8

Interfacing through Visual Pointers

C. Colombo, A. Del Bimbo and S. De Magistris

Abstract

A key issue in advanced interface design is the development of friendly tools for natural interaction between user and machine. In this chapter, we propose an approach to non intrusive human-computer interfacing in which the user's head and pupils are monitored by computer vision for interaction control within on-screen environments. Two different *visual pointers* are defined, allowing simultaneous and decoupled navigation and selection in 3D and 2D graphic scenarios. The pointers intercept user actions, whose geometry is then remapped onto the environment by a *drag and click metaphor* providing dialogue with a natural semantics.

8.1 Introduction

In the last few years, a huge effort has been made towards building advanced environments for human-machine interaction and human-human communication mediated by computers. Such environments can improve both the activity and satisfaction of individual users and computer supported cooperative work. Apart from some obvious implementation and design differences, virtual reality [262], augmented reality [318] and smart room [246] environments share the very same principle of providing users with a more natural dialogue with (and through) the computer with respect to the past. This is obtained through a careful interface design involving interface languages mimicking everyday experience and advanced interaction techniques.

Recently, the simultaneous growth of computing power and decrease of hardware costs, together with the development of specific algorithms and techniques, has encouraged the use of computer vision as a non

intrusive technology for advanced human-machine interaction. Experiments using computer vision for gesture interpretation tasks such as lip-reading, recognition of facial expressions, and head motion understanding [175, 119, 14] have been carried out. Vision-based interfaces allow user gestures and actions to be intercepted and used to support interaction in a wide semantic range. For example, in [286] a vision-based hand gesture interpretation technique is used to develop an interface based on sign language, where each sign is related to a specific hand configuration pattern which must be known a priori to the user. Such a rich interaction semantics at the user level is not always desirable, as it may limit the naturality of the dialogue.

In this chapter, we propose a vision-based technique to communicate with a computer through pupil shifts and head motion. A friendly interface is obtained by stressing the continuous-geometric aspects of interaction and supporting natural languages based on visual inspection and selection. Interaction via head and eyes can be effective for both disabled users affected by severe limb motor pathologies and general users requiring non-intrusive interaction tools. The technique, which is presented in the context of a virtual reality environment provided with hypertextual access, can be easily adapted to other tasks and scenarios, and support interaction in multimedia, videoconferencing, telepresence, usability monitoring, and augmented reality environments.

In our approach, a set of image features encoding head and eye position is continuously tracked via computer vision. The features are then remapped onto the interaction environment, as the result of a geometric and semantic interpretation of user action. The location currently observed and the user's interest are inferred by monitoring head and pupil displacement and geometric persistency. This allows us to replicate the two basic functions of dragging and clicking in 2D and 3D with a pair of *visual pointers*, according to a paradigm which allows pointing and navigation (geometric), and selection (semantic) actions.

8.2 Visual pointers

As shown in Figure 8.1, our operating context is composed by a camera and an on-screen multi-window environment. User actions are the result of a change of interest, due either to the low-level visual content of the environment (color, texture, etc.) or to some purposive, high-level user task (context-switching while browsing a hypertext, ameliorating interaction through a viewpoint change, exploring the scene, etc.).

Fig. 8.1 The basic elements of vision-based interaction: environment, user, and camera.

8.2.1 Geometry

To intercept shifts of interest, which involve the user's visuo-motor system, a vision-based interaction tool is designed to track and process simultaneously the displacements of both the head — rigid body in space, six degrees of freedom (DOF) — and the eyeball — two DOF, orientation of the visual axis in space† — subsystems with respect to some fixed reference frame. Such a tool is equivalent to the combination of a 3D pointing device and a 2D pointer, respectively. As the two subsystems are kinematically independent, the associated pointers can also be made independent one from the other by decoupling the measurement of head and eyeball displacements through computer vision.

From the geometric viewpoint, the basic elements of the interaction (camera's image plane, screen and user's face) can be modeled as planar surfaces in space. We also model the visible portions of the user's eyeballs, actually pseudo-spheres, as planar surfaces, and *regard any gaze shift related to an eyeball rotation as a translation of the pupil in the face plane.* Let A, B and Γ denote the screen, user and camera planes respectively, and fix the local frames α, β and γ as in Figure 8.2.

A first geometric aspect of the interaction concerns the head's relative geometry (position and orientation) w.r.t. a given reference frame ρ. This is encoded in the linear transformation between the coordinate

† As for the eyeball DOF, we will actually refer to the intersection of the visual axis giving the direction of gaze, with the screen plane providing the location actually observed.

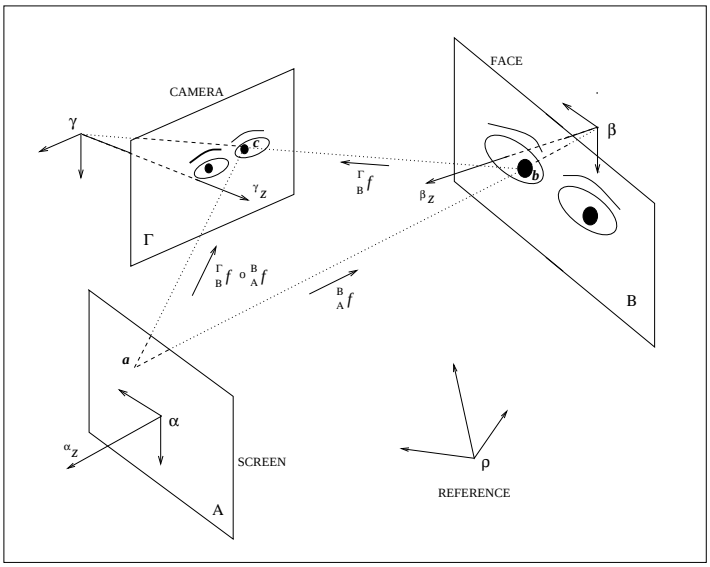

Fig. 8.2. Visual interaction: geometric aspects.

representations of a generic point in space $\boldsymbol{p}$ in the β- and ρ-frames. Using homogeneous coordinates for $\boldsymbol{p}$, such a transformation can be compactly expressed by a 4×4 matrix ${}^{\beta}_{\rho}\mathrm{T}$ s.t. $[{}^{\beta}\boldsymbol{p}^{\mathrm{T}}\ 1]^{\mathrm{T}} = {}^{\beta}_{\rho}\mathrm{T}\ [{}^{\rho}\boldsymbol{p}^{\mathrm{T}}\ 1]^{\mathrm{T}}$. As changes of frame can be composed linearly, the $\rho \mapsto \beta$ transformation can be reconstructed from the $\rho \mapsto \gamma$ and $\gamma \mapsto \beta$ transformations:

$$ {}^{\beta}_{\rho}\mathrm{T} = {}^{\beta}_{\gamma}\mathrm{T}\,{}^{\gamma}_{\rho}\mathrm{T} \quad , \tag{8.1} $$

thus letting the camera come into play as an element of the interaction. The ${}^{\gamma}_{\rho}\mathrm{T}$ transformation is time-independent, as long as the camera is fixed, while the ${}^{\beta}_{\gamma}\mathrm{T}$ transformation does depend on the current position of the user's head.

A second aspect of the interaction involves user's eyeball movements and screen inspection. As the result of an eyeball shift, a specific location of the screen is pointed, and its visible content is projected through the pupil onto the highest-resolution area of the retina, the fovea. Thus, geometrically speaking, we can *regard at each instant the pupil's position in the face plane,* $\boldsymbol{b}$*, as the perspective image of the observed screen point* $\boldsymbol{a}$, i.e. $\boldsymbol{b} = {}^{B}_{A}f(\boldsymbol{a})$, with ${}^{B}_{A}f : A \rightarrow B$ a nonlinear projection map depending, among other parameters, on the entries of ${}^{\beta}_{\alpha}\mathrm{T}$. The relative distance between the user and the camera is such that, in our context,

conditions are met to approximate the perspective camera with an affine camera model [232]. This allows us to say that a linear backprojection map ${}^{B}_{A}h^{-1} = {}^{A}_{B}h : B \rightarrow A$ (with ${}^{B}_{A}h \approx {}^{B}_{A}f$) exists which brings pupil positions to screen locations. The same reasoning can be applied to define the camera-to-face backprojection ${}^{B}_{\Gamma}h : \Gamma \rightarrow B$, and write the map ${}^{A}_{\Gamma}g$ between camera-projected pupil positions $\boldsymbol{c}$ and screen locations $\boldsymbol{a}$ as a composition of linear maps:

$$ {}^{A}_{\Gamma}g = {}^{A}_{B}h \circ {}^{B}_{\Gamma}h : \Gamma \rightarrow A \quad . \tag{8.2} $$

Such a camera-to-screen backprojection map is a linear approximation to a perspectivity (the "picture of a picture" [232]) and, as the product of two maps which involve the user frame β, it is time-dependent.

The transformations of eqs. (8.1) and (8.2) embed the information used by our 3D and 2D pointers, respectively. In Section 8.3 we show how to compute and update such information based on image data.

8.2.2 Semantics

The dialogue semantics currently implemented in our visual interface is based on a *drag and click metaphor*, which lets the user interact with the environment with his head and eyeballs to perform navigational, pointing and selection actions as with conventional pointing devices.

Explicitly, eyeball and head displacements with respect to a fixed reference are interpreted as pointer drags, or *navigational actions*. Pointers can also trigger a *selection action* (click) as they persist in the neighborhood of a geometric configuration for a convenient time interval. Since the production of discrete events is reduced here to the on-off mechanism of time thresholding for selection, pointers have basically the same semantic expressivity of a 1-button mouse. This design choice virtually demands the whole burden of expressivity from the interaction environment.

The semantics implemented attempts to assign a natural meaning to the user's head and eye movements (basically head translations and rotations and pupil shifts of fixation) so that even a naive user can feel comfortable interacting with the system. The head can be used to navigate or to displace objects (drag) in the environment and also to select, or "freeze," a 3D scene of interest (click). Eyeball drags, which take into account a displacement within the scene, can be used e.g., for on-screen drawing or exploration, and eyeball clicks to select relevant 2D information.

8.3 Implementation

This section discusses the modeling and measurement aspects of our drag and click approach to vision-based interaction.

8.3.1 Projection and user modeling

Affine projection model. The relative geometry between any two frames is described by six DOF. For example, the transformation between the user and camera frames can be expressed as

$$ {}^{\gamma}_{\beta}\mathrm{T} = \begin{bmatrix} & \mathrm{R}(\tau,\sigma,\phi) & & \boldsymbol{t} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad , \tag{8.3} $$

where $\boldsymbol{t} = [t_1\, t_2\, t_3]^{\mathrm{T}}$ is a translation 3-vector, and $\mathrm{R}(\tau,\sigma,\phi)$ is a 3×3 rotation matrix, completely defined by the three angles τ (tilt), σ (slant) and ϕ (orientation). Specifically, if each of the interaction frames is chosen with its third axis normal to the associated plane (see again Figure 8.2), then $\sigma \in [0, \pi/2]$ is the angle between the face and the image planes, vanishing identically if the two planes are parallel.

Affine perspective projection† of a face point onto the camera plane, ${}^{\Gamma}_{B}h$, can be described in terms of a translation 2-vector $\boldsymbol{h}$, and a 2×2 projection matrix

$$ \mathrm{H} = \kappa \begin{bmatrix} \cos\tau & -\sin\tau \\ \sin\tau & \cos\tau \end{bmatrix} \begin{bmatrix} -\cos\sigma & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \sin\phi & -\cos\phi \\ \cos\phi & \sin\phi \end{bmatrix} \quad , \tag{8.4} $$

which is the composition of two planar rotations by τ and $\phi - \pi/2$, of an anisotropic scaling by $\cos\sigma$ along a specific image direction, and an isotropic scaling of the imaged pattern by a factor κ proportional to the camera's focal length [86].

User features being tracked. Head displacements and pupil positions can be estimated at one time by tracking the visual appearance of the users' eyes. Indeed, the *external contour* of the eye, which is fixed to the head‡, can be related to head displacements, while the internal *iris* contour, which is concentric with the pupil, moves according to user

† We assume here that all intrinsic parameters of the projection, and especially the pixel size ratio, have their ideal values. In such a case, the affine projection is a weak perspective [232].

‡ Our rigidity constraint assumes that any change of the visual appearance of the eye contour is related to a head displacement. At a very short time scale, even a nonrigid change of facial expression — say, passing from normality to utmost stupor — influences the external eye appearance. However, at longer time scales only geometric-driven changes are significant.

pointing actions (eyeball shifts). Analyzing separately the pupil pointing and head displacement characteristics of user actions allows us to adopt a two-step eye tracking approach:

(i) *Track the external eye contour.* This determines a search bound for the pupil's position in the image.

(ii) *Track the pupil inside the search bound.* This is motivated by the physiological constraint that the pupil must always lie inside the external eye contour.

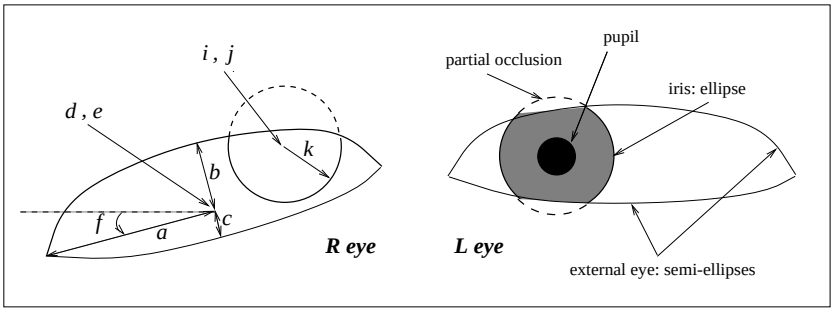

Fig. 8.3. Modeling the left and right eyes.

Image measurements for head tracking and pupil detection are obtained by deforming a reference elastic template so as to adapt it to current image data. Let us assume that the reference frame ρ is the β-frame at initialization time ($t = 0$), i.e. $\rho = \beta(0)$. Let's also assume that ρ and γ have mutually parallel third axes, i.e. $\sigma(0) = 0$. Then, by eq. (8.4), the reference template models a frontoparallel view of the eyes, possibly translated by $\boldsymbol{h}(0)$, and scaled and mirrored w.r.t. the face plane content by

$$\mathrm{H}(0) = \kappa(0) \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad . \tag{8.5}$$

The template is composed of two sub-templates, which capture the characteristics of the user's external eye and iris, respectively (Figure 8.3).

The external eye sub-template features two semi-ellipses sharing their major axis, thus depending on six parameters, namely the common major axis (a), the two minor axes (b, c), the ocular center image coordinates (d, e), and the common orientation (f). A circle, parameterized through the coordinates of its center (i, j) and its radius (k), is used for the iris

sub-template. Notice that part of the iris is usually occluded by the eyelids: this fact has to be explicitly taken into account in the tracking strategy.

8.3.2 Sensing the user

Template initialization. At system startup, a raw estimate of the eye location and shape in the image is derived, and the template is accordingly initialized. To speed up processing, the two image regions containing the eyes are first roughly located by means of *edge dominance maps*, i.e., maps which take into account the dominance of brightness gradient in a given direction [63]. Figure 8.4 illustrates the two-step procedure for automatic eye localization.

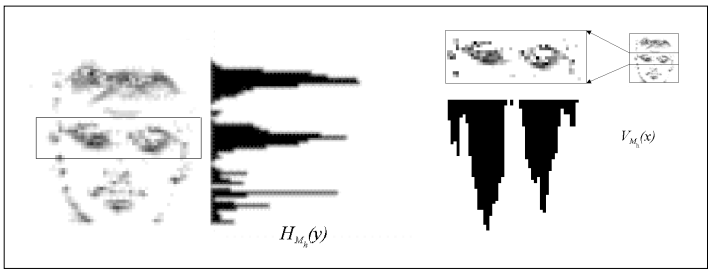

Fig. 8.4. Eye localization using dominance maps.

Once the regions including the eyes are found, the template is adjusted against image data by an energy-minimization criterion (see e.g., [337]). A quadratic energy term, including the $9 = 6 + 3$ template parameters per eye, is minimized, so that the template is relaxed inside the eye regions by gradient descent. After relaxation, the deformable template is used to initialize the run-time tracker, a lightweight process which allows for faster tracking behavior than template relaxation.

Tracking the external eye. The tracker is modeled after the external eyes only. Its image shape is controlled by a discrete set of points, which are initialized soon after template relaxation, by uniformly sampling the external eye contours. At run-time, the tracker parameters are refined and updated by means of a simple tracker-to-image fitting approach, based on least squares and the extraction of brightness edge points (Figure 8.5). Thanks to our affine projection model, the tracker can be made quite robust by allowing it to deform only in an affine fashion.

Explicitly, let the tracker instance at a generic time $t > 0$ be $\{\boldsymbol{x}_i(t)\}$, $i = 1 \ldots N$, let its centroid be $\boldsymbol{x}_{\text{B}}(t)$ and let the reference tracker be $\{\boldsymbol{x}_i(0)\}$ with centroid $\boldsymbol{x}_{\text{B}}(0)$. Template tracking proceeds as follows:

(i) *Prediction.* In an image neighborhood of the previous tracker instance (time $t-1$), a local search of edge points (brightness gradient maxima) takes place at each of the N tracker points and along normal directions to the tracker itself. The set $\{\widetilde{\boldsymbol{x}}_i(t)\}$ of edge points (predicted set) is computed by means of a 5-point, recursive coarse-to-fine algorithm based on finite differences [100]. Search intervals along each direction are adaptively adjusted based on the previous tracking results.

(ii) *Least squares fit.* The LS approximation of the new tracker centroid is simply the centroid (average point) of the predicted set evaluated in the previous step, i.e. $\widetilde{\boldsymbol{x}}_{\text{B}}(t) = \frac{1}{N}\sum_i \widetilde{\boldsymbol{x}}_i(t)$. A 2×2 matrix $\widetilde{\text{L}}(t)$ is also evaluated via LS as the best approximation of the affine transformation about the origin between the predicted set and the reference template. This is done by minimizing the quadratic cost $\sum_i \|(\widetilde{\boldsymbol{x}}_i(t) - \widetilde{\boldsymbol{x}}_{\text{B}}(t)) - \widetilde{\text{L}}(t)\,(\boldsymbol{x}_i(0) - \boldsymbol{x}_{\text{B}}(0))\|^2$ by solving the linear homogeneous system obtained by partial differentiation w.r.t. the unknown four parameters.

(iii) *Filtering.* The six parameters of the affine image transformation $(\widetilde{\text{L}}(t), \widetilde{\boldsymbol{x}}_{\text{B}}(t))$ are smoothed using a recursive filter. For the generic parameter p, the filtered value is $\widehat{p}\,(t) = w_p\, p(t) + (1 - w_p)\,\widehat{p}\,(t-1)$. To achieve a better control of the tracking process, a different filter gain $w_p \in [0, 1]$ can be assigned to each parameter.

(iv) *Affine projection.* Once the affine transformation $(\widehat{L}(t), \widehat{\boldsymbol{x}}_{\text{B}}(t))$ is obtained, the new tracker instance is finally computed as $\boldsymbol{x}_i(t) = \boldsymbol{x}_{\text{B}}(t) + \widehat{\text{L}}(t)\,(\boldsymbol{x}_i(0) - \boldsymbol{x}_{\text{B}}(0))$ for all i, with $\boldsymbol{x}_{\text{B}}(t) = \widehat{\boldsymbol{x}}_{\text{B}}(t)$. This last step ensures that at each time $t > 0$ the tracker is an affine-transformed instance of the reference tracker, initialized at $t = 0$. Besides, such a tracking approach is independent of the specific tracker being used: different eye shapes can be tracked by simply changing the shape of the reference template — say, using parabolic arcs instead of elliptic arcs.

Locating the pupil. Once the new external eye position is found, it is necessary to search for the current iris shape and position inside the new tracker contour. This is done according to a technique akin to the one used before, but also different from that in some significant points:

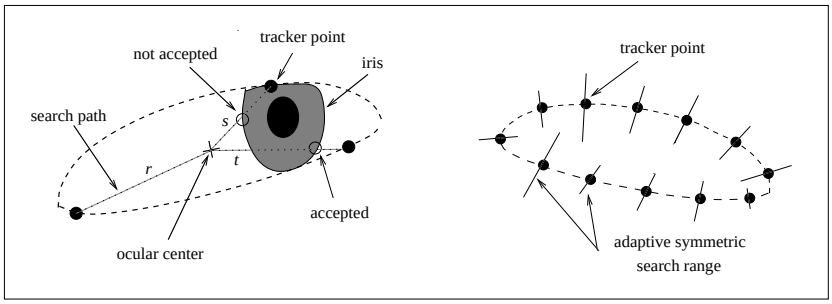

Fig. 8.5. Affine-deformable eye tracker and search of iris candidate points.

- *Affine model.* To deal with generic face views with nonzero slant angles† which would produce an elliptic-shaped iris, an elliptic iris tracker is considered. This is initialized with the reference circular iris obtained after template relaxation.
- *Occlusions.* To avoid erroneous results due to possible partial iris occlusions, the edge points belonging to the external eye tracker are automatically excluded from LS fitting with the iris tracker; in this way, an external contour point is not taken into account as a predicted iris point $\widetilde{\boldsymbol{y}} = [\widetilde{x}\ \widetilde{y}]^{\mathrm{T}}$. To avoid occlusions, we use edge search lines which are radial paths connecting external eye tracker points with the tracker's centroid: the search proceeds inwards and, as shown in Figure 8.5, only negative gradient points are taken into account.
- *Matching.* The steps of fitting and affine projection are simultaneous. This is obtained by explicitly using the elliptic tracker analytic expression into the objective function. Specifically, the new raw ellipse parameters are extracted by fitting predicted points with a generic conic via LS, and then constraining the conic to be an ellipse. That is, found the generic five conic parameters c_l which minimize the squared error $\sum_i \left[\widetilde{x}_i^2 + c_1\,\widetilde{x}_i\widetilde{y}_i + c_2\,\widetilde{y}_i^2 + c_3\,\widetilde{x}_i + c_4\,\widetilde{y}_i + c_5\right]^2$, these are expressed in terms of ellipse center, major and minor axes, and orientation [85].

After the filtering step, the current center of the iris is taken as the new location of the pupil and used to update the 2D pointer.

† This proves useful when the intrinsic camera parameters such as the pixel size ratio are not perfectly compensated.

8.3.3 Controlling the environment

Using the 3D pointer. Head displacements relative to the reference frame can be estimated from a comparison between the current and reference trackers. At any time $t \geq 0$, the tracker is obtained by a camera projection of the face plane when the face frame is $\beta(t)$; following the notation of Subsection 8.3.2, we denote this projection as $(\mathrm{H}(t), \boldsymbol{h}(t))$, where the generic tracker is related to the reference tracker through the affine transformation $(\mathrm{L}(t), \boldsymbol{x}_{\mathrm{B}}(t))$ s.t. $\boldsymbol{x}_{\mathrm{B}}(t) = \boldsymbol{h}(t)$ and $\mathrm{L}(t) = \mathrm{H}(t)\,\mathrm{H}^{-1}(0)$.

If we can assume frontoparallel interaction, i.e., that the user's face remains almost parallel to the screen during the whole interaction time†, then $\sigma(t) \approx 0\ \forall t > 0$,

$$\mathrm{H} = \kappa \begin{bmatrix} -\sin\phi & \cos\phi \\ \cos\phi & \sin\phi \end{bmatrix} \quad , \tag{8.6}$$

and rotations are fully described by the orientation angle $\phi(t)$ in $[0, 2\pi]$, initialized to $\phi(0) = -\pi/2$ to satisfy eq. (8.5). Notice that during frontoparallel interaction the head DOF are reduced from six to four — namely, a translation 3-vector and a rotation in the face plane. The head DOF can be conveniently referred to the ρ-frame as $\boldsymbol{t}(0) - \boldsymbol{t}(t)$ and $\phi(t)$, and related to 2D quantities as follows:

$$\boldsymbol{t}(0) - \boldsymbol{t}(t) = 1/\kappa(0) \left(\begin{bmatrix} \boldsymbol{h}(0) \\ \lambda \end{bmatrix} - \frac{\kappa(0)}{\kappa(t)} \begin{bmatrix} \boldsymbol{h}(t) \\ \lambda \end{bmatrix} \right) \quad ; \tag{8.7}$$

$$\begin{bmatrix} \sin\phi(t) & \cos\phi(t) \\ -\cos\phi(t) & \sin\phi(t) \end{bmatrix} = -\frac{\kappa(0)}{\kappa(t)}\,\mathrm{L}(t) \quad . \tag{8.8}$$

Given focal length λ and an estimate of $\mathrm{L}(t)$, $\boldsymbol{h}(t)$ and $\boldsymbol{h}(0)$, from eq. (8.8) it is possible in principle to recover rotation $\phi(t)$ and relative scaling $\kappa(0)/\kappa(t)$: the latter can be used into eq. (8.7) to recover translation parameters up to an unknown scale factor‡ $1/\kappa(0)$.

However, we prefer to use a more direct approach to evaluate relative scale and rotation from the current and reference trackers — see Figures 8.6 (*a* through *d*). The approach does not involve eq. (8.8) but instead exploits the ocular centers as located in the image. Observe in fact that relative scaling can be estimated as the interocular distance ratio $\delta(0)/\delta(t)$ (Figure 8.6, *c*). Similarly, as shown in Figure 8.6 (*d*), the

† The frontoparallel is a very natural way to stand in front of a computer, as it is optimal in terms of ergonomics.

‡ This indetermination originates from modeling not the eyes themselves in the face plane but, instead, their image appearance.

relative inclination of the line connecting the ocular centers can be used to estimate rotation.

The estimated 3D parameters correspond one-one to the four basic head displacements which the user can mix together during frontoparallel interaction.

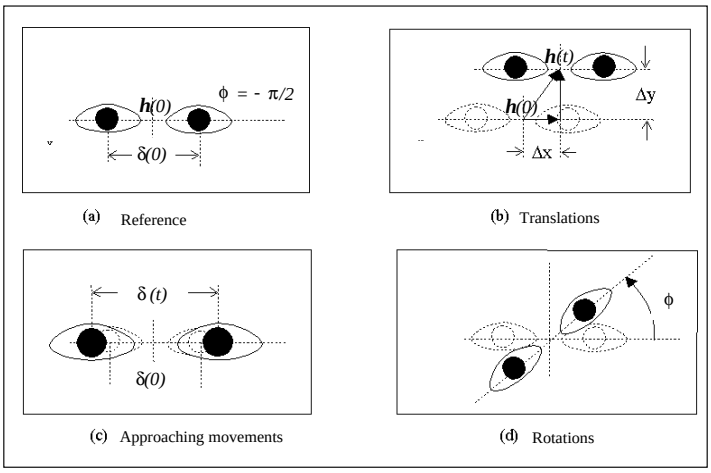

Fig. 8.6. The four head DOFs considered for the experiments.

Calibrating and using the 2D pointer. Remapping the 2D pointer has a more quantitative nature. The key idea for such a pointer is in fact to be able to know where the user is actually looking in the screen. Such a strategy is usually implemented using precise but otherwise intrusive infrared light equipment [159, 154] and allows the eye pupil to be used as a 2D mouse.

Having denoted the affine eye projection map by $(\mathrm{K}, \boldsymbol{k})$, having defined a new camera projection map as $(\mathrm{H}', \boldsymbol{h}')$, with $\boldsymbol{h}'$ the projection of the ocular center of the eye where the pupil is located†, and having expressed image and screen points $\boldsymbol{c}$ and $\boldsymbol{a}$ as 2-vectors using their native coordinate frames γ and α respectively, it holds:

$$^{\alpha}\boldsymbol{a} = \mathrm{M}\,^{\gamma}\boldsymbol{c} + \boldsymbol{m} \quad , \tag{8.9}$$

where $\mathrm{M} = \left[\mathrm{H}'\mathrm{K}\right]^{-1}$ and $\boldsymbol{m} = -\mathrm{M}\,\boldsymbol{h}' - \mathrm{K}^{-1}\,\boldsymbol{k}$.

† Such a projection is also the centroid of the external eye sub-tracker including the pupil's image. Vectors $\boldsymbol{h}$ of eq. (8.7) and $\boldsymbol{h}'$ differ by the centroid of the remaining sub-tracker.

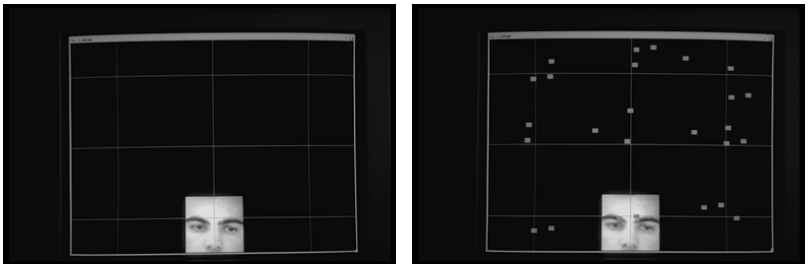

Fig. 8.7. 2D calibration. *Left*: Calibration grid. *Right*: Remapped points.

The camera-to-screen map $(\mathrm{M}, \boldsymbol{m})$ involves time-dependent quantities, some of which — e.g., the distance from eye to screen and eye focal length — are not known or difficult to know. Nevertheless, this transformation can be estimated at startup via a plane-to-plane calibration procedure:

(i) *Observation.* A set of $M \geq 6$ image observations of pupil positions is collected and recorded, obtained by tracking the pupil while letting the user execute a sequence of gaze fixations on M given screen locations. Figure 8.7 (*left*) shows the case $M = 9$, where the locations to fix are organized in a 3×3 calibration grid covering the entire screen. To reduce errors during this phase, each new observation is obtained as an average of subsequent image measurements.

(ii) *Estimation.* The affine model is then estimated using data and observations, as the LS solution of an overdetermined system obtained from eq. (8.9). LS computations are carried out in two steps, by first computing data and observation centroids ${}^{\alpha}\boldsymbol{a}_{\mathrm{B}}$ and ${}^{\gamma}\boldsymbol{c}_{\mathrm{B}}$ and estimating the matrix M from $({}^{\alpha}\boldsymbol{a} - {}^{\alpha}\boldsymbol{a}_{\mathrm{B}}) = \mathrm{M}\,({}^{\gamma}\boldsymbol{c} - {}^{\gamma}\boldsymbol{c}_{\mathrm{B}})$, and then obtaining the translation as $\boldsymbol{m} = {}^{\alpha}\boldsymbol{a}_{\mathrm{B}} - \mathrm{M}\,{}^{\gamma}\boldsymbol{c}_{\mathrm{B}}$.

At run-time, pupil positions in the image are remapped onto the screen using the calibrated map. Figure 8.7 (*right*) shows the qualitative results of a test on calibration accuracy. In the test, the user is asked to fix the calibration grid points successively, while the corresponding image pupil positions, suitably averaged, are remapped onto the screen using eq. (8.9). The few remapped points in the figure which are halfway between grid crossings are due to user eye movements between successive fixations.

Quantitative results on the same calibration test are shown in Table 8.1. This reports the average and maximum mismatch between the set of calibration points ("ground truth") and the remapped points both in the two coordinate directions and in magnitude. Magnitude errors provide us with a resolution limit for our 2D pointer, which must be taken into account in the design of the graphic interface. To reduce map reconstruction errors during calibration, head motions can be compensated by normalizing the image plane observations w.r.t. the reference ones. Similarly, the calibrated map can be updated at run-time based on current visual data [87].

CALIBRATION ACCURACY	*Error on* x	*Error on* y	*Uncertainty radius*
Average (mm)	6.23	10.69	12.38
Maximum (mm)	17.51	19.93	26.53

Table 8.1. *Calibration accuracy.*

Pointer timing. Choosing the right threshold value is of key importance for 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* [159]. A good dwell time for the 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. Since head motions are slower, a time threshold about three times longer can be used for the 3D pointer.

A second timing mechanism is implemented in the interface, avoiding the occurrence of "interference" between the pointers. Consider for example the case of a 3D click, which occurs when the head persists in a fixed position for one second or more. If, in the same time period, the pupil is also fixed, then a 2D click event is generated, which is most probably involuntary. To solve the interference problem, the 2D and 3D pointers are decoupled, by letting the pupil be actually remapped only if the head is in a suitable neighborhood of its reference position — i.e., if the current tracker is not too different from the reference tracker.

8.4 Interface

An interface has been implemented to support interaction using our visual pointers within a virtual museum containing a digital collection of canvases by famous 20th century artists such as Klimt, Picasso, Modigliani, etc. The virtual environment is complemented by a 2D on-screen hypertext providing the museum with on-line catalogue facilities.

The graphic interface uses the OpenGL graphic libraries and runs on a Silicon Graphics Indy workstation (MIPS R4000/R4010 with 100 MHz clock). The vision subsystem software also runs on the Indy, and gets raw image data through a VINO frame grabber board from an off-the-shelf B/W camera.

To achieve natural interaction, the loop delay must be short enough to provide the user with feedback about his latest action. The overall interaction loop time for our system is the sum of the time spent doing visual computations and graphic environment manipulation. Initializating visual algorithms involves automatic eye extraction and template initialization and takes around 450 ms to complete. At run-time visual tracking runs almost at video rate (50 ms) instead, using $N = 64$ sampling points for both external eye and iris search. Without special hardware for graphics acceleration, most of the loop time is taken by graphic remapping (some hundreds ms at an intermediate image quality).

3D viewpoint control and navigation. Examples of a typical interaction session using the 3D pointer are illustrated in Figures 8.8 and 8.9 (time flows top to bottom and left to right). The screen content and a mirrored image of the user's eye region are shown, in order to emphasize the relationship between user action and environment changes.

Figure 8.8 shows a zoom-in sequence. Zooming is obtained by approaching the screen with the head; this causes the painting in the middle of the wall to be displayed at full resolution. A new viewpoint can always be selected by head fixation. However, in this case the user decides to go back to the initial view by moving away from the screen (last image of the sequence).

Another kind of viewpoint adjustment can be obtained by head rotation around the visual axis (Figure 8.9). This proves to be useful when interesting on-screen features do not have the right orientation. Thanks to the remapping semantics, a clockwise head rotation produces a natural counterclockwise rotation of the environment.

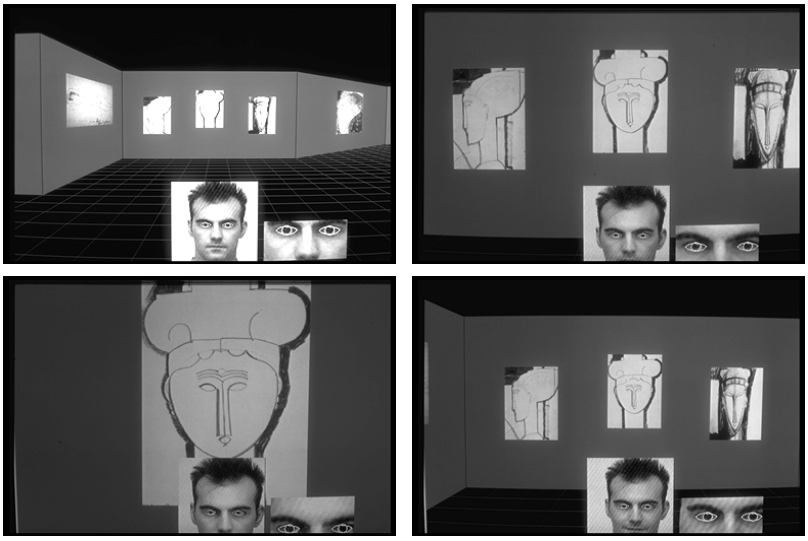

Fig. 8.8. 3D pointer: zoom sequence.

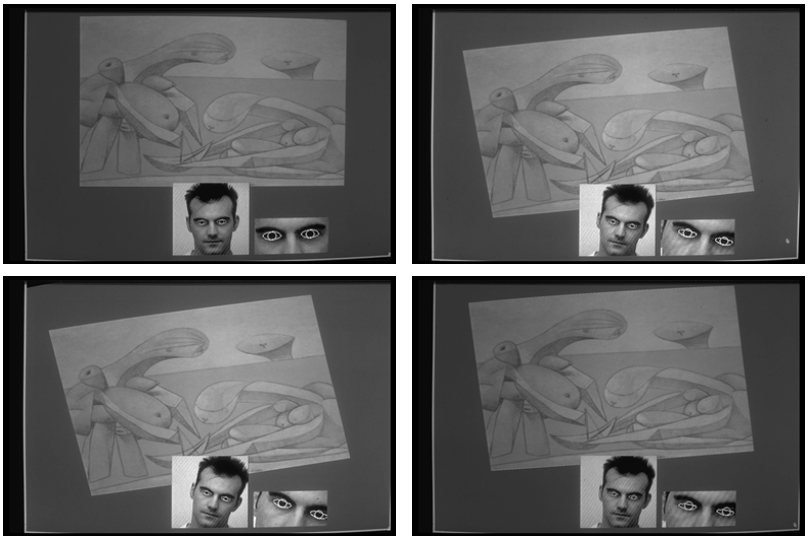

Fig. 8.9. 3D pointer: rotation sequence.

Viewpoint changes are produced by navigational actions corresponding to head translations parallel to the screen. This excludes, in principle, the possibility of executing head pan movements, as the constraint on frontoparallel interaction would not be met. For example, panning the head leftwise would be interpreted in our interface as a rigid leftward head translation (correctly remapped as a rightward environment shift), *and* a decrease of interocular distance, incorrectly remapped as a zoom-out. Thus, the correct image resolution must be recovered by an additional zoom-in action.

Besides, notice that proper semantics must be introduced to generate rotations in the environment using our 4-DOF pointer. In our interface, if a side museum wall is reached by a translational movement, an environment rotation is automatically generated, so as to show the paintings on that wall.

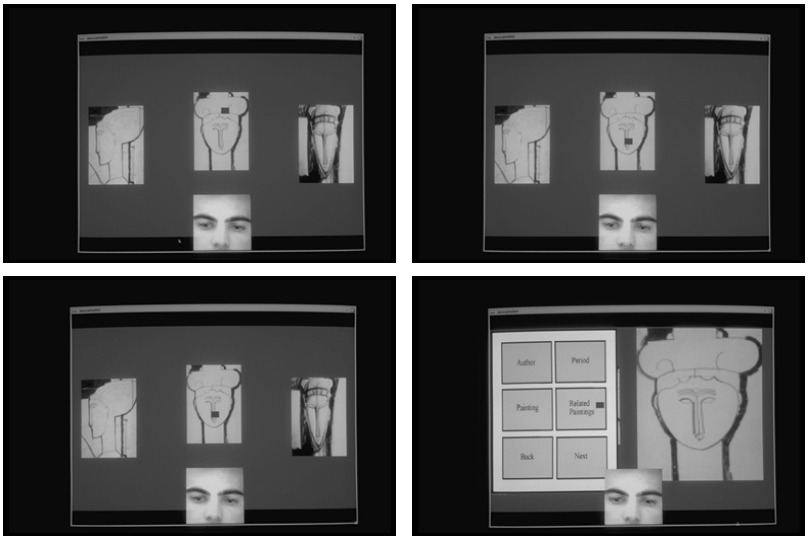

Fig. 8.10 2D selection. The 1st menu level is accessed from level 0 by steadily pointing the eye to a specific painting.

2D inspection and selection. Once a specific viewpoint has been selected by head rotation, the user can go back to the reference position, inspect the on-screen scene, and possibly learn more about a given painting in it, or about its author, by selecting a canvas by pupil pointing (Figure 8.10).

An important aspect of the interface concerns the screen regions which can accept eye-gaze commands, which we refer to as *active windows*. At menu level 0, as commands are issued directly by fixing a canvas, each canvas is an active window. At menu level 1 (Figure 8.10, *bottom right*), the screen is partitioned in two regions. The left-hand region has six active windows, four of which allow access to information about the author, the historical period, selected and related paintings; the remaining two windows are used for paging control ("back," "next"). The right-hand side of the screen contains output information, and as such it can be fixed by the user without timing constraints (passive window). As the first menu level is accessed, the passive window is filled by default with the painting which was selected in the museum. Figure 8.11 shows the process of getting information about the author of the painting: the "author" button is repeatedly fixed, and this causes the author information to be pasted into the passive window to be read.

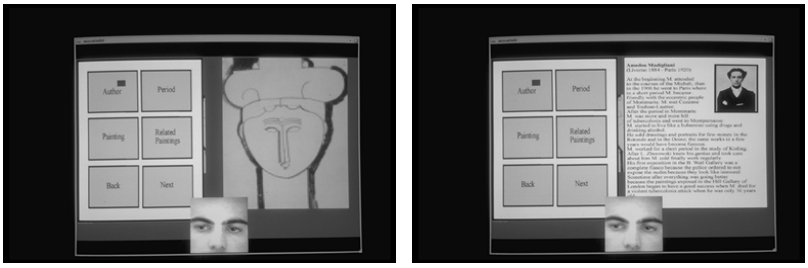

Fig. 8.11. 2D selection: second menu level.

The spacing and size of active windows strictly depend on the magnitude of the pupil remapping error (see again Table 8.1). The average pointing error can be used to properly design the 2D interface and size-up active windows in order to reduce the risk of false alarms during slightly incorrect pointing.

8.5 Innovation

In this chapter, we have presented an approach to advanced human-machine interfacing based on computer vision and computer graphics. The key idea is to use computer vision to develop non intrusive pointer devices based on user head and eye motions, and to couple them with a simple drag and click semantics for interaction in graphic 2D and 3D

environments. Basically, the pointers capture significant parameters of user action, which are then remapped onto the environment. The main operations which the user can perform are environment exploration and data selection. Head tracking is used to interact with a 3D environment, while eye pupil tracking allows to control interaction with 2D-based interfaces. To implement the 3D pointer, we continuously track the image of the external part of the user's eyes, and use its deformations to infer head motion parameters. Concerning the 2D pointer, we propose an approach derived from infrared technology applications, and novel to computer vision, i.e., to track eye pupils so as to measure gaze pointing actions in terms of observed screen points rather than simply as directions in the 3D space.

Some directions for future work with our visual pointer technique can finally be outlined. One way to cope with uncertainty in measurements and human intentions alike is to use *adaptive interfaces*. Such interfaces should cooperate with the user in making a decision, either by allowing him to refine a command, or by suggesting a series of alternatives during interaction. The interaction framework can be extended to deal with *remote environments*. Indeed, while interaction with the on-screen environment always takes place locally, the displayed scene needs not to be necessarily a graphic environment, but could be e.g. the output of a remote, possibly motorized, camera. Thus our approach can be effectively extended to applications such as teleconferencing, telepresence and teleoperation. Other interesting applications are in the field of *augmented reality*. Think of replacing the screen plane with a real-world plane, say a desk, an office wall or some other work panel. The approach can be easily adapted to this new scenario. Another extension of the approach is to allow for *richer semantics at the user level*. That is, besides the basic interaction capabilities offered by our approach, the user could be provided with other means of interfacing with the machine. A way to extend semantics naturally in a computer vision context can be that of interpretation of gestures and expressions.

Bibliography

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

[63] R. Brunelli and T. Poggio. Face recognition: Features versus templates. *PAMI*, 15(10):1042–1052, October 1993.

[85] C. Colombo, S. Andronico, and P. Dario. Prototype of a vision-based gaze-driven man-machine interface. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems IROS'95*, pages 188–192, 1995.

[86] C. Colombo and J.L. Crowley. Uncalibrated visual tasks via linear interaction. In *Proc. 4th European Conference on Computer Vision ECCV'96*, pages 583–592, 1996.

[87] C. Colombo and A. Del Bimbo. Interacting through eyes. *Robotics and Autonomous Systems*, 19(3–4), 1997. (To appear.)

[100] R. Curwen and A. Blake. Dynamic contours: Real-time active splines. In A. Blake and A. Yuille, editors, *Active Vision*, chapter 3, pages 39–57. MIT Press, 1992.

[119] Essa, I., Pentland, A., (1994) A Vision System for Observing and Extracting Facial Action Parameters, In *Proc. Computer Vision and Pattern Recognition*, pp. 76-83, Seattle, WA., June 1994.

[154] T.E. Hutchinson. Computers that sense eye position on the display. *IEEE Computer*, 26(7):65–67, 1993.

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

[175] R. Kaucic, B. Dalton, and A. Blake. Real-time lip tracking for audio-visual speech recognition applications. In *Proc. 4th European Conference on Computer Vision ECCV'96, Cambridge, England, April 1996*, pages 376–387, 1996.

[232] J. L. Mundy and A. Zisserman, editors. *Geometric Invariance in Computer Vision*. MIT Press, Cambridge MA, 1992.

[246] A.P. Pentland. Smart rooms. *Scientific American*, 274(4):54–62, 1996.

[262] H. Rheingold. *Virtual Reality*. Secker and Warburg, 1991.

[286] Starner, T., and Pentland, A., Visual Recognition of American Sign Language Using Hidden Markov Models, In Proc. Int. Workshop on Automatic Face and Gesture Recognition Zurich, Switzerland, pages 189–194, June 1995.

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

[337] A. Yuille and P. Hallinan. Deformable templates. In A. Blake and A. Yuille, editors, *Active Vision*, chapter 2, pages 21–38. MIT Press, 1992.