Visual Capture and Understanding of Hand Pointing Actions in a 3D Environment

Carlo Colombo, Alberto Del Bimbo, and Alessandro Valli

***Abstract*— We present a non intrusive system based on computer vision for human-computer interaction in 3D environments controlled by hand pointing gestures. Users are allowed to walk around in a room and manipulate information displayed on its walls by using their own hands as pointing devices. Once captured and tracked in real-time using stereo vision, hand pointing gestures are remapped onto the current point of interest, thus reproducing in an advanced interaction scenario the "drag and click" behavior of traditional mice. The system, called *PointAt*[1], enjoys a careful modeling of both user and optical subsystem, and visual algorithms for self-calibration and adaptation to both user peculiarities and environmental changes. The concluding sections provide an insight into system characteristics, performance, and relevance for real applications.**

***Index Terms*— Computer Vision, User-Adapted Human-Computer Interaction, Hand Pointing, 3D Visual Modeling and Real-Time Tracking.**

I. Introduction

In modern computer engineering, special care is devoted to the design of human-machine interfaces enabling users to communicate efficiently with the system [1]. Interfaces usually enable a bi-directional communication: on the one hand, input devices allow users to issue commands to the system; on the other hand, output devices provide users with both responses to commands and feedback about user action [2]. In a standard graphic user interface, keyboard and mouse are typical input devices, used to type commands, to write text, and to point and select graphic elements at specific locations of a graphic display. The display, usually integrated with loudspeakers, outputs information which is related to commands and selections performed by the user. The graphic interface features a bi-dimensional (2D) graphic environment where icons are "clickable" elements representing procedures or pieces of multimedia information. Other conventional input devices often used in standard interfaces are the 3D mouse and the joystick, and are typically coupled with three-dimensional graphic environments. Standard interfaces have the advantage of working equally well in every working place, but physically limit the mobility of the user, who is typically constrained to sit in front of the computer monitor.

Advanced user interfaces, such as those employed in many augmented and virtual reality applications, ensure a higher user mobility [3]. The input devices for advanced interfaces can either be wearable or non wearable devices. The first category includes, among the others: data helmets and glasses (complemented by head-mounted displays), data gloves (with or without force feedback), body markers (with associated marker detection software), and smart card technologies. Usually accurate in estimating user position and action, wearable devices can be effective in supporting sophisticated interaction, close to that experienced by the user in the real world. Nevertheless, they are intrusive, and force the user to adopt new instruments and sensors as prolongations of his senses: this usually induces in the user a feeling of uneasiness.

Non wearable devices basically include systems based on non-contact sensors, such as cameras (with image processing and computer vision software) and microphones (with audio processing and voice recognition software). These devices are non intrusive, and can support natural interaction as the user can express commands and actions through voice and gestures in the same way as in everyday life. However, in order for image and audio understanding algorithms to work appropriately, several conditions (e.g. illumination and acoustic characteristics, number and mobility of the persons sharing the working space, etc.) often have to be set on the interaction environment, thus limiting the number of possible application scenarios and posing objective constraints on their general applicability [4].

An active research trend in computer vision is the development of robust, environment-independent tracking methodologies for the development of effective human-computer interfaces [5], [6]. Several vision-based interaction approaches have been presented so far. In some of them, the aim is to derive a semantic interpretation of human hand gestures and facial expressions [7], [8], [9]. Other approaches capture instead the geometric aspects of user action to develop advanced interaction devices based on full body, hand, head or eye motion [10], [11], [12], [13]: in this category, vision-based hand pointing systems appear to be particularly promising. In fact, since hand pointing is an everyday life operation reflecting a specific interest into a specific portion of the visible space [14], it does not require any a priori skills or training, and is a perfect candidate for the design of a natural interaction device based on computer vision [15].

A simple example of vision-based hand pointing interface is the "digital desk" introduced in [16], and re-proposed with diverse enhancements in the "virtual touchscreen" and "finger paint" systems described respectively in [17] and [18]. All these systems share the same elements, i.e. a video projector, a camera and a planar surface (screen). The camera is located so as to have the screen in view. System output is displayed by the projector onto the screen, whose locations can be pointed at by direct contact of the index finger. The term "virtual" arises from the fact that the screen, although actually not touch-sensitive, is apparently so, thanks to the evaluation of the finger contact point carried out in real-time by simple

[1]Patent pending.

image analysis techniques. An extension of the touchscreen concept, in which the user is not constrained to physically touch the screen, is discussed in [19]. In this case, the system includes two cameras, which, as before, must be placed in the interaction environment in order to have both the pointing hand and the screen in view; the cameras must also be far enough from the user to ensure that the conditions for weak perspective (affine) projection are met. The user can interact with the system by adopting the rigid "pistol" pointing gesture, thus enabling the vision system to evaluate the finger direction in each image by active contour tracking and compute the screen location currently pointed at by stereo triangulation. The shooting hand configuration is also used in the "finger pointer" system described in [20], but only to locate the fingertip, and not to evaluate the finger direction. In fact, in this case, the pointing direction is modeled as the direction of the line joining the fingertip and the so-called "magic point," approximately located between the eye and hand positions: the interest location in the screen is computed by intersecting in 3D space this direction with the screen plane. The system requires two cameras, which are not constrained, as before, to have the screen in view; anyway, since magic point calibration is a delicate task, users have to interact with the system from a predefined and fixed position. In order to allow users to move inside a wider interaction environment without loosing hand pointing resolution, the approach exposed in [21] uses an active pan/tilt stereo system to track the pointing hand. In this work, an attempt is also made to eliminate the fixed position constraint of the previous work. To this aim, the magic point is replaced by the corresponding eye, whose 3D position is roughly inferred by combining a special calibration procedure with anthropometric considerations. However, the eye-to-fingertip remapping method thus defined constrains the user to adopt a pointing style in which eye, fingertip and interest points are collinear.

This paper presents a novel vision-based approach to hand pointing, aimed at preserving naturality of interaction and ensuring usability through a careful modeling of the various elements of the interaction environment. Users are allowed to walk around freely in a room and select information displayed on a wall screen by using their own hands as pointing devices. Once captured and tracked in real-time, hand pointing actions are remapped onto the screen, thus reproducing in an advanced interaction scenario the "drag and click" behavior of traditional 2D pointers. The approach has evolved from an earlier and simpler prototype based on color tracking and a pair affine cameras [22]. The main characteristics of the system, which was called *PointAt*, can be summarized as follows:

- The system runs in real-time with standard, low cost equipment. The screen point of interest is computed through a line-based stereo approach which, for the sake of numerical accuracy, does not involve explicit 3D line measurements.
- The interaction model is independent of the number of cameras used, their minimum number being two. The model is compatible with both the full perspective and the affine projection camera models.
- No explicit constraints are set on camera placement; specifically, it is not required that the screen be visible by the cameras. A specific camera layout may induce the use of the full perspective model instead of the simpler affine model.
- The user is allowed to move freely while pointing. Two different adaptive visual tracking subsystems run simultaneously, thus making the system largely independent of environmental changes and user position; yet, depending on a specific camera layout, certain user positions may cause the system to rely on only one of the two tracking subsystems, with a consequent loss of precision.
- Users are not requested to calibrate the system before interacting with it: self-calibration at run time ensures adaptation to user characteristics such as physical dimensions and pointing style.

An extensive experimental section and a prototype system implementation in a real application context conclude the paper, providing an insight into system characteristics, performance, and relevance to applications.

The paper is organized as follows. The next section, opening with an overview of a typical hand pointing application, provides a comprehensive discussion of system design choices, encompassing geometric models (§II-B), image analysis algorithms (§II-C), and adaptation strategies (§II-D). Section III reports on system setup and experimental evaluation. Then, in section IV a recent implementation of *PointAt* in a real application context is illustrated. Finally, in section V conclusions are offered and some directions of future work outlined.

II. System Design

A. Overview

To introduce the main elements of hand pointing interfaces, let us refer to the typical application scenario currently under installation in the Museum of Palazzo Medici Riccardi of Florence (for details, see section IV). Fig. 1 (left) shows a room provided with a large screen, on which paintings are displayed by a computer. Users can ask the system to display on the screen additional information about specific parts of the paintings by pointing at the screen locations of interest. The main interaction elements involved are shown in Fig. 1 (right). The system gets its input from a pair of cameras placed so as to have the user in view; a computer performs image analysis and computes the screen location the user is currently pointing at.

Interface operation is based on both spatial and temporal characteristics of user action. On the spatial side, the screen location P_s currently pointed at by the user is continuously evaluated as the intersection of the pointing direction with the screen plane. To this end, the parameters encoding hand pointing direction are tracked from both images, and then used as the input of a stereo triangulation algorithm. On the temporal side, the system monitors pointing persistency: as point P_s remains inside a limited portion of the screen for an appropriate amount of time (e.g. about one second), a discrete event similar to a mouse click, i.e. a selection action, is generated for the interface. In conclusion, the overall interaction system behavior is that of a one-button mouse,

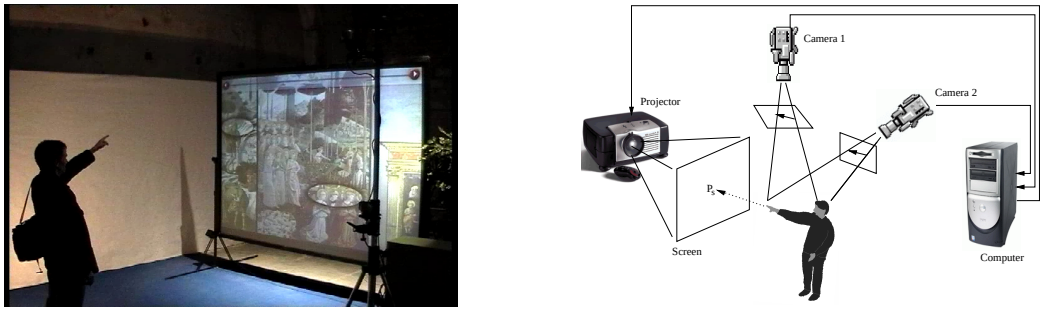


Fig. 1. An application of vision-based hand pointing. *Left*: Interaction scenario. *Right*: The main interaction components.

whose "drags" and "clicks" reflect respectively changes and fixations of interest as communicated by the user through natural hand pointing actions.

B. Modeling interaction

1) The visual geometry of hand pointing: Here we introduce the main geometric aspects of camera projection and hand pointing, and discuss a geometric approach to screen point remapping, i.e. to the computation of the screen position of interest. For the sake of generality, we will refer to a configuration with K cameras monitoring the user. The next sections will explain how to compute the model parameters involved, and provide details on the actual stereo implementation adopted for the experiments.

Let Π_s be the screen plane, and consider a point P_s of interest in it (see Fig. 2). We can regard the action of pointing to P_s as generating an ideal 3D *pointing line* L whose intersection with the screen is the interest point:

$$P_s = L \cap \Pi_s \ . \tag{1}$$

The pointing line concept is a useful abstraction allowing the definition of a geometric interaction model regardless of the actual peculiarities of physical user pointing. Consider now the generic camera $\mathcal{C}_k$, $k = 1 \dots K$, and assume to be able to trace at any time the projection of L onto the image plane Π_i; such a line is referred to as *image line* and denoted as l_i. The image line can be regarded as the intersection of the *projection plane* Π (i.e., the plane from the optical camera center through L) with the image plane:

$$l_i = \Pi \cap \Pi_i \ . \tag{2}$$

If the camera internal parameters are known, it is possible to "invert" the equation above, and construct Π from l_i. By definition, the projection plane must contain the interest point; more specifically, such a point must lie on the so called *screen line* l_s resulting from the intersection of the projection plane with the screen plane:

$$l_s = \Pi \cap \Pi_s \ . \tag{3}$$

The condition $P_s \in l_s$ can be expressed in homogeneous coordinates (where both points and lines are represented as 3-vectors) by a linear constraint of the form

$$\boldsymbol{l}_s^T \boldsymbol{P}_s = 0 \ , \tag{4}$$

referred to as *screen line constraint*. Moreover, by inspection of Fig. 2 it is clear that planes Π_s and Π_i correspond under a plane projective transformation, or planar homography [23], i.e. a 3×3 homogeneous matrix H transforming points as $\boldsymbol{P}_i = \mathrm{H}\boldsymbol{P}_s$ and lines as $\boldsymbol{l}_i = \mathrm{H}^{-T}\boldsymbol{l}_s$. Thus the screen line can alternatively be expressed as $\boldsymbol{l}_s^T = \boldsymbol{l}_i^T\mathrm{H}$, i.e. as the back-projection of the image line through the homography, and the screen line constraint rewritten in the form

$$\boldsymbol{l}_i^T \mathrm{H} \boldsymbol{P}_s = 0 \ . \tag{5}$$

The equation above can be regarded as the basic theoretical tool for our hand pointing system: it both provides a clear geometrical interpretation of the role played by the different interaction components (user, camera, screen), and formulates the screen line constraint into an immediately implementable way. Specifically, the user-dependent element in eq. 5 is the image line l_i, whose parameters reflect the direction of the pointing line L, thus encoding the current user's interest. The image line must be estimated from image data and continuously updated as the result of the tracking of user action.

Before being used for screen point *remapping*, i.e. to determine the screen location of interest, the constraint of eq. 5 must be used for map *calibration*, i.e. to compute the projection homography H, whose entries encode both intrisic and extrinsic camera parameters. Calibration is performed by recording image line parameters while pointing at screen points with known coordinates (see also §II-B.2 and §II-C.1). At remapping time, the projection homography H is used in eq. 5 together with the current image line parameters to determine, for each camera, a linear constraint on the two coordinates of P_s. The screen location of interest is then obtained by intersecting the screen line constraints from $K \geq 2$ cameras (explicitly, P_s is estimated as the vector product $\boldsymbol{l}_s^{\mathrm{L}} \times \boldsymbol{l}_s^{\mathrm{R}}$ using a LR stereo pair, and as the least squares pseudo-intersection of constraint lines if more than two cameras are used).

2) Derivation of the screen line constraint: We develop explicitly here the screen line constraint of eq. 5 for the general case of perspective camera, and also discuss (form, conditions of validity) its specialization to the affine camera model. Let us consider the image line l_i of equation $\alpha u + \beta v + \gamma = 0$, whose parameters α, β and γ are measured in the image plane uv. If the intrinsic camera parameters k_u, k_v, s, u_0 and v_0 are known [23], then the line l_i can be back-projected onto the

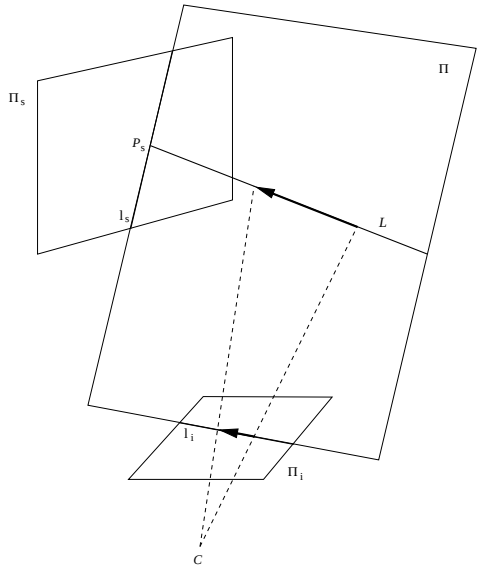


Fig. 2. The geometry of hand pointing and camera projection.

projection plane Π through the pinhole camera model

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{Z_c} \begin{bmatrix} k_u & s & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \ , \qquad (6)$$

where $[X_c\, Y_c\, Z_c]^T$ is the camera coordinate representation of a generic point $P \in \Pi$. The projection plane equation is

$$\alpha' X_c + \beta' Y_c + \gamma' Z_c = 0 \ , \qquad (7)$$

where the plane coefficients (camera coordinates) evaluate as

$$\begin{bmatrix} \alpha' & \beta' & \gamma' \end{bmatrix} = \begin{bmatrix} \alpha & \beta & \gamma \end{bmatrix} \begin{bmatrix} k_u & s & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \ . \qquad (8)$$

By applying the change of frame mapping to eq. 7, we obtain a different expression of the projection plane in terms of world coordinates $[X\, Y\, Z]^T$ and extrinsic camera parameters:

$$\begin{bmatrix} \alpha' & \beta' & \gamma' \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = 0 \ , \qquad (9)$$

where the r_{ij}'s are the entries of the rotation matrix, and the t_k's are those of the relative translation of the two frame origins. The change of frame mapping can also be applied to eq. 6, to derive an expression relating a 3D point (world coordinates) with its image projection (pixel coordinates):

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{Z_c} \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \ , \qquad (10)$$

where the q_{ij}'s are functions of both the extrinsic and intrinsic camera parameters. From a comparison of eqs. 8 through 10 it emerges that

$$\begin{bmatrix} \alpha & \beta & \gamma \end{bmatrix} \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = 0 \ , \qquad (11)$$

which expresses the projection plane in terms of image line parameters and twelve projection parameters (of which only eleven are independent), for varying P's. Since $r_{31}X + r_{32}Y + r_{33}Z + t_3 = Z_c$, eq. 11 rewrites as

$$\begin{bmatrix} \alpha & \beta & \gamma \end{bmatrix} \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ 0 & 0 & 0 & Z_c \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = 0 \ . \qquad (12)$$

It is important to remark that, by construction, the above expression holds at any 3D point, and even at P's which are away from the visual field of the camera, and are not directly visible. Such a *virtual projection* results from considering a geometrically infinite image plane in the place of the physical, photosensitive plane of the imaging sensor. In the special case when P belongs both to the projection plane and to the screen plane $Z = 0$, an explicit expression for the screen line constraint of eq. 5 is obtained:

$$\begin{bmatrix} \alpha & \beta & \gamma \end{bmatrix} \begin{bmatrix} q_{11} & q_{12} & q_{14} \\ q_{21} & q_{22} & q_{24} \\ r_{31} & r_{32} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = 0 \ . \qquad (13)$$

The screen line constraint can be regarded as a function of camera parameters mapping an image line onto its corresponding screen line. To calibrate the map, its eight projection parameters (i.e., the independent entries of the planar homography H) have to be estimated from a minimum of eight known screen points—see § II-C.1. However, the number of model parameters can be reduced if a proper camera placement is chosen, allowing to linearize the projection map (affine camera model). In fact, from eq. 12 it emerges that, if the screen point depth Z_c can be assumed equal to a constant $\overline{Z}_c$, then the full perspective constraint of eq. 13 can be rewritten so as to depend on six parameters (and hence on six calibration points) only:

$$\begin{bmatrix} \alpha & \beta & \gamma \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{14} \\ a_{21} & a_{22} & a_{24} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = 0 \ , \qquad (14)$$

being $a_{ij} = q_{ij}/\overline{Z}_c$. The validity of this simplified model (affine version of the screen line constraint) is restricted to the case when $|t_3| \gg |r_{31}X + r_{32}Y|$, i.e. when the screen size is small w.r.t. to its average distance from the camera, and/or when the image and screen planes are nearly parallel. If camera layout is such that neither of these conditions is met, the full perspective screen line constraint of eq. 13 should be used, since normalizing eq. 12 by Z_c would lead to projection matrix entries which are not constant over the screen plane Π_s.

3) Properties: The interaction model described above has some peculiar features that make our approach different from other approaches proposed recently [19], [20], [21]. First, our model does not set any conditions on camera placement: for example, it does not constrain the flat surface pointed to by the user to be visible by the cameras, as prescribed in [19]: the only condition on camera placement is that head and pointing hand be always visible by at least two cameras. Another important difference w.r.t. the method proposed in [19] is that our model does not work only under the assumption of affine projection, but also covers the full perspective case. Of course, whenever possible, the affine version of the model should be used, as the number of parameters is lower and each parameter can be estimated more reliably than in the full perspective case, provided that camera parameters are such that perspective effects are negligible. A second point concerns the introduction of the projection line concept. It allows to define a theoretical model which is actually decoupled from the measurement model, i.e. the way the ideal projection line is estimated. It also allows to employ line-based triangulation, which is more effective than point-based triangulation in order to guarantee model invariance with respect to user position. With our solution, after calibration, the user can move freely in the environment while pointing: this is not allowed by most of the current hand pointing systems, as the one presented in [20]. It should also be noted that the 3D pointing line is not computed explicitly as in [21], but it is used only implicitly in order to carry out computations at a purely appearance-based level, hence with the most reliable estimates.

C. Image analysis

Here we discuss the image analysis algorithms used to monitor hand pointing actions and compute in real-time the image line parameters required for both back-projection map calibration and screen point remapping.

1) Symbolic stereo: The current system implementation is based on two cameras whose corresponding screen lines are intersected to obtain the screen point P currently pointed to. A grid of N points regularly sampled on the screen are used to calibrate simultaneously (but separately) the two cameras via the least squares solution, by singular value decomposition, of two overdetermined homogeneous linear systems. Explicitly, if the full perspective model is adopted, the basic screen line constraint of eq. 13 can be rearranged as

$$\left[\begin{array}{cccccccccc} \alpha X & \alpha Y & \alpha & \beta X & \beta Y & \beta & \gamma X & \gamma Y & \gamma \end{array}\right] \boldsymbol{h} = 0 \ , \quad (15)$$

where $\boldsymbol{h} = [q_{11}\ q_{12}\ q_{14}\ q_{21}\ q_{22}\ q_{24}\ r_{31}\ r_{32}\ t_3]^T$ is the unknown homogeneous 9-vector (eight independent parameters) of planar homography entries to be computed from $N \geq 8$ line observations $(\alpha_i, \beta_i, \gamma_i)$ and corresponding screen points (X_i, Y_i), $i = 1 \ldots N$. For calibration in the affine case, an expression similar to eq. 15 can be derived from eq. 14, requiring the estimation of six unknowns from a minimum of six screen points and line observations.

Each image line l_i is simply measured as the line passing through the image of head centroid and hand tip (see Fig. 3). While suggested by robustness considerations (the evaluation of finger direction is known to be less reliable [21]), the choice of this simple bilocal strategy for image line computation implies the adoption of a specific measurement model of the pointing line L, whose validity was assessed experimentally in terms of pointing performance and maximum remapping error (see section III). At remapping time, image lines are estimated independently and then associated to the same physical entity (the pointing line L) to perform stereo triangulation by screen line intersection: this fast computational scheme is referred to as *symbolic stereo*, since it does not require to compute pixel-by-pixel correspondences, but simply to label separately in each image two fiducial "hand" and "head" points, and then to let points with identical labels to correspond each other.

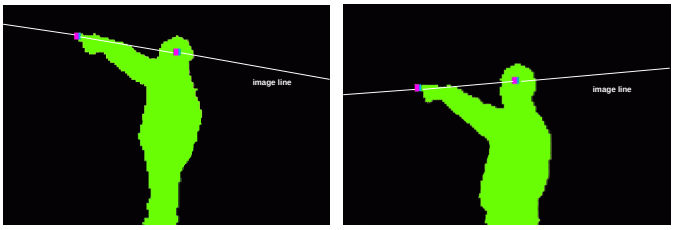


Fig. 3. Bilocal image line tracing in the stereo pair through head and hand localization. Black pixels indicate background.

2) Early vision: In order to locate both the hand and the head in each pair of images, color and motion visual data are processed and user tracking is performed.

The first image processing step is the extraction of the foreground region (i.e., the pixels where the user is imaged) from the background. The system first acquires the background from an empty scene (i.e., with no user). To determine whether or not a pixel is part of a foreground region, a measure of the departure from the corresponding pixel in the background model is computed, and compared to a threshold. In order to achieve good results in a real environment, each color component of background pixels is modeled according to a time-varying second order statistics [10], whose mean and variance parameters are continuously updated, to compensate for temporal changes in illumination. Spatial changes in illumination are also addressed, during background subtraction, by taking into account two different color spaces, i.e. the standard RGB space and the brightness normalized rgb space $(r, g, b) = (R/Y,\ G/Y,\ B/Y)$, where $Y = R + G + B$ and $r + g + b = 1$. A pixel p is classified as foreground if its color distance $d_{\text{FG}}(p, b)$ from the corresponding background pixel b is over a certain threshold ϑ_{FG} either in the standard RGB space *or* in the normalized ($\star$) one:

$$d_{\text{FG}}(p, b) > \vartheta_{\text{FG}} \ \vee \ d^{\star}_{\text{FG}}(p, b) > \vartheta^{\star}_{\text{FG}} \ , \quad (16)$$

where

$$d_{\text{FG}}(p, b) = \max\left\{ \frac{|Y_p - Y_b|}{\sigma^2(Y_b)}, \frac{|R_p - R_b|}{\sigma^2(R_b)}, \frac{|B_p - B_b|}{\sigma^2(B_b)} \right\} \ , \quad (17)$$

$\sigma^2(x)$ standing for the variance of x. An original feature of this algorithm is that, for both color spaces, the classification threshold is time-varying, being computed at each frame as the value providing a fixed percentage of isolated pixels misclassified as foreground pixels due to CCD camera noise.

This method is remarkably useful to obtain a time-varying discrimination sensitivity and compensate for even strong changes in illumination and camera parameters, thus achieving adaptation to environmental changes in the low level vision algorithms.

Once extracted, the raw foreground is topologically filtered to obtain connected blobs and fixing small holes. The blobs are then analyzed so as to extract the tip of the pointing hand and the head centroid. Whenever possible, blob analysis is carried out by the weighted combination of two different low level processes, namely *skin color analysis* and *body silhouette analysis*. The latter process is generally more accurate and less dependent on user clothing and background characteristics than the former; yet, it is also more dependent on specific camera layout, and even not usable at all if the user is imaged in such a way that the pointing hand or the head are occluded.

Concerning skin color analysis, the distance function used to check if a foreground pixel f belongs to the skin color class s is

$$d_{\mathrm{CC}}(f,s) = \left[\frac{Y_f - Y_s}{\sigma^2(Y_s)}\right]^2 + \left[\frac{R_f - R_s}{\sigma^2(R_s)}\right]^2 + \left[\frac{B_f - B_s}{\sigma^2(B_s)}\right]^2 \quad . \tag{18}$$

Differently from foreground extraction, a pixel is classified as skin if both distances in the standard color space *and* in the normalized one are below two appropriate thresholds:

$$d_{\mathrm{CC}}(f,s) < \vartheta_{\mathrm{CC}} \ \wedge \ d^{\star}_{\mathrm{CC}}(f,s) < \vartheta^{\star}_{\mathrm{CC}} \ . \tag{19}$$

After color-based segmentation, the head and the pointing hand are located through heuristic considerations on the shape and position of skin regions.

Body silhouette analysis is carried out by considering the overall foreground region. The location of user's head and hand are estimated by following simple heuristics. Explicitly, provided that the camera is located so as to get a side view of the user, the head is easily located at the top of the silhouette, and the pointing hand as the tip of the arm, the latter identified as the dominant protrusion of the silhouette.

As time goes by, current head and hand measurements are combined with previous ones using temporal low-pass filters, both to reduce noise and smoothen the tracking behavior; a constant velocity predictive filter is also used, with beneficial effects on tracking speed and on the management of critical tracking situations, such as the detection of and the recovery from occlusions (hand-hand or hand-face).

D. User-adapted interaction

Here we address the problem of adaptation to user behavior and characteristics. The purpose of adaptation is to improve the quality of pointing, and let the user forget as much as possible that his interaction with the environment is mediated by a computer vision system that interprets his actions.

1) Adaptation at the system input level: In our approach, pointing errors can arise for three main reasons: (1) use of wrong projection models for the screen line constraint; (2) bad approximation of the ideal pointing line from image measurements; (3) pointing style and/or user characteristics at remapping time differ from those at calibration time.

The first two errors are hard to compensate for at remapping time, but they can be possibly recovered by using a different projection model (in the first case) or a different measurement model (in the second case). For instance, if the affine projection constraint of eq. 14 does not provide good results, it is possible that the current camera layout be partially incompatible with it, hence suggesting to switch to the full perspective constraint of eq. 13. Analogously, if image line measurements reconstruct only partially the geometry of the pointing line, residual errors have to be expected even after accurate calibration session: these errors can be lowered by using more clever ways to compute the image line.

The error due to the discrepancies between user's behavior and characteristics at remapping and calibration time arises either because a user different from the *calibrator* (i.e., the person who performed calibration) is currently interacting with the system, or because the calibrator is acting differently from calibration time. These discrepancies can be reduced by self-calibration at remapping time. Self-calibration is performed at selection ("pointer click") time, and can be implemented in two different ways: (1) by *accumulation*, (2) by *substitution*. In the first case, the image line parameters computed at selection time are recorded together with the selected screen point (typically, the center of a clickable screen region, such as an interface button) to be used for all future recomputations of calibration parameters. The second case is analogous, except for the fact that each new selected point and corresponding image line parameters replace the previous observations associated to the same point. Self-calibration by substitution is expected to be more rapid than the accumulation method in adapting the calibration parameters; however, being less conservative, this method is more sensitive than the other to the effects of inaccurate image measurements.

The main difference between off-line calibration and self-calibration is that in the latter case the user is completely unaware that the system is performing an adjustment of its internal parameters: he simply keeps interacting with the system, perhaps with increasing psychological satisfaction due to the increasing accordance of system response to his will. Self-calibration can also recover from inaccuracies due to a poor off-line calibration, thus letting the calibrator perceive an improvement in the quality of interaction.

2) The role of feedback and user action: To improve the overall remapping performance, it is extremely important that the user be provided with a visual feedback of his own pointing action. Different kinds of feedback can be presented to the user at the interface level, according to the specific application requirements. A basic distinction is between continuous vs discrete feedback, the former being represented by an on-screen icon that informs at any time the user about the current position being pointed to, in the same way as with conventional mouse-based interfaces. Discrete feedback is instead related to selection actions only, and can be implemented either after a selection has been made (click mode) or some instants before a selection takes place (pre-click mode).

In the presence of continuous feedback, if the pointing error is small enough to let the user forget about the system and concentrate uniquely on his task, then remapping errors are

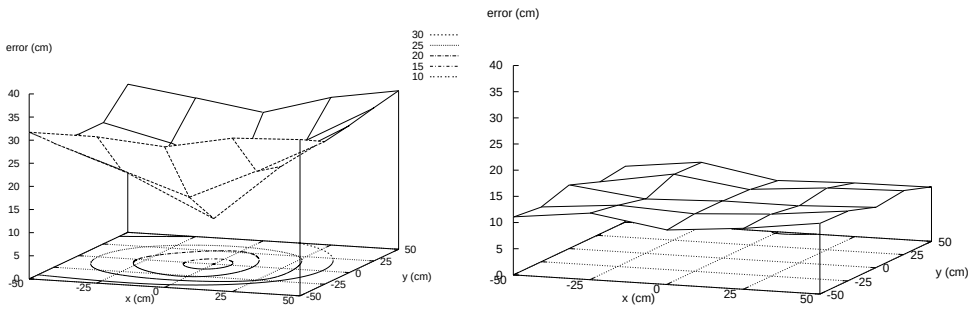


Fig. 4. Remapping error at floor position Q_j, $j = 1 \dots 25$. *Left*: Calibration from a single floor position. *Right*: Calibration from multiple floor positions.

compensated for directly by the user by slight, unconscious adjustments of the pointing arm. Discrete feedback can play instead an important role during self-calibration. In particular, the pre-click feedback mode can be used to recover from wrong pointing caused by very poor off-line calibration: as a pre-click signals the risk of a wrong selection, the user can slightly (and, again, unconsciously) adjust his pointing action until obtaining a correct selection.

III. System Evaluation

A. Equipment and experimental setup

PointAt was implemented in C++ and runs in real-time (50 Hz) on a Pentium PC 600 MHz running Windows; two USB webcam devices at a resolution of 160×120 are used for image acquisition. To be compliant with a large number of application scenarios, the hardware platform of the system was deliberately chosen as the most standard and low cost possible. The hardware/software structure of the system is organized into three different layers (physical: image acquisition and display, logical: image analysis and graphic synthesis, application: human-computer interfacing). Updated information about new *PointAt* versions, videos and running applications are available at `http://viplab.dsi.unifi.it/hci/PointAt`.

The experimental setup includes a wall screen with a maximum size of 5 m $\times$ 4 m, pointed to from an average distance of about three meters. Pointer clicking actions are issued after a temporal persistence of half a second. In the following experiments, the relative position of cameras and user is such as it allows hand/head localization by body silhouette analysis. In particular, user standing positions on the floor are included in a square of 2.5 m $\times$ 2.5 m.

B. Experiments

Several tests were performed so as to assess both system accuracy. The tests were carried out by eleven volunteers, with different physical characteristics and pointing styles, alternating during interaction. Also, to gain an insight into system robustness w.r.t. real conditions of use, three different interaction configurations (size of screen, layout of cameras) were tested. Experimental results report on system accuracy expressed in terms of remapping and calibration errors measured at generic screen points (i.e., not necessarily those used for calibration). The experiments address, in order: *(i)* the dependency of remapping errors on the position of the user on the floor; *(ii)* the intrinsic error of the model in terms of residual calibration error; *(iii)* the influence of user's pointing style and physical characteristics, and how this influence is reduced by user adaptation; *(iv)* to which extent performance is affected by camera layout and projection model.

a) : To estimate the dependency of remapping errors on the position of the user on the floor, the screen space was discretized into a 5×4 point grid, and the floor space into a 5×5 position grid. The *local remapping error* δ_{ij} is defined as the Euclidean distance, in the screen plane, between the screen position P_i (ground truth, $i = 1 \dots p$) and the remapped point $\widehat{P}_i$, for a user standing at floor position Q_j, $j = 1 \dots q$. The *remapping error at floor position* Q_j, $\mathcal{F}(Q_j) = (1/p) \sum_{i=1}^{p} \delta_{ij}$, is the average of all local remapping errors during pointing actions from position Q_j. The *overall remapping error* is defined as $\mathcal{E} = (1/pq) \sum_{i=1}^{p} \sum_{j=1}^{q} \delta_{ij}$.

TABLE I

Residual error at calibration points.

Calibration floor positions	$\mathcal{C}$, cm	$\mathcal{C}$, deg
single	2.8	0.48
multiple	3.8	0.66

TABLE II

Overall remapping error for different test conditions.

Testing subjects	*Adapt*	$\sigma_{95\%}$, cm	$\mathcal{E}$, cm	$\mathcal{E}$, deg	*Wrong*, %
A, right hand	No	18.2	11.7	2.02	2
A, left hand	No	22.6	14.9	2.58	9
as tall as A	No	23.0	15.3	2.65	11
different from A	No	32.1	18.1	3.09	24
different from A	Yes	19.5	12.4	2.15	3

Fig. 4 (left) reports the remapping error $\mathcal{F}(Q_j)$ at all floor grid positions, for a calibration carried out standing at a single floor position (the center of the floor grid) and using all the 20 points of the screen grid. The reduced, affine camera projection matrix of eq. 14 is used. The configuration for this experiment is: screen size of 2.9 m $\times$ 2.2 m, cameras located at different heights from the floor, both pointing the user's side and with optical axes in a plane parallel to the screen. In the figure, the user stands at position $(0, 0)$, while cameras are located at his left at $x = -100$ cm. As the user moves away

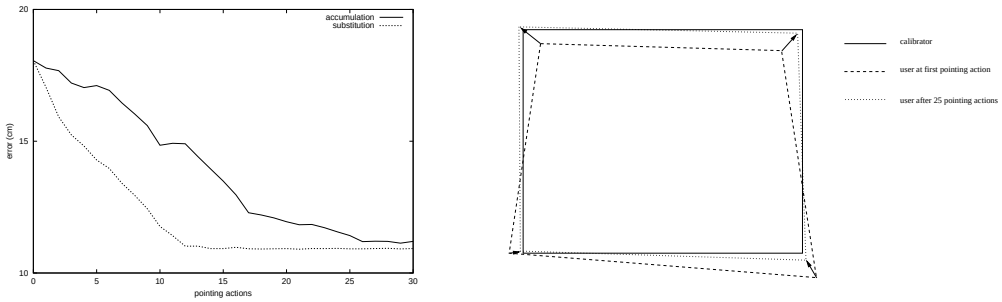


Fig. 5. *Left*: Adaptation of calibration parameters as a function of the number of pointing actions. *Right*: Remapping improvement during adaptation (screen space).

from the unique calibration point, the remapping accuracy decreases: the decrease rate is higher when the user moves away from the cameras. This can be explained by observing that image line measurement (and hence triangulation) is less robust as the user becomes smaller in the image. Fig. 4 (right) reports the remapping error at all floor grid positions, for a calibration carried out standing at 5 different floor positions and pointing to 4 different screen points (for a total of 20 pointing actions, as above). In this case, the dependency of calibration parameters from floor position is greatly reduced. The average remapping error is equal to 11.7 cm.

b) : The average remapping error is due to incorrect modeling, tracking and calibration inaccuracies, as well as on camera-user relative position and environmental conditions. As a way to evaluate the influence of modeling on the overall error, the residual calibration error $\mathcal{C}$ is computed by substituting back the estimated calibration parameters in the quadratic function minimized by least squares. The residual calibration error is reported in Table I, both in cm and in degrees, for the cases of calibration from a single floor position and from multiple floor positions. It can be observed that, despite the fact that calibrating from multiple positions is beneficial for remapping, for a fixed number of overall calibration points single position calibration has a lower residual error, as it relies on a greater number of reference screen points.

c) : Table II reports the results of several tests the purpose of which is to assess the dependency of remapping performance on differences in pointing style and physical characteristics between the user at remapping time and the calibrator. Remapping performance is provided in terms of overall remapping error, 95% standard deviation and percentage of screen point equivocations. Each line corresponds to a test performed by a different testing subject: for all the tests the same calibration map, relative to a right-handed calibrator, was used. Not surprisingly, the best performance ($\mathcal{E} = 11.7$ cm corresponding to 2.02 deg, with $\sigma_{95\%} = 18.2$ cm and 2 % of equivocation) is achieved when the user is the same person who acted as calibrator and performs pointing with his right hand. Performance slightly degrades when the calibrator switches the pointing hand, and when the user is a person with a different pointing style but the same height as the calibrator. The same performance as the best case is obtained when the user is a different person from the calibrator and self-calibration is introduced ($\mathcal{E} = 12.4$ cm corresponding to 2.15 deg, with $\sigma_{95\%} = 19.5$ cm and 3 % of equivocation). As a matter of fact, *self-calibration totally removes the need of performing an off-line calibration session for each new user*, being it sufficient that a raw calibration map be already available at remapping time.

TABLE III

SYSTEM ACCURACY FOR TWO DIFFERENT CAMERA LAYOUTS.

LAYOUT	CALIBRATION		REMAPPING			
	$\mathcal{C}$, cm	$\mathcal{C}$, deg	$\sigma_{95\%}$, cm	$\mathcal{E}$, cm	$\mathcal{E}$, deg	*Wrong*, %
"lateral"	3.8	0.66	18.2	11.7	2.02	2
"front/rear"	3.9	0.68	19.6	12.2	2.11	3

TABLE IV

AFFINE *vs* PERSPECTIVE PROJECTION MODEL: SYSTEM ACCURACY.

MODEL	CALIBRATION		REMAPPING			
	$\mathcal{C}$, cm	$\mathcal{C}$, deg	$\sigma_{95\%}$, cm	$\mathcal{E}$, cm	$\mathcal{E}$, deg	*Wrong*, %
affine	6.2	1.08	23.9	18.6	3.23	5
perspective	2.5	0.44	16.9	11.5	2.00	1

Fig. 5 (left) shows the improvement of remapping performance that is induced by adaptation as the number of pointing actions increases. Both the solutions based on the accumulation and substitution mechanisms are presented. As foreseen earlier, substitution is faster than accumulation to reach the asymptotic remapping error below which adaptation only cannot go. Fig. 5 (right) provides a qualitative insight into the adaptation mechanism. The figure shows, connected by a solid line, the four corners of the screen grid as remapped by the calibrator: the same corners as reconstructed by a user different from the calibrator at two distinct times are shown with dashed lines. The effect of adaptation is to make the appearance of the user grid more and more similar to the reference, calibrator grid; the arrows indicate the trajectory and speed of each remapped point in the screen plane.

d) : To test the dependency of system accuracy on camera layout, the results obtained with the "lateral" layout of the previous experiment are compared here with those obtained with an alternative "front/rear" layout featuring the two cameras located both on the left side of the user, but at the same height from the floor. Specifically, the left camera lies between the screen and the user, of whom takes a side-frontal view; the right camera has instead a side-rear viewpoint, being

Fig. 6. *Left*: The image stripe summarizing the whole fresco of the "Cappella dei Magi" at Palazzo Medici-Riccardi, Florence. *Right*: The chapel's ground-plan: each wall contains a part of the fresco.

closer to the user than to the screen. Like in the previous experiments, the affine projection model is used. Table III shows the values of the calibration error (multiple floor positions) and overall remapping error (testing subject: right-handed calibrator, no adaptation). Despite the large difference between the two camera layouts, the results are very similar in the two cases; this lets us conclude that, as long as the conditions for using the affine projection model are met with both the left and right cameras, the system is largely invariant w.r.t. changes of camera layout. Of course, the margins of invariance are even stronger when the more general perspective model is used.

To investigate the accuracy degradation when the relative position of camera and screen is such that at least one of the cameras cannot be successfully modeled as affine, we refer to a third configuration, featuring a screen larger (4.9 m $\times$ 3.7 m) than before, and the same "lateral" camera layout used in the previous experiments. Table IV shows the values of the calibration error (multiple floor positions) and the overall remapping error (testing subject: right-handed calibrator, no adaptation) obtained with this new configuration using the affine and perspective projection models, respectively. From a comparison of the results, it is evident that, due to screen enlargement, the affinity conditions were not met in the screen periphery, with a consequent loss of system accuracy. In particular, remapping accuracy with the affine model (3.23 deg, with 5% of equivocation) is not only worse than the one obtained with the perspective model (2.00 deg, with 1% of equivocation), but is also well below the one obtained previously with the affine model and a smaller screen (2.02 deg, with 2% of equivocation—see again Table III). The latter accuracy value, being very close to the one obtained under strong perspective, tells us that with the smaller screen the affine and perspective homographies are almost equal; hence, in that case the affinity assumption is fully verified. As a rule of thumb, considering a setup with a 3 meters wide screen placed at 3 meters from the user, in order to obtain a 95% success on recognition of pointing actions, the distance on the screen between two neighbouring clickable locations should not be lower than 40 cm in the case of use by the general untrained and uncalibrated public. This rule was followed in the development of the application scenario described below.

IV. Application Scenario

The *PointAt* hand pointing system discussed in this paper was recently embedded into a prototype system for human-computer interaction in a museum. The system is currently being installed in the premises of the museum of Palazzo Medici Riccardi, Florence, Italy, with the purpose of providing visitors with a new way to explore Renaissance masterworks. Using *PointAt*, visitors can explore and interact with the digital reproduction of the famous fresco "La cappella dei Magi," by Benozzo Gozzoli (c. 1421–1497), whose original can be visited in another room of the palace. This is intended to prepare people interested to visit the real chapel, by providing them with information about the human figures, animals, tissues etc. represented in the fresco. The original fresco was digitally acquired so as to obtain a continuous "stripe" covering the content of each of the different walls of the chapel (see Fig. 6). Since only a part of the fresco is displayed on the screen, the user can shift the pictorial content left or right by pointing at the scroll buttons (Fig. 7, left), thus simulating a physical tour inside the real chapel. "Clickable" regions, i.e. regions which can be selected after a persistent (1.2 s) pointing action are highlighted, as shown in Fig. 7 (right). Selection of one of these regions is interpreted by the system as a sign of specific interest into a visual particular of the fresco; the system then responds to a selection action with both a visual and acoustic output. Visual output consists of an enlarged view of the selected area, accompanied by a text caption display illustrating the names of the people portrayed therein; acoustic output provides a more comprehensive explaination of the area of interest.

Fig. 7. Example of display for interactive fruition of large size art images in the Palazzo Medici Riccardi. *Left*: A portion of the fresco and its selectable regions. *Right*: Hypertextual information displayed in response to a selection.

V. Conclusions and Future Work

We have discussed the design and evaluation of the *PointAt* system for vision-based hand pointing in an advanced human-computer interaction scenario. The system works in real time on a low-cost hardware platform, is fairly accurate and independent of user characteristics and position, camera layout, and environmental changes. *PointAt* was recently embedded

in a larger system to support interaction in the context of an augmented museum, and is currently being adapted to 2D and 3D educational and entertainment interfaces. Future work will be especially devoted to extend system operation to the management of several simultaneous users. To this end, more sophisticated tracking algorithms capable of dealing with severe occlusion conditions will be developed, allowing two or more users to share the same interaction space and also to interact together through a computer support based on natural gestures.

ACKNOWLEDGMENT

This work was partially funded by Provincia di Firenze under contract no. 83/02 in the context of the project "Laboratorio di Lorenzo." The authors wish to heartily thank Dr. Gabriele Baggiani and Dr. Grazia Pietrasanta for their valuable help in the preparation of this paper.

REFERENCES

[1] R.J.K. Jacob, "User interfaces," in *Encyclopedia of Computer Science*, A. Ralston, E.D. Reilly, and D. Hemmendinger, Eds. Grove Dictionaries Inc., fourth edition, 2000.

[2] J. Nielsen, "Noncommand user interfaces," *Communications of the ACM*, vol. 36, no. 4, pp. 83–99, April 1993.

[3] R.J.K. Jacob, "Human-computer interaction: Input devices," *ACM Computing Surveys*, vol. 28, no. 1, pp. 177–179, March 1996.

[4] B.A. Myers, "A brief history of human-computer interaction technology," *Interactions*, vol. 5, no. 2, pp. 44–54, 1998.

[5] A. Blake and M. Isard, *Active Contours: the application of techniques from graphics, vision, control theory and statistics to visual tracking of shapes in motion*, Springer-Verlag, 1998.

[6] B. Carlson, "The consumer desktop looks at vision: R&D directions at Microsoft and MIT," *Advanced Imaging*, pp. 26–28, 1998.

[7] A.P. Pentland, "Smart rooms," *Scientific American*, vol. 274, no. 4, pp. 68–76, 1996.

[8] V.I. Pavlovic, R. Sharma, and T.S. Huang, "Visual interpretation of hand gestures for human-computer interaction: A review," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 677–695, 1997.

[9] I.A. Essa and A.P. Pentland, "Coding analysis, interpretation, and recognition of facial expressions," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 757–763, 1997.

[10] C.R. Wren, A. Azarbayejani, T. Darrell, and A.P. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, July 1997.

[11] S. Iwasawa, J. Ohya, K. Takahashi, T. Sakaguchi, S. Kawato, K. Ebihara, and S. Morishima, "Real-time, 3D estimation of human body postures from trinocular images," in *IEEE International Workshop on Modelling People*, Corfu, Greece, September 1999, IEEE Computer Society, pp. 3–10.

[12] J.L. Crowley, J. Coutaz, and F. Bérard, "Things that see," *Communication of the ACM*, vol. 43, no. 3, pp. 54–64, March 2000.

[13] C. Colombo, A. Del Bimbo, and S. De Magistris, "Interfacing through visual pointers," in *Computer Vision for human-machine interaction*, chapter 8, pp. 135–153. Cambridge University Press, 1998.

[14] G. Butterworth and L. Grover, "Joint visual attention, manual pointing, and preverbal communication in human infancy," in *Attention and Performance XIII*, M. Jeannerod, Ed., pp. 109–127. Lawrence Erlbaum, 1990.

[15] J.L. Sibert and M. Gokturk, "A finger-mounted, direct pointing device for mobile computing," in *Proceedings of the ACM Symposium on User Interface Software and Technology*, 1997, Picking and Pointing, pp. 41–42.

[16] P. Wellner, "Interacting with paper on the DigitalDesk," *Communications of the ACM*, vol. 36, no. 7, pp. 86–96, 1993.

[17] C. Maggioni and B. Kämmerer, "GestureComputer – history, design and applications," in *Computer Vision for human-machine interaction*, chapter 2, pp. 23–51. Cambridge University Press, 1998.

[18] J.L. Crowley, "Vision for man-machine interaction," *Robotics and Autonomous Systems*, vol. 19, no. 3–4, pp. 347–358, March 1997.

[19] R. Cipolla and N.J. Hollinghurst, "Visually guided grasping in unstructured environments," *Robotics and Autonomous Systems*, vol. 19, no. 3–4, pp. 337–346, 1997.

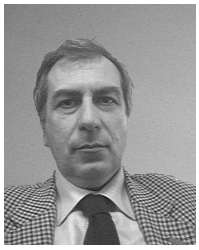
[20] K. Mase, "Human reader: A vision-based man-machine interface," in *Computer Vision for human-machine interaction*, chapter 3, pp. 53–81. Cambridge University Press, 1998.

[21] Y.-P. Hung, Y.-S. Yang, Y.-S. Chen, I.-B. Hsieh, and C.-S. Fuh, "Free-hand pointer by use of an active stereo vision system," in *Proceedings of the 14th International Conference on Pattern Recognition, Brisbane, Australia*. 1998, pp. II:1244–1246, IEEE Press.

[22] G. Baggiani, C. Colombo, and A. Del Bimbo, "Advanced man-machine interface for cultural heritage," in *International Conference on Image Processing ICIP 2001, Thessaloniki, Greece*, October 2001, pp. 582–585.

[23] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, 2000.

Carlo Colombo Carlo Colombo was born in Bari, Italy, on March 3, 1966. He holds an MS in Electronic Engineering (1992) from the University of Florence, Italy, and a PhD in Robotics (1996) from the Sant'Anna School of University Studies and Doctoral Research, Pisa, Italy. From 1996 to 1999 he was an assistant professor at the Department of Electronics for Automation, University of Brescia, Italy. He is currently an associate professor at the Department of Systems and Informatics of the University of Florence. His main research interests include Computer Vision and its applications to Human-Machine Interaction, Advanced Robotics and Multimedia Systems. From 1996 to 2000 he was the secretary of the Italian Chapter of the International Association for Pattern Recognition. Since 2001 he is an editorial board member of the international journal *Robotics and Autonomous Systems* (Elsevier).



Alberto Del Bimbo Alberto Del Bimbo was born in Florence, Italy, on February 15, 1952. He is full professor of Computer Engineering at the University of Florence, Italy. Since 1998 he is the director of the Master in Multimedia of the University of Florence. At the present time, he is Deputy Rector of the University of Florence, in charge of Research and Innovation Transfer. His scientific interests are Pattern Recognition, Image Databases, Multimedia and Human-Computer Interaction. He has delved into object recognition and image sequence analysis, multimedia databases and content based retrieval. Prof. Del Bimbo is the author of over 170 publications in the most distinguished international journals and conference proceedings. He is the author of the "Visual Information Retrieval" monography on content-based retrieval from image and video databases. He is Member of IEEE (Institute of Electrical and Electronic Engineers) and IAPR (International Association for Pattern Recognition). Prof. Del Bimbo was the President of the IAPR Italian Chapter, from 1996 to 2000 and Member at large of the IEEE Publication Board from 1998 to 2000. He is presently Associate Editor of Pattern Recognition, Journal of Visual Languages and Computing, Multimedia Tools and Applications Journal, Pattern Analysis and Applications, IEEE Transactions on Multimedia, and IEEE Transactions on Pattern Analysis and Machine Intelligence. He was the Guest Editor of several special issues on image databases and image analysis.

Alessandro Valli Alessandro Valli was born in Pisa, Italy, on June 28, 1975. He received the MS degree in Computer Engineering from the University of Florence, Italy, in 2000. He is currently a PhD candidate in the Department of Systems and Informatics of the University of Florence, working at the Media Integration and Communication Center. His research interest is focused on the problem of natural interaction between machines and human subjects.