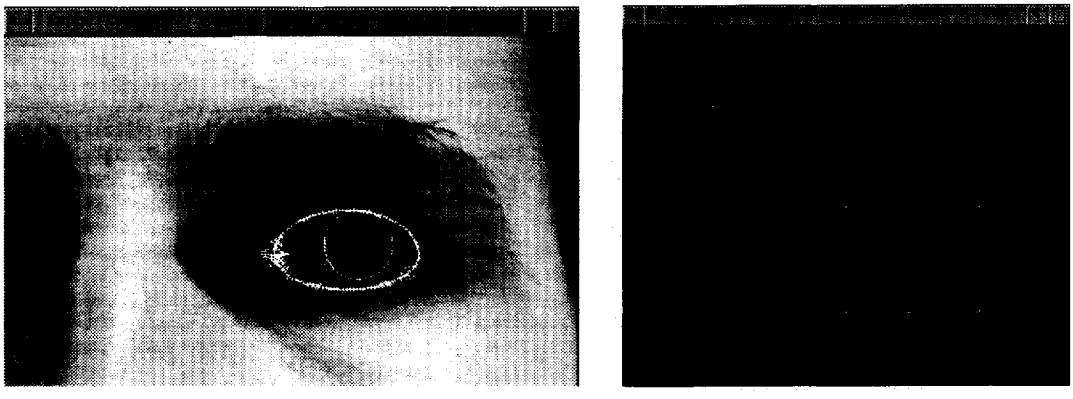


Fig. 5. Eye tracking and calibration. Left: Image plane. Right: The calibration grid, and the point being currently observed (indicated by a small square).

backprojection map $\boldsymbol{h}$ and Eq. (1). This rewrites as

$$\boldsymbol{a} = \boldsymbol{h}(\boldsymbol{g}(\boldsymbol{0})) + H\,[\boldsymbol{c} - \boldsymbol{g}(\boldsymbol{0})], \tag{2}$$

where $\boldsymbol{h}(\boldsymbol{g}(\boldsymbol{0})) = \boldsymbol{f}^{-1}(\boldsymbol{0}) = -F^{-1}\boldsymbol{f}(\boldsymbol{0})$ and $H = [GF]^{-1}$, with $\boldsymbol{f}(\boldsymbol{a}) = \boldsymbol{f}(\boldsymbol{0}) + F\,\boldsymbol{a}$ and $\boldsymbol{g}(\boldsymbol{b}) = \boldsymbol{g}(\boldsymbol{0}) + G\boldsymbol{b}$, F and G are 2×2 matrices, and each point is expressed in the native coordinates of the plane which it belongs to.

Before being used, the affine image-to-screen mapping $\boldsymbol{h}$ is initially estimated at system startup by means of a two-step 2D–2D calibration procedure. First, a set of $M \geq 6$ image observations of pupil positions is collected and recorded (to reduce errors during this phase, each new observation is obtained as an average of subsequent image measurements), obtained by tracking the pupil while letting the user execute a sequence of gaze fixations on a number of given screen locations. Fig. 5 (left) shows the case with $M = 9$, where the locations to fix upon are organized in a 3×3 calibration grid covering the entire screen. Then the affine model is estimated, using data and observations, as the least squares solution of an overdetermined system obtained from Eq. (2).

Sources of calibration errors are inaccuracies in the phase of pupil localization and modeling approximations (e.g. when the mapping is not actually affine). Calibration can be improved by suitably choosing the distribution and the number of grid points – say, 5×5 instead of 3×3. However, the major source of error for the above calibration procedure is the presence of head displacements, as these are not contemplated

in the affine map construction, and can produce large errors on the screen, due to the fact that image and screen pixel coordinates usually differ by a scaling factor much smaller than unity. We can easily extend the calibration procedure by explicitly allowing head displacements. Let us assume for simplicity that the two head translations parallel to the screen are *dominant* w.r.t. the other two DOF which characterize frontoparallel interaction (translation perpendicular to the screen, rotation in a plane parallel to the screen plane). Let us assume also that the image and screen planes are approximately parallel, and that the amount of head translation is small enough to allow the user not to rotate the eyeball in order to fix a same point on the screen. During calibration then, head translations can be compensated for by normalizing the image plane observations w.r.t. a reference $\boldsymbol{g}(\mathbf{0})$, say $\boldsymbol{g}_1(\mathbf{0})$, corresponding to the first observation. Vectors $\boldsymbol{g}_i(\mathbf{0})$ can be easily estimated in the image as the centroids of the external eye contours at discrete observation times $i = 1, \ldots, M$. Compensation is accomplished by properly shifting pupil observations $\boldsymbol{c}_i$ by the amount of image translation of the head:

$$\boldsymbol{c}_i^c = \boldsymbol{c}_i - (\boldsymbol{g}_i(\mathbf{0}) - \boldsymbol{g}_1(\mathbf{0})), \tag{3}$$

where the apex denotes compensation.

Similarly, the calibrated map can be updated at runtime by head motion compensation based on current visual data. To this aim, observe that Eq. (2) can be written at a generic time instant as

$$\boldsymbol{a}_t = \boldsymbol{h}_t(\boldsymbol{g}_t(\mathbf{0})) + H[\boldsymbol{c}_t - \boldsymbol{g}_t(\mathbf{0})], \tag{4}$$

where H is a constant independent of head translation, and the translation term at time t can be expressed as a function of current (t) and reference ($t = 1$) centroid positions and of the translation $h_1(g_1(0))$ obtained at calibration:

$$\boldsymbol{h}_t(\boldsymbol{g}_t(\mathbf{0})) = h_1(g_1(0)) + k_t^{-1}[g_t(0) - g_1(0)]. \tag{5}$$

Head compensation at remapping time is accomplished by properly updating the translation in Eq. (4) according to Eq. (5). The scaling factor k_t in Eq. (5) equals the ratio of camera focal length and eye centroid depth. It can be measured as the ratio of the external eye width in the image (depending on t as the face moves towards or away from the screen) and in the face plane (a constant).

3.2. Interpreting gaze shifts: controller and actuator

Implementation of dialogue semantics is based on a simple *drag and click metaphor* [4], which lets the user interact with the environment using gaze-shifts to perform navigational, pointing and selection actions as with a conventional mouse. Explicitly, gaze displacements are geometrically interpreted as pointer drags. The pointer can also trigger a discrete-time event (click) as gaze persists in the neighborhood of a screen point for a convenient time interval. Notice that, since the production of discrete events is reduced here to the on–off mechanism of time thresholding for clicking, our visual pointer is basically equivalent to a 1-button mouse.

An issue of relevant practical importance is the time responsiveness of the interface; this is directly related to the time threshold used to measure persistence and assign the proper semantics to user action. Choosing the right threshold value is of key importance to have a good balance between speed of operation and naturality of interaction. If the threshold value is too low, then every action is interpreted as a click command, an undesirable situation usually referred to as *Midas touch* [8]. A good dwell time for our 2D pointer, which takes into account the high mobility of the pupil, is 1 s, which on the one hand guarantees a fast response, and on the other limits the occurrence of false alarms.

A different implementation of the system can be obtained for the simple monitoring of user activity in front of the screen. That is, the visual interface is "transparent" to the user, and eye-gaze information is neither traduced into command input (the controller and actuator parts of the visual interpreter are missing), nor a visual feedback of any sort is provided to the user. Monitoring ocular activity can have important applications, e.g. as an aid to the diagnosis of eye pathologies and the performance assessment of advanced interfaces in terms of usability and user satisfaction.

4. Experiments

The system has been developed in C, and uses the X Window libraries for the graphic interface. Experiments have been carried out on a Silicon Graphics Indy platform, featuring a MIPS R4000/R4010 processor with a 100 MHz clock. The video acquisition

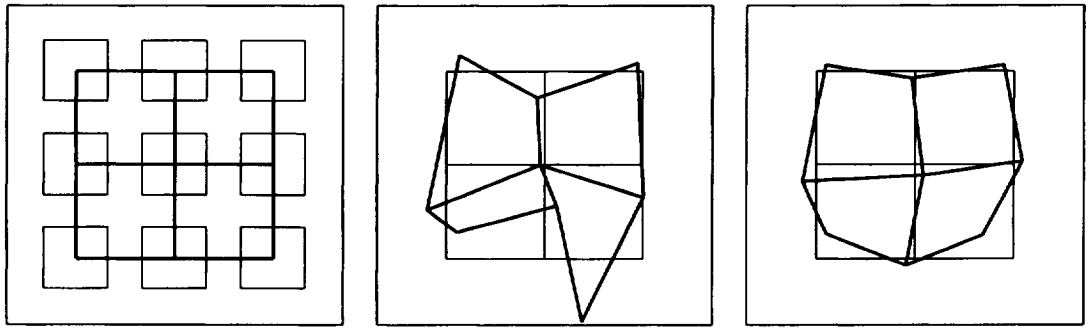


Fig. 6. Calibration results. Left: The nine windows, and the grid connecting their centers. Middle: Results for uncompensated head motion. Right: Results for compensated head motion.

software also runs on the Indy, and gets raw image data through a VINO frame grabber board from an inexpensive B/W camera provided with zoom facilities. The graphic interface is composed of three windows, one of which is used for real-time image display, another for on-screen graphics, and a third for off-line control of interaction parameters.

An interaction session usually proceeds as follows. At startup, the affine template is automatically initialized on a close view of the user's face, and the tracker keeps itself locked onto the external eye appearance. After a short time, a 9-point calibration sequence is started, and the image-to-screen mapping is reconstructed. At this point, an icon indicating the current screen point observed by the user is displayed on the "screen" window. It has been proved experimentally with a number of interviews that users prefer a pointer with some inertia during the motion; this has been taken into account by introducing a proper low-pass filtering of remapped points before display. As far as similar simple graphic interfaces are used, the interaction loop delay is well short enough, visual tracking runs at video rate (40 ms delay), thus providing the user with feedback about his latest action, and a natural interaction with the system is guaranteed.

Several experiments have been conducted in order to test the friendliness of our ocular pointer, two of which are reviewed below.

4.1. Eye reading

The first experiment is devoted to understanding the precision limits of eye-to-screen remapping with and without head compensation, and to assess system behavior and user satisfaction when the ocular mouse is used for "clicking" in a window. This modality can prove useful, for example, to select hot points of hypertext information by eye-gaze input [4].

In the experiment, the user is asked to point in sequence to the centers of nine screen windows. The windows are identical, equally spaced, and arranged in a grid similar to the calibration grid. During such a visual inspection, the user can control the result of his action by monitoring the feedback icon.

Remapping results are shown pictorially in Fig. 6, where neighboring screen points are joined through straight lines (performance grid). The closer the performance grids of Figs. 6(middle, right) resemble the theoretical performance grid of Fig. 6(left), the better is the remapping accuracy. The figures clearly show that, while the performance grid for the uncompensated case appears to deviate significantly from the theoretical grid, in the compensated case the grid closely approximates the correct square shape.

Quantitative test results confirm the importance of doing head motion compensation. The average remapping error, i.e. the average of the magnitude mismatch between the set of theoretical grid ("ground truth") and performance grid points is about 2 cm in the compensated case and more than double (4.4 cm) for uncompensated interaction. Magnitude errors provide us with a resolution limit for the eye pointer, which must be taken into account in the design of a graphic interface, e.g. to properly size up the distance between "clickable" portions of the screen. (This is an example of how advanced interface design is closely coupled

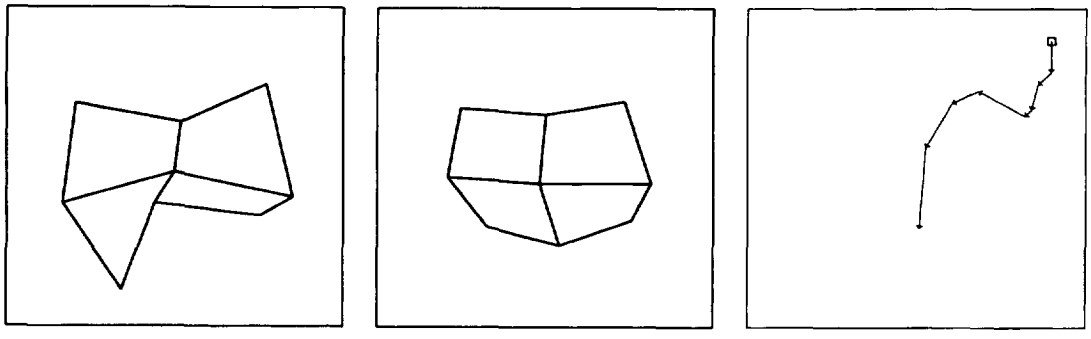


Fig. 7. Image plane measurements during pupil remapping (not in scale). Left: Uncompensated pupil positions. Middle: Compensated pupil positions. Right: External eye image displacements.

with the characteristics of both interaction tools and users.)

To the remapping error contribute calibration errors, inaccuracies in the phase of pupil localization, a non-perfect compensation of head movements and modeling approximations (the mapping is not perfectly affine, actually). Fig. 7 shows, scaled up by a constant factor, the image plane measurements for the remapping experiment: uncompensated and compensated pupil positions, and estimated head displacements. The similarity, apart from a specular coordinate reflection, between the observed point grids and the remapped points grids (performance grids) validates the affine projection model assumption.

4.2. *Eye writing*

The second experiment aims at providing a deeper insight into the role of visual feedback in a man-machine interaction context. The point is: to what extent do we really need to precisely calibrate our system? and: to what extent may we instead rely on the adaptation characteristics of users?

The experiment, performed *without doing calibration*, is conducted as follows. We use as H (the matrix component of the image-to-screen mapping) a predefined matrix of the simple form:

$$H = kI, \tag{6}$$

where I is the 2×2 identity matrix and k is a constant, basically related to pointer inertia. The image translation term of the mapping, $\boldsymbol{h}(\boldsymbol{g}(\mathbf{0}))$, is easily derived, instead, directly from image data, assuming that

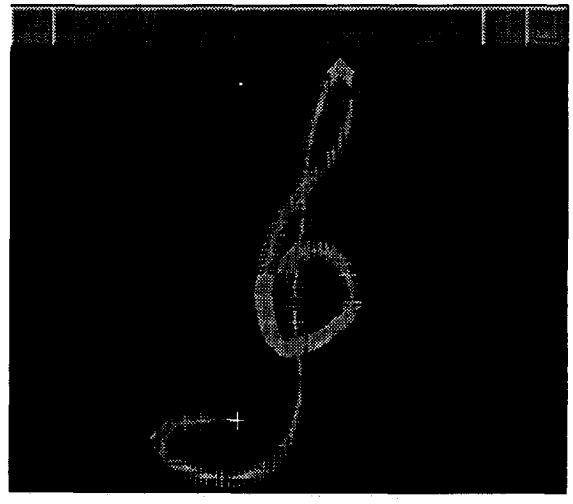


Fig. 8. An example of eye writing: drawing the G-key starting from the center of the screen.

at startup the user is fixating the center of the screen. At startup, the user simply "wears" the ocular pointer much in the way he finds the screen icon representing the current mouse position in a predefined location of the screen, when running a commercial software application. From then on, he is able to displace the icon on the screen at his will according to the given (plausible, yet arbitrary) mapping law, and control, by direct visual inspection, the results of his action.

The pattern of Fig. 8, which represents a musical key, was sketched easily by a non-trained user. (B.t.w., the user was a musician.) In other tests, we experimented the combined use of eye drags and eye clicks for producing "eye-written" text on the computer display with word separation. These promising

results have encouraged us to develop other applications, such as text and graphics editing systems, based on eye pointing and advanced pattern recognition facilities like OCR.

It is remarkable how feedback can be powerful in experiments like this: the user tends to simply forget about machine mediation and semantics, experiments direct interaction, and is mentally projected into the environment. This is a psychological phenomenon which should be well known to trained car drivers, who are seldom conscious of using steering wheel, gear, and brakes. Yet humans, although by far smarter and more adaptable than machines, are hardly ever *exploited by machines* to improve the interaction and dialogue characteristics of a system. Our claim is that such a reversal of perspective, i.e. regarding machines as the main subjects of the interaction, and humans as a (relevant) part of their environment can teach us a lot about human–machine interaction.

Acknowledgements

The authors wish to thank Dr. Giorgio Buttazzo of Scuola Superiore Sant'Anna, Pisa, Italy, for the gentle humour of his hand-drawn robots, appearing in Fig. 1.

References

[1] A. Azarbayejani, T. Starner, B. Horowitz and A. Pentland, Visually controlled graphics, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15 (6) (1993) 602–605.

[2] D. Cleveland and N. Cleveland, Eyegaze eyetracking system, in: *IMAGINA 92*, Monte Carlo (January 1992) 17–23.

[3] C. Colombo, *Artificial Dynamic Vision and Robot-Environment Interaction*, Ph.D. thesis, Scuola Superiore Sant'Anna, Pisa, Italy, February 1996 (in italian).

[4] C. Colombo, A. Del Bimbo and S. de Magistris, Interfacing through visual pointers, in: R. Cipolla and A.P. Pentland, eds., *Computer Vision in Human–Machine Interfaces* (Cambridge University Press, Cambridge, MA, 1996) (to appear).

[5] I.A. Essa and A. Pentland, A vision system for observing and extracting facial action parameters, Technical Report 247, MIT Media Laboratory Perceptual Computing Section (June 1994).

[6] A.H. Gee and R. Cipolla, Determining the gaze of faces in images, Technical Report CUED/F-INFENG/TR 174, University of Cambridge, Department of Engineering, Trumpington Street, Cambridge CB2 1PZ, UK (1994).

[7] T.E. Hutchinson, K. Preston White, Jr., W.N. Martin, K.C. Reichert and L.A. Frey, Human–computer interaction using eye-gaze input, *IEEE Transactions on Systems, Man and Cybernetics*, 19 (6) (1989) 1527–1534.

[8] R.J.K. Jacob, What you look at is what get, *IEEE Computer* 26 (7) (1993) 65–66.

[9] J.L. Mundy and A. Zisserman, Projective geometry for machine vision, in: J.L. Mundy and A. Zisserman, eds., *Geometric Invariance in Computer Vision* (MIT Press, Cambridge, MA, 1992).

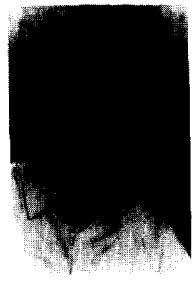
[10] D. Noton and L. Stark, Eye movements and visual perception, *Scientific American* 224 (6) (1971) 34–43.

[11] J.M. Rehg and T. Kanade, 'Digiteyes': Vision-based human hand tracking, Technical Report CMU-CS-93-220, Carnegie Mellon University (December 1993).

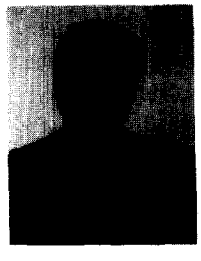
[12] H. Rheingold, *Virtual Reality* (Secker and Warburg, 1991).

[13] P. Wellner, Interacting with paper on the DigitalDesk, *Communication of the ACM* 36 (7) (1993) 86–96.

[14] A.L. Yarbus, *Eye Movements and Vision* (Plenum Press, New York, NY, 1967).



Carlo Colombo was born in Bari, Italy, in 1966. He holds a B.S. degree in electronic engineering from the University of Florence, Italy, and a Ph.D. in robotics from the Scuola Superiore di Studi Universitari e di Perfezionamento Sant'Anna, Pisa, Italy. In 1994 he was a visiting scientist at the Polytechnic of Grenoble, France. He was the local organizer of the 3rd International Symposium on Intelligent Robotic Systems SIRS'95, Pisa, July 1995. He is at present an Assistant Professor at the Department of Electronics for Automation, Brescia, Italy. Dr. Colombo is a member of IEEE and IAPR, and the Secretary of the IAPR Italian Chapter. His main research activities are in the field of artificial vision, with a special interest for attention in machines, autonomous robots and man–machine interfaces.



Alberto Del Bimbo was born in Firenze, Italy, in 1952. He received the doctoral degree in electronic engineering from the Università di Firenze, Italy, in 1977. He was with IBM ITALIA from 1978 to 1988. He is Full Professor of Computer Systems at the Università di Firenze, Italy. Prof. Del Bimbo is a member of the Institute of the Electrical and Electronic Engineers (IEEE) and of the International Association for Pattern Recognition (IAPR). He is in the board of the IAPR Technical Committee no. 8 (Industrial Applications) and 12 (Multimedia) and the President of the IAPR Italian Chapter. He presently serves as Associate Editor of the *Pattern Recognition* Journal and of the *Journal of Visual Languages and Computing*. His research interests and activities are in the field of image analysis, image databases, visual languages and human–computer interfaces.