Vision-Based Interpretation of Hand Gestures for Remote Control of a Computer Mouse

Antonis A. Argyros and Manolis I.A. Lourakis

Institute of Computer Science,
Foundation for Research and Technology - Hellas (FORTH),
Vassilika Vouton, P.O.Box 1385, GR 711 10,
Heraklion, Crete, Greece
{argyros, lourakis}@ics.forth.gr
http://www.ics.forth.gr/cvrl/

Abstract. This paper presents a vision-based interface for controlling a computer mouse via 2D and 3D hand gestures. The proposed interface builds upon our previous work that permits the detection and tracking of multiple hands that can move freely in the field of view of a potentially moving camera system. Dependable hand tracking, combined with fingertip detection, facilitates the definition of simple and, therefore, robustly interpretable vocabularies of hand gestures that are subsequently used to enable a human operator convey control information to a computer system. Two such vocabularies are defined, implemented and validated. The first one depends only on 2D hand tracking results while the second also makes use of 3D information. As confirmed by several experiments, the proposed interface achieves accurate mouse positioning, smooth cursor movement and reliable recognition of gestures activating button events. Owing to these properties, our interface can be used as a virtual mouse for controlling any Windows application.

1 Introduction

Being simpler and more intuitive compared to typing at a command prompt, the WIMP (Windows, Icons, Menus and Pointing devices) paradigm dominates most modern graphical user interfaces (GUIs) [12]. WIMP represents an interaction style according to which the user communicates with a computer by means of a pointing device (e.g., mouse, trackball, stylus, etc), that is used to select commands from drop-down menus or icons on the display screen that correspond to predefined actions. In its most common application, the WIMP paradigm demands the user to make physical contact with the pointing device to convey his input. This requirement imposes constraints that, in certain situations, render working with a computer uncomfortable. Such difficulties are emphasized further by the increasing ubiquity of computing power and the continuous downsizing of computer hardware.

In recent years, research efforts seeking to provide more natural, human-centered means of interacting with computers have gained growing interest. A particularly important direction is that of *perceptive user interfaces*, where the computer is endowed with perceptive capabilities that allow it to acquire both implicit and explicit information about the user and the environment. Vision has the potential of carrying a wealth of information in a non-intrusive manner and at a low cost, therefore it constitutes a very attractive sensing modality for developing perceptive user interfaces. Proposed approaches for vision-driven interactive user interfaces resort to technologies such as head tracking, face and facial expression recognition, eye tracking and gesture recognition. A detailed review of existing methods and applications is given in [9]. Due to our interest in systems supporting direct, natural interaction with GUI components, below we limit discussion to such systems only. Zhang et al [14], for instance, propose a vision-based gesture interface system which employs an arbitrary quadrangle-shaped planar object (e.g., an ordinary piece of paper) and a tip pointer (e.g., a fingertip) as an intuitive, mobile input device. Corso et al [5] detect simple visual interaction cues that provide more complex interaction capabilities when sequenced together. Their system supports direct interaction with interface components through actions and gestures. Robertson et al [10] rely on monocular hand tracking and gesture recognition to control an intelligent kiosk. Malik and Laszlo [8] describe a system that supports two-handed, multi-finger gestural interaction by relying on a stereo vision system that estimates finger orientations and fingertip 3D positions.

In this paper, we describe a vision-based approach towards providing a remote, non-contact mouse control interface. This perceptive interface employs standard web-cameras with robust vision techniques and allows the user's hands to subsume the hardware pointing devices in the framework of the WIMP paradigm. Our vision algorithms are fast enough to operate at high video rates on commodity hardware, thus ensuring unhampered interactivity. Two different variants of mouse control interfaces have been investigated. The first makes use of 2D information regarding detected and tracked hands and is based on a single-camera system. The second makes use of 3D information regarding the detected and tracked hands and requires stereoscopic input. Both interfaces have been extensively tested in several real-world situations and the experience we gained from these user trials is presented in detail.

The rest of the paper is organized as follows. Section 2 provides a description of the vision techniques giving rise to the perceptual input on which the proposed gesture-based mouse control interface is built. Section 3 describes the two developed mouse control interfaces. Section 4 presents experimental evidence on the accuracy and the usability of the developed interfaces followed by a critique on the advantages and disadvantages of each of them. Finally, section 5, summarizes the paper by providing the main conclusions from this work.

2 From Images to Gestures

The proposed approach to gesture-based human-computer interaction is based on our previous work on 2D and 3D tracking of multiple skin colored objects. In the following sections, we provide a brief overview of that work. More detailed presentations can be found in [1, 2].

2.1 The 2D Hand Tracker

Our 2D tracker supports tracking of multiple blobs exhibiting certain color distributions in images acquired by a possibly moving camera. The tracker encompasses a collection of techniques that enable the detection and the modeling of the blobs possessing the desired color distribution(s), as well as their temporal association across image sequences. Hands, corresponding to skin-colored blobs, are detected with a Bayesian classifier which is bootstrapped with a small set of training data. Then, an on-line iterative training procedure is employed to refine the classifier using additional training images. On-line adaptation of color probabilities is used to enable the classifier to cope with illumination changes.

Tracking over time is realized through a scheme which can handle multiple targets that may move in complex trajectories, occlude each other in the field of view of a possibly moving camera and whose number may vary over time. Briefly, tracking operates as follows. At each time instant, the camera acquires an image on which the appropriately colored blobs (i.e. connected sets of skin-colored pixels) are detected. A set of hand hypotheses that have been tracked up to the current time instant is also being maintained. The detected hands are then associated with the existing hand hypotheses. The goal of this association is twofold: first, to assign a new, unique label to each new hand that enters the field of view of the camera for the first time; and second, to propagate in time the labels of already detected hands. The tracker has the ability to dynamically adapt to skin color distribution variations that are caused by illumination changes. Furthermore, its prototype implementation on a conventional Pentium IV processor at 2.5 GHz operates on 320×240 live video in real time (30Hz). It is worth pointing out that this performance is determined by the maximum acquisition frame rate that is supported by our IEEE 1394 camera, rather than the latency introduced by the computational overhead for tracking hands.

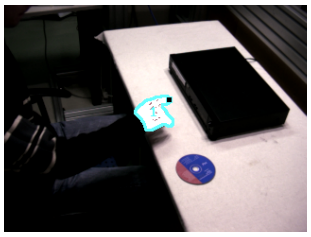
2.2 3D Hand Tracking and Reconstruction

The 3D hand tracker employs a stereoscopic camera system that delivers two synchronized video streams. Each of these streams is processed by an instance of the previously described 2D hand tracker. In order to achieve 3D reconstruction of the hand positions and hand contours (i.e. silhouettes), correspondence of hand blobs between stereo images needs to be established. We formulate this problem as an instance of the *stable marriage problem*. The two sets from which elements to be paired are selected, correspond to the sets of hands detected and tracked in the two images of the stereo pair. According to the original formulation of the stable marriage problem, the two sets whose elements are paired have equal cardinalities. In our case, this might not hold due to the different numbers of hands detected and tracked in each of the stereo images. For this reason, we have extended [6] to handle the case of sets with unequal cardinalities. The required preferences among the members of sets are formed by employing the epipolar constraint [13] on their centroids. Specifically, the better the centroids of the hands satisfy the epipolar constraint, the higher their mutual preference becomes.

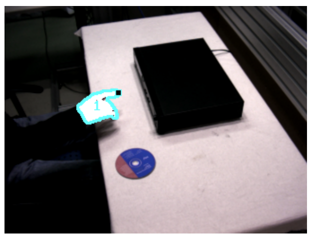
The algorithm described above matches hands between the synchronous images of a stereo pair. To be able to recover the 3D contour of a particular hand, point-wise correspondences of contour pixels are also required. The lack of texture in the images of hands and the presence of considerable depth discontinuities are conditions that do not favor correlation-based approaches towards solving the correspondence problem. Instead, we compute correspondences through a top-down approach in which the basic idea is that if two matching hand contours can be aligned, then the necessary correspondences for 3D reconstruction can easily be extracted. To perform this type of alignment, we employ a robust variant of the Iterative Closest Point (ICP) algorithm [3]. Several robust variants of the ICP algorithm have been proposed that can solve the problem in the presence of measurement outliers and, possibly, shape defects. In our hand tracking scenario, such outliers and shape defects can be due to inaccuracies in skin color detection. Filtering them out is very important because it safeguards the later process of 3D reconstruction against gross errors due to erroneous point matches. The robust variant of ICP that we employ is similar in spirit with the one described in [4]; the major difference is that we use the Least Median of Squares (LMedS) robust estimator [11] in all steps of the general ICP algorithm, instead of the Least Trimmed Squares (LTS) estimator of [4]. The initial contour alignment that is necessary for bootstrapping the ICP algorithm is easily achieved by exploiting orientation information already available to the 2D hand tracker [2].

To recover 3D coordinates, camera calibration parameters along with point correspondences (either matched hand centroids or matched hand contour points) serve as input to a triangulation technique. A typical problem with triangulation relates to the fact that noise in 2D points combined with calibration inaccuracies often result in making skew the back-projected 3D lines defined by the camera optical centers and the corresponding image points. This problem is dealt with by reconstructing each 3D point as the midpoint of the minimal length straight line segment whose endpoints lie on the skew back-projected lines [7].

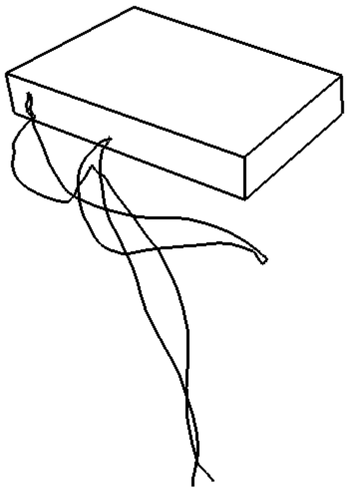
The developed method for binocular hand tracking and 3D reconstruction has served as a building block in a number of diverse applications and has been tested extensively in various environments. One particular application concerns a cognitive vision system whose goal is the automatic interpretation of the activities of people handling tools. Figures 1(a),(b) show a stereo pair from a related experiment in which a person operates a CD player while a pair of cameras is observing the scene. Detected skin-colored pixels are illustrated in white. The contour of the hand is delineated in light blue. Figure 1(c) shows the trajectory of the centroid of the hand as this was computed by the proposed system. As it can be verified from this figure, the hand moves towards the CD player, opens the tray, moves towards the CD, picks up the CD, puts it on the open tray, closes the tray and retracts to its rest position. For improving the readability of the 3D



(a)



(b)



(c)

Fig. 1. (a), (b) a stereo pair from a 3D hand tracking experiment, (b) the computed hand trajectory in 3D

plot, the CD player has also been reconstructed in 3D. The 3D reconstruction of hand contours is achieved at 21 fps on a conventional Pentium IV processor.

2.3 Finger Detection

In many applications like the one dealt with in this paper, it is very important to be able to identify the fingertips of hands. Having already defined the contour of a hand, finger detection is performed by evaluating at several scales a curvature measure on contour points. The curvature measure assumes values in the range $[0.0, 1.0]$ and is defined as

$$K_l(P) = \frac{1}{2}\left[\frac{1 \; + \; \overrightarrow{P_1P} \cdot \overrightarrow{PP_2}}{||\overrightarrow{P_1P}|| \; ||\overrightarrow{PP_2}||}\right], \tag{1}$$

where P_1, P and P_2 are successive points on the contour, P being separated from P_1 and P_2 by the same number of contour points l. The symbol $(\cdot)$ denotes the vector dot product. The algorithm for finger detection computes $K_l(P)$ for all contour points of a hand and at various scales (i.e. for various values of the parameter l). A contour point P is then characterized as the location of a fingertip if both of the following conditions are met:

- $K_l(P)$ exceeds a certain threshold for at least one of the examined scales, and,
- $K_l(P)$ is a local maximum in its (scale-dependent) neighborhood of the contour.

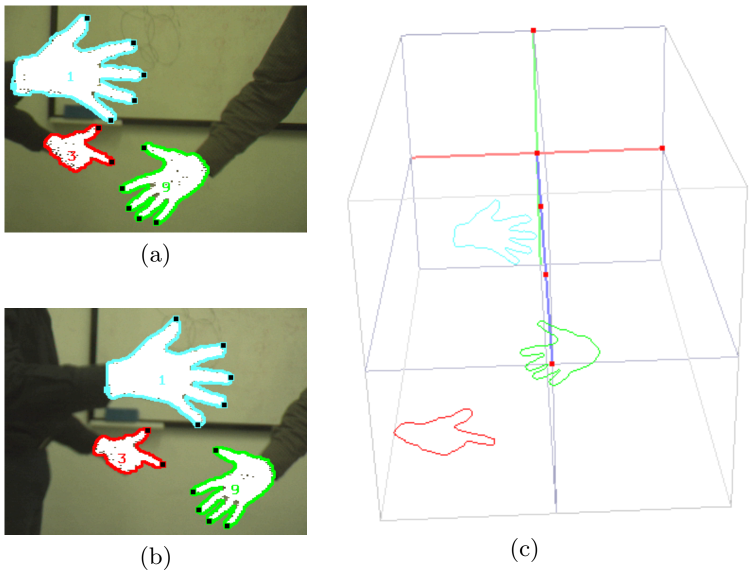


Fig. 2. (a), (b) a stereo pair from a 3D hand tracking experiment; hand contour points identified as fingers appear as black squares, (c) the estimated 3D hand contours. The origin of the coordinate system is at the far face of the parallelepiped.

Figures 2(a),(b) show the left and the right image from a sequence in a 3D hand tracking and finger detection experiment. In this experiment, there exist several hands which are successfully tracked among images. Fingers are also detected and denoted with the aid of black squares. Figure 2(c) shows the 3D reconstruction of the contours of the tracked hands which was achieved as described in section 2.2. In the reported experiments, the curvature threshold of the first criterion was set to 0.7.

3 From Gestures to Mouse Control

The 2D and 3D trackers that were previously described provide the perceptual input for defining a set of hand gestures through which gesture-based control of a computer's mouse can be exercised. We have defined two such gesture vocabularies, one based on the perceptual input supplied by 2D tracking and one based on that supplied by 3D tracking. In both cases, the set of gestures have been defined according to the following criteria:

– *Intuitiveness and ergonomics:* The defined gestures should be easy to learn and use.
– *Lack of ambiguity and ease of recognition:* The definition of the hand gestures should facilitate their automatic interpretation.

In the remainder of this section, we describe the two defined sets of gestures.

3.1 Mouse Control Based on 2D Hand Gestures

The defined 2D gesture vocabulary involves *static gestures*, i.e., gestures in which the information to be communicated lies in the hand and finger posture at a certain moment in time. To avoid errors due to image noise, it is assumed that these hand postures last for at least a short, fixed amount of time. In the actual implementation of the system, a minimum duration of half a second is employed. Assuming a frame rate of 30Hz, this means that in order to recognize a certain posture, this has to be maintained for a minimum of fifteen consecutive image frames.

The 2D hand gestures involve both hands of a user, each of which has a special role. More specifically, one hand is responsible for moving the mouse pointer and is therefore called the "pointer hand". The other hand is mainly responsible for issuing different commands and is therefore called the "commanding hand". These roles are not statically determined but can be chosen by the user: The first hand that appears in the field of view of the camera with one extended finger becomes the pointer hand. Then, the second appearing hand assumes the role of the commanding hand. Note that the roles of hands may be interchanged during system operation; This simply requires moving the hands out of the camera field of view and bringing them back in with the appropriate order and posture.

What follows, is a list of hand postures that the developed application is able to recognize, accompanied with their corresponding interpretation in the context of the computer mouse control application. Representative instances of these gestures can also be seen in Fig. 3.

- *Mouse control activation and deactivation:* In many cases, the user needs to activate and deactivate the interpretation of gesture-based commands for the mouse. This is achieved by a gesture involving both hands, each of which is presented with five extended fingers (see Fig. 3(a)). This posture toggles between the activation and the deactivation of gesture-based mouse control.
- *Mouse move:* This is achieved through the movement of the pointer hand (see Fig. 3(b)). The coordinates of the centroid of the pointer hand within the processed image are appropriately mapped to mouse cursor coordinates on the computer's desktop. The hand centroid is selected because its coordinates are less susceptible to image noise. Moreover, this makes the system's operation independent of the number of extended fingers. Nevertheless, the hand centroid cannot easily be located in extreme image positions (i.e. those close to its top, bottom, left and right borders). For this reason, strips along image borders are excluded from consideration; the rest of the image is then linearly mapped onto the computer's desktop.
- *Press left mouse button:* The commanding hand shows five extended fingers (see Fig. 3(c)).
- *Release left mouse button:* The commanding hand shows less than five extended fingers after a detected "press left mouse button" event.
- *Left mouse button click:* This is achieved by the combination of a "press left mouse button" and a "release left mouse button" gesture.

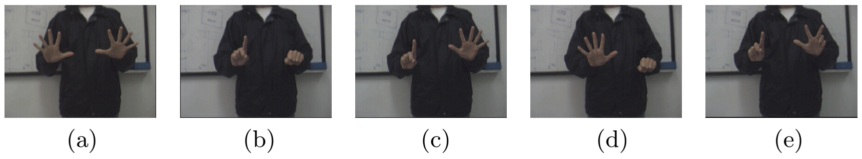


Fig. 3. Representative instances of the gestures used to control the mouse pointer (a) activate and deactivate gesture-based mouse control, (b) mouse move, (c) left button click, (d) right button click and, (e) left button double click

- *Right mouse button click:* The pointer hand presents five extended fingers (see Fig. 3(d)).
- *Left mouse button double click:* The commanding hand shows three extended fingers (see Fig. 3(e)).

At this point, it should be noted that in contrast to previous methods such as [14, 5], our approach does not require the presence in the image of any artificial objects for defining and/or supporting hand gestures. Based on the above set of gestures, the user of the system may also perform other useful mouse operations such as mouse drag-and-drop. This is naturally implemented as a "press left mouse button" gesture with the commanding hand, followed by a move operation achieved with the pointer hand and finally by a "release left mouse button" gesture, again by the commanding hand.

3.2 Mouse Control Based on 3D Hand Gestures

The key idea behind the definition of a vocabulary involving 3D gestures is that additional information can be conveyed to the system depending on the distance of the hand(s) from the camera system. Contrary to the case of the 2D vocabulary, the 3D one requires gestures of only one hand, with the exception of the case of activation and deactivation of the gesture-based mouse control. Another important difference is that the set of gestures includes two *dynamic gestures* as opposed to the strictly static ones employed in the 2D case. The complete 3D gestures vocabulary is as follows:

- *Mouse control activation and deactivation:* This is implemented as in the 2D vocabulary, i.e. ten extended fingers detected in both hands. As soon as mouse activation is detected, the distance between the commanding hand and the camera is estimated and considered as a reference distance for interpreting other gestures.
- *Choice of the pointer hand:* The pointer hand is the one that is closest to the camera.
- *Press left mouse button:* The pointer hand is extended towards the camera with none of its fingers extended. This is a dynamic gesture that is recognized based on posture characteristics (zero extended fingers) and the decrease of the distance of the pointer hand from the camera system.
- *Release left mouse button:* The pointer hand roughly retracts back to the reference distance after a "press left mouse button" gesture.

– *Left mouse button click:* Implemented as the combination of a "press left mouse button" gesture followed by a "release left mouse button".
– *Right mouse button click:* Similar to the left mouse button click, with the difference that the pointer hand has five extended fingers instead of none.
– *Left mouse button double click:* Similar to the left and right mouse button clicks, with the difference that the pointer hand has three extended fingers.

4 Experiments and Validation

The developed methods for controlling a computer mouse through 2D and 3D hand gestures have been extensively tested in several real world situations. The focus of the conducted experiments was not on the robustness and the performance of the supporting 2D and 3D hand tracker components since these have been extensively reported elsewhere [1, 2]. Instead, the focus was on assessing the usability and the ergonomics of each of the proposed human-computer interfaces and on comparing their relative performance.

As a first step, ten of our colleagues were introduced to the operation of both interfaces and were asked to perform certain mouse functions by employing both of them. One specific task that the users were asked to perform was to launch the MS Paint application, move the MS Paint window away from its popup location, select the free pencil tool, a specific pen size, a shape and a color and write a small piece of text on the image canvas. This type of test requires the use of all gestures. Analogous tests were designed involving the operation of the calculator application. Figure 4 shows an example of what one of the users scribbled by using the 2D gestures interface. The quality of the text shows that the developed interface supports smooth and accurate control of the mouse pointer and the button events triggered by hand gestures. This is particularly important, especially if one takes into account that a 200×280 window of the processed image was linearly mapped onto a 1024×768 computer desktop.

User experiences were then gathered and evaluated. The general conclusions drawn can be summarized as follows. The 3D gestures are more easy to understand and assimilate compared to the 2D gestures. An additional advantage of the 3D set is the fact that, excluding mouse activation and deactivation, only one hand is required to operate the mouse. The users also found very intuitive the implementation of the various mouse clicks as a hand motion towards the camera, followed by a hand motion towards the torso because this is analogous to pressing an imaginary 3D button.

However, the same users found the interface based on the 2D gestures more responsive and therefore much more user-friendly. This is attributed to the higher frame rate that is achievable in the single camera setting compared to the stereo one. An additional disadvantage of the 3D interface is the fact that mouse click, although more intuitive, is less accurate compared to its 2D counterpart. This is because the commanding hand and the pointer hand coincide. Therefore, if a user does not move his hand in a direction strictly perpendicular to the image plane, this affects the desktop location on which the image

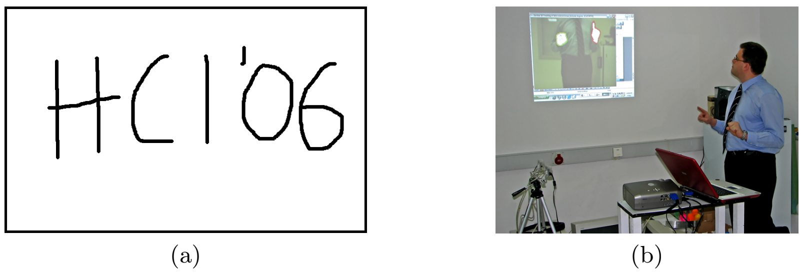


Fig. 4. (a) Text scribbled by a user operating the MS Paint application using the 2D gestures mouse control interface. (b) A snapshot from an experiment where the developed mouse control interface is being used to control a MS PowerPoint presentation. The employed system of cameras can be seen at the lower right corner of this image.

click is issued. As a general remark, the consensus among the users participating in these usability tests was that the set of 3D gestures is probably more appropriate for the novice user. However, after some user training and acquaintance with the gestures of the 2D interface, the former constitutes a much preferred choice. This conclusion has been also validated by tests involving experienced users of the system that were able to control and run formal MS PowerPoint presentations in real-world conditions for extended (i.e., more than half an hour) periods of time. A snapshot from such a presentation is shown in Fig. 4(b). A representative video from such an experiment can be found at http://www.ics.forth.gr/∼argyros/research/virtualmouse.htm. The increased user acceptance of the 2D interface over the 3D one is also important from a technical point of view, since the former has less hardware requirements (one, as opposed to two cameras), less setup effort (no need for stereo calibration, better portability, etc) and lower computational requirements. Finally, it is also worth mentioning that the permissible volume for moving the hands is smaller in the case of the 3D interface. This is because it requires the hands to be visible in both cameras, thus restricting them to lie in a space smaller than the visual pyramid of any camera alone.

5 Summary

In this paper, we proposed two different systems for human-computer interaction based on hand gestures. More specifically, two different hand gesture vocabularies were proposed for remotely operating the mouse of a computer. One of the developed vocabularies is based on static, 2D hand postures while the second relies on 3D information and uses a mixture of static and dynamic gestures. Both interfaces have been extensively tested and their relative advantages and disadvantages have been assessed. Overall, both proposed approaches are robust and capable of supporting vision-based HCI in real-world situations. However, the 2D one seems preferable to the trained user.

From a design and implementation point of view, the proposed 2D and 3D hand gesture vocabularies and the computational techniques used to recognize them were kept quite simple. Clearly, much of the robustness of the proposed system is attributed to the quality and the stability of the underlying perceptual processes that transform raw image data into a symbolic description that is amenable to interpretation. Characteristics such as the accuracy of skin color detection under varying illumination conditions, the robustness of tracking several moving objects under severe occlusions and the capability of accurate segmentation of fingertips and extraction of 3D information are of paramount importance in terms of the usability of the developed system. Current efforts focus on the development of richer 2D and 3D sets of human-body gestures for increasing the information content in the interface between humans and computers.

Acknowledgements

This work has been partially supported by EU IST-2001-32184 project ActIPret and EU-IST NoE MUSCLE (FP6-507752). The contributions of Mickael Marechal, Benjamin Marie and Cedric Groyer in the implementation and testing of the proposed interface are gratefully acknowledged.

References

1. A.A. Argyros and M.I.A. Lourakis. Real Time Tracking of Multiple Skin-Colored Objects with a Possibly Moving Camera. In *Proceedings of ECCV'04*, pages 368–379, 2004.
2. A.A. Argyros and M.I.A. Lourakis. Binocular Hand Tracking and Reconstruction Based on 2D Shape Matching. *Submitted to ICPR'06, under review*, 2006.
3. P. Besl and N. McKay. A Method for Registration of 3-d Shapes. *IEEE Trans. on PAMI*, 14(2), 1992.
4. D. Chetverikov, D. Svirko, D. Stepanov, and P. Krsek. The Trimmed Iterative Closest Point Algorithm. In *Proceedings of ICPR'02*, 2002.
5. J.J. Corso, D. Burschka, and G.D. Hager. The 4D Touchpad: Unencumbered HCI With VICs. In *Proceedings of CVPR-HCI'03*, page 55, 2003.
6. D. Gale and L.S. Shapley. College Admissions and the Stability of Marriage. *American Mathematical Monthly*, 69(9), 1962.
7. R. Goldman. Intersection of two lines in three-space. *In Graphics Gems I*, page 304, 1990.
8. S. Malik and J. Laszlo. Visual Touchpad: A Two-Handed Gestural Input Device. In *Proceedings of ICMI'04*, pages 289–296, 2004.
9. M. Porta. Vision-based User Interfaces: Methods and Applications. *Int. J. Human-Computer Studies*, 57(1):27–73, 2002.
10. P. Robertson, R. Laddaga, and M. van Kleek. Virtual Mouse Vision Based Interface. In *Proceedings of IUI'04*, pages 177–183, 2004.
11. P. J. Rousseeuw. Least Median of Squares Regression. *Journal of American Statistics Association*, 79:871–880, 1984.

12. A. van Dam. Post-WIMP User Interfaces. *Commun. ACM*, 40(2):63–67, 1997.
13. Z. Zhang. Determining the Epipolar Geometry and its Uncertainty: a Review. *Int. Journal of Computer Vision*, 27:161–195, 1998.
14. Z. Zhang, Y. Wu, Y. Shan, and S. Shafer. Visual Panel: Virtual Mouse, Keyboard and 3D Controller With an Ordinary Piece of Paper. In *Proceedings of PUI'01*, pages 1–8, 2001.