

Computer Vision approaches to solve the screen pose acquisition problem for PerspectiveCursor

Miguel A. Nacenta

Technical Report HCI-TR-06-01, Computer Science Department, University of Saskatchewan

December 5th, 2005

ABSTRACT

PerspectiveCursor is a new interaction technique for multi-display environments that significantly improves human interaction with computer systems. Computer Vision has the potential to provide a solution for the implementation of this interaction technique that might be better than the current ones. A survey of the current state-of-the-art in 2D and 3D vision shows that there are many techniques that might be of use in finding the spatial relationships between the point of view of users and the displays in the environment, which is the main requirement for an implementation of PerspectiveCursor.

INTRODUCTION

With the advent of cheap cameras and the continuous increase in computing power of consumer micro-processor systems, Computer Vision and Image Processing have moved from being research fields with applications only in very specialized fields (i.e. medicine, robotics) to being pervasive in consumer products (i.e. optical mice, video cameras).

Tracking is one of the technologies that is already being transformed radically by the substitution of electromagnetic measurement and ultra-sound methods by computer vision. Numerous new applications in research (but also developing towards consumer products) make use of tracking in some way. The new paradigms of computing environments and Human-Computer interaction [23, 28, 34] make extensive use of the position coordinates of users and objects to provide meaningful and easy to use services.

Computer Vision Tracking is getting to the point where it can provide unobtrusive, reasonably precise, reliable tracking at an affordable price.

The particular application that we are interested in is called PerspectiveCursor [22]. PerspectiveCursor is an interaction technique that provides access to a dynamic multi-screen environment from a single input device. This technique, which is explained in more detail in the next section, requires knowledge of the position of the user and the position of the different displays in order to create an intuitive input/control translation map.

PerspectiveCursor has the potential of improving single-user and groupware systems, but the current implementation, based in electro-magnetic tracking, has several very important drawbacks: it is very unstable in

presence of metallic objects, it is extremely expensive (around \$10 000 for the most basic tracking equipment) and it is tethered.

This paper aims at exploring the different alternatives inside the Computer Vision field that would allow a cheap and reliable implementation of PerspectiveCursor. For this, we consider the minimal requirements of the PCursor technique and the knowledge of the particular setting to try to find a match between the different CV techniques and an acceptable solution.

The rest of the report is organized as follows: first, we describe PerspectiveCursor and clearly state the particularities of the Computer Vision problem that we want to solve, including the constraints, requirements, and desirable characteristics of a solution. Then we explore current literature at the same time that we comment on the benefits and drawbacks of each technology and how these would be helpful in providing a full or partial solution of the problem. Finally, we present our conclusions and the techniques we consider most valuable for this particular endeavor.

MOTIVATION & PROBLEM STATEMENT

In trying to find a suitable solution for our problem, we first need to explain how PCursor works, and then detail what exactly "suitable" means.

PerspectiveCursor

The technique of PerspectiveCursor is motivated by the need of accessing several displays using a single cursor in complex multi-display scenarios. In most current multi-display settings, users usually work with two displays that are connected to the same machine and positioned in a very simple layout, namely, in the same plane and next to each other (see Figure 1).

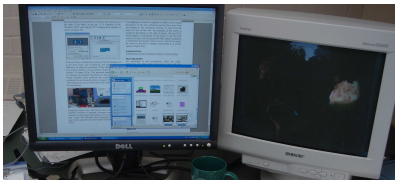


Figure 1. Simple multi-monitor configuration



Figure 2. Input map for the configuration in Figure 1

The standard mapping of input providing by most operating systems (see Figure 2) is sufficient in these cases because the layout of screens is very simple. This mapping connects the contiguous sides of two displays so that moving the cursor with the mouse across the contiguous side of the screen makes it appear in the next one.

However, this approach is not good enough if we have a setting like the one in Figure 3, in which the different (and possibly mobile) displays are located in different planes, with different angles, and possibly occluding each other.



Figure 3. A more realistic multi-display setting

A solution to this problem is to calculate the input mapping of the display dynamically in the way that each user perceives it. This means that the cursor will not be able to access points in a screen that she cannot see from her point of view (for example, because the screen is partially occluded by another one, or just looking to the opposite side). The equivalent of Figure 2 for a user sitting in the chair to the right in the room of Figure 3 is displayed in Figure 4.

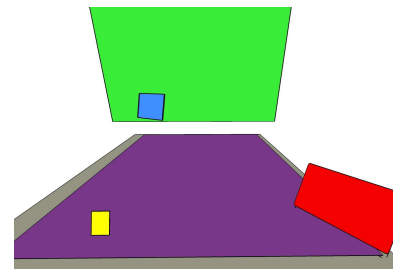


Figure 4. The display-input map of the scenario in Figure 3

Simplifying the technique, we can say that Figure 4 depicts in a flat image which position of the mouse would correspond to which display in a particular moment (only if the user stays sitting in the chair to the right). The coordinates of the image (plain x,y coordinates) represent the position of the mouse, and for each position of the mouse we have to find the corresponding point in the corresponding display.

In this particular example, it means that to move the cursor from any position in the PDA (represented by the yellow screen) to the vertical wall display, the cursor would have to go through the table-top display (purple), until it reaches the green surface, i.e., we would have to move the mouse “up”.

Current implementation

The proof of concept for the technique was created in a room setting very similar to that in Figure 3. Some of the displays are fixed, and thus, their positions and orientations were known a priori, and the system did not have to track them. However, in order to create the perspective map shown on Figure 4, the system had to track the user’s head pose and the position of any other mobile display.

This tracking was done with a commercial electromagnetic tracking device [25]. Although these kinds of devices provide good precision and an easy way to access the data (it provides absolute 6-degrees of freedom information), they are very expensive and are not compatible with metallic objects.

With all the pose information of displays and user, the system creates a simple 3D model that is updated several times a second. From the 3D model and with the knowledge of the position of the user’s head, we make a perspective projection that represents how the user perceives the displays from her point of view, and in this way we apply the control-display mapping (i.e. how the mouse moves the cursor in and between displays).

Expectations from CV Technologies

So far we have not discussed much about Computer Vision, but the reader has probably noticed already that there several aspects of the problem described that could be satisfactorily solved by a CV solution, in particular:

- The use of CV might reduce the cost of the hardware needed to implement PCursor in one to two orders of magnitude due to the availability of commercial inexpensive cameras (a couple of medium quality cameras are still much cheaper than a single magnetic tracker or ultrasound tracking systems).
- CV is not affected by electromagnetic properties of the elements of the environment that we need to track.
- A CV solution could be made non-obtrusive, and could provide non-tethered solutions through the use of existing wireless networks.
- The perspective/projective nature of the problem fits perfectly with a solution based on CV from cameras because cameras perform implicitly the same geometric transformations that we use (perspective).

Requirements

There are a certain number of requirements that the CV solution of PCursor must fulfill to be effective. We divided them into functional and performance/interaction.

Functional requirements

1. *Projection.* From whatever number of input frames that the system receives as input (it might have several cameras, and use several frames taken in different times), the system should be at least able to determine the projection of the displays in the user's image plane for a reasonable angle. Although the exact angle has not yet been studied, a good estimate could be above the 50° in the horizontal plane and 40° in the vertical plane (see Figure 5).

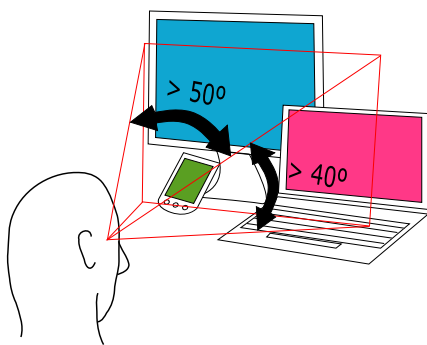


Figure 5. The angle of operation should be wide enough (ideally, it would cover all the space around the user)

2. *Identification and mapping.* The system must be able to identify to which display each projection corresponds (Figure 6), and have at least an approximate mapping from the points of the projection into each display's coordinate system. This is equivalent to having an approximation of

the transformation between the three different coordinate systems of Figure 7.

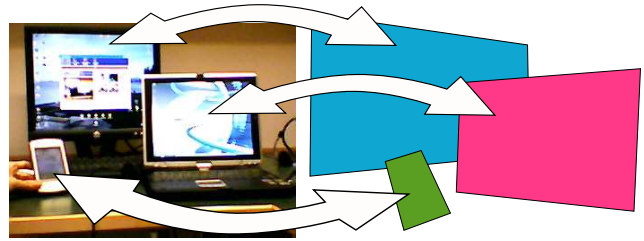


Figure 6. The system should be able to identify which display projection (right) corresponds to which physical display (and machine).

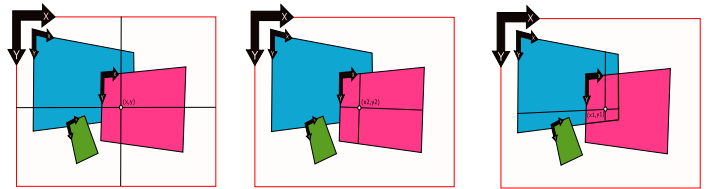


Figure 7. Coordinate systems of the input mapping (x,y), pink screen (x_2,y_2) and blue screen (x_1,y_1).

Interface/Performance requirements

PerspectiveCursor is an interaction technique, and as such, it is important that it complies with requirements imposed by the human side of the interface.

3. *Interaction rate.* The speed and timeliness of the analysis of the frames to map the input to the different screens is fundamental for the users. If the system is too slow, or cannot actualize the input mapping enough times per second, the users will not be able to use the technique efficiently. Ideally, the system will work a rate higher than 20 processed frames per second and a delay below 100ms, but less restrictive parameters could be acceptable depending on the application.
4. *Mobility.* The user must be able to move around and change positions at a normal rate, i.e. we cannot assume that the user is static or has constricted movements.
5. *Robustness.* The system should provide the input mapping reliably, even in the presence of occlusion. Although some errors in the mapping might be acceptable, the quality of the interaction degrades very fast with an increase of glitches.
6. *Self-configurability.* Although some initial configuration steps might be acceptable at the beginning of the session, the CV methods that the system uses must not rely in continuous prompting on the user (for example, to identify salient features of an image).

7. *Load.* Besides the necessary computations to run PCursor, the systems that control the different displays of the multi-display environment will also need to run other applications; therefore, it is advisable that the computations don't overload the system.

Non-requirements

When analyzing which CV technologies can be applied to solve our problem, we must also take into account that some of the restrictions that previous CV applications impose on their systems can be relaxed due to the knowledge of our particular scenario or to the control that we can exert on the environment. This is just a list of common restrictions of CV systems that do not necessarily apply to our setting.

1. *Calibration and initialization.* Some 3D vision techniques have been developed lately that do not depend on the calibration of the camera(s) or that self-calibrate [non-calibrated]. We can safely assume that our cameras are calibrated, and that the intrinsic parameters of the cameras are known to the system.
2. *Precision.* The real degree of precision required for PCursor is not very high due to the use of a relative positioning device. Although a high level of precision might be beneficial, errors below 5° might be acceptable.
3. *Modification of the environment.* While some systems have the restriction that they must capture the world "as is", we can modify the environment to make the technique simpler, more reliable or more efficient. For example, we can use InfraRed or other kinds of fiducial markers [1, 15, 16]. In fact, we can also include *active* elements with their own sensors that change the image that is perceived by the camera(s) as in [17].

Desired characteristics

Although not strictly required, there are some characteristics that would make our solution even better.

1. *Non-invasiveness.* The less we burden the user with cables, things to wear or to attach to her devices, initialization or re-initialization procedures, the better the solution.
2. *Camera costs.* The results and quality of the implementation may depend strongly on the quality of the cameras used, but so does the price, as cameras (and lens) can easily become the most expensive component of our system. As we stated in the introduction, the affordability of the system is one of the main goals, and we should strive to implement algorithms and techniques that are robust and can work with the most economical cameras.

3. *Modification of the environment.* In the *non-requirements* section we suggested that modification of the environment may greatly improve reliability and simplify the CV techniques used; however, it also restricts the scenarios in which our solution could be used, and we must thus be aware of the trade-off.

SURVEY

In this section we review the technologies that *might* be of value in finding a CV implementation of the PCursor technique. The survey is built bottom-up, as it analyzes first pixel-level techniques that might not by themselves provide a full solution, but might be part of a more complex system. Throughout this section, we keep increasing the level of abstraction and we discuss also the drawbacks and trade-offs that different technologies pose.

Most of the solutions that we propose take advantage of the particularity of the problem we are trying to solve, that is, solutions could be viable because they are ad-hoc. This constitutes an application of the principle that the more information and assumptions we can make about the image(s) and the problem space, the easiest it is to provide a CV solution.

2D Approaches

As we mentioned earlier, cameras provide us with perspective projection for free without any need to make 3D inferences by software (requirement 1) [9, 24]. If the camera is located just in the point of view of the user, or close enough to it, we would be able to obtain images that are already very close to the input map that we want to obtain (see Figure 8).



Figure 8. A head mounted camera (left) and one of its possible captures (right).

Now our task would be to analyze the image captures to transform them into input maps that assign regions to particular displays (requirement 2).

Detecting a screen in an image

From the image in Figure 8 it would be very easy for a human to find and delimit the screens. However, even though this particular example is relatively simple (no occlusion, screens perpendicular to the line-of-view), it is difficult to provide a generic algorithm that could do it due to several limitations:

- There are many other objects in a frame that share geometric characteristics with a screen (frame, mouse pad).
- Screens are rectangular, but their projections in the image don't have to be (see Figure 9). Size and geometry of the representations varies wildly with position, distance and orientation.
- Color and brightness of the screen or the frame are not constant across different screens and devices, ruling out simple thresholding methods.
- The content of a screen can be virtually anything, making it even more difficult to detect the edges.

There are multiple techniques that could help us detect displays in one screen. A simple approach would be to detect straight lines or segments from the edge detection map using Regression Analysis [20] or the Hough Transform [5, 29, 35]. Ideally, this would result in a line map like that in Figure 10.



Figure 9. The shapes of the screen representations in the image are not rectangular, but trapezoidal

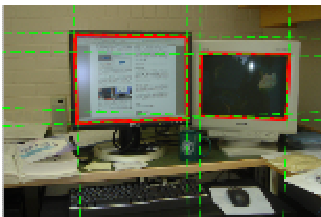


Figure 10. Ideal search for screens using line-finding algorithms.

Even assuming that the line detection is reliable and robust, we would still have to find which groups of lines form closed geometric shapes, and which correspond to actual displays.

This approach has the advantage that it could find the corners of the screens with sub-pixel precision, but we are not sure to what extent the line-detection mechanisms and segment-grouping algorithms can be fine-tuned to obtain a reasonable result (Figure 11 is an example of detection for one of our images).

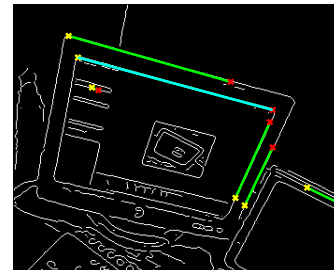


Figure 11. A very unrefined example of line detection using the Hough transform (Matlab)

Using a higher-level approach, it would be possible to detect geometrical shapes instead of segments in order to obtain candidates of screen surfaces. This is known to be possible using the Hough transform [16, 19] and shape invariants [21]. An intermediate step between both would be detecting corners and T-junctions, as in [8].

A third possible approach in order to identify the displays in an image would be to use template matching. The idea is that if we know the content of each screen (which could be sent over the network to the machine in charge of the image processing), we can try to find it in the image within certain limits of deformation, as is done in [13]. However, to our knowledge, this method has not yet been applied in a dynamic context with changing templates, and might not be computationally viable.

Assuming that all the techniques discussed so far are capable of finding the screens in an image, we still have the problem of occlusion. If one screen is occluding another, we still want to be able to detect both of them, but with occlusion the geometrical shapes of a screen's representation of an image could have any number of sides. Hough-transform is probably more robust for these cases than the other alternatives because it is known to deal well with occlusion.

Identification and coordinate transformation

Once we have found the displays in the image, we must be able to identify to which physical display they correspond. This is trivial in the case of template matching (assuming that the images displayed are different from each other), but is not trivial for the other techniques, which leads us to think that the optimal solution might probably be a combination of active template matching and any of the other low-level techniques.

Markers and active elements

A very simple way of making the detection and segmentation problem much easier is to modify the environments to contain markers that are known in some way by the system. These could be special patterns that the system is trained to recognize [26, 14, 15], or markers using a different spectrum of detection, normally infra red [1, 27].

Modifying the environment in this way might limit the applicability of our system, but it is a relatively easy and

cheap method and can improve robustness (requirement 5) significantly.

Sequences of images

So far we have assumed that we only have one single frame to analyze. However, the analysis can be made much more efficiently if we analyze sequences of images that make more information available, in a similar way to what was done in [3]. This could also potentially help our identification problem: a particular screen could be identified interactively or through pattern matching once at the beginning of the session (see the Identification and coordinate transformation section above). After, the system could just keep track of it through time.

Movement model estimation and Kalman filtering [12] can also help to predict the new positions of the screen and therefore reduce the computational cost of algorithms (many algorithms can benefit from knowing where to start looking for screens).

Optical Limitations

The projective perspective that we have assumed so far is fundamental for most of the algorithms that we have reviewed. However, the image taken by cameras doesn't correspond perfectly with a *pinhole perspective* and are subject to lens distortion. Moreover, these deviations of the real images from the idealized model are even more extreme when the camera is wide angle. The main consequence of the distortions is that straight lines of the real world become curves in the image, invalidating many of the assumptions made in the previous paragraphs [36].

Fortunately, this kind of trade-off between distortion and wide-angle can be softened by good calibration and transformations in the digital representation of the image [6, 36].

3D Approaches

As we have seen in the previous section, putting the camera in the point of view of the PCursor-user simplifies the problem, and keeps us from having to compute 3D calculations of the environment. However, it has two main drawbacks: it is intrusive to force the user to wear a camera in her head, no matter how light it is, and in a cooperative situation where there are several users, the processing made for one user are useless for others. This results in lots of redundant computations.

If, instead, we could form a centralized 3D model of the users and displays in the environment, all computations would be performed once taking the data from several sources of information, and would serve any number of users, which, in turn, could help improve the accuracy of the system.

Building a complete 3D model from what is captured from one or several cameras is, however, a big challenge. Most basic text books on computer vision and 3D vision include chapters that introduce projective geometry [10, 30, 37],

which is fundamental to translate points in the images obtained into real-world 3D coordinates.

Stereo and Multiple-View Vision

If a camera is static, and we know the camera's position and inner parameters (the calibration matrix), each pixel in the image represents a ray in space. If we have more than one camera looking at the same scene, and we have identified a pixel that corresponds to the same object in several images, then we can determine the position of that object by finding the intersection of the different rays.

Of course, the difficult part is finding the pixels that correspond to the same location in several images. Fortunately, we can use fiducial markers and pretty much the same shortcuts we have described above in the paper (active markers, infrared capture).

For this approach, the cameras should be in fixed positions, and this may limit our range of possible applications, because the tracking becomes *room dependent*. However, several commercial products [33, 7, 2] have proven that this solution is viable (although not yet cheaper than the electromagnetic tracking counterparts).

Single Camera

Using multiple cameras is not the only solution to 3D tracking. In fact, monocular vision 3D tracking is a rapidly growing field of research that is of particular interest to us. The field is surveyed in an extraordinary work by Lepetit and Fua [18].

The main premise in this approach is that if we know some correspondences of the pixels in our image and the physical relationships of the corresponding points, we can calculate pose (position and orientation) of the camera respect to that object (for example, the calibration object shown in Figure 12). This is the approach taken by the ARToolkit [14, 15], in which only the relative position of the camera (user) and the marker are needed.

This technique works in a way inverse to camera calibration; if the internal parameters of the camera are known and the spatial relationships of several identifiable points are determined (for example, a marker pattern in a plane), then we can infer the displacement and rotation of the camera that captures the actual recorded image.

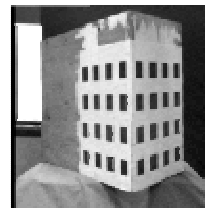


Figure 12. Calibration object for monocular 3D tracking

In our solution it is not enough to find the position of the camera with respect to a single object as in [32]; we need the position of the camera also with respect to a number of

other objects (the screens). It is possible to solve such problem, but there is, to our knowledge, not yet a complete development that we could apply to our particular situation. However, Peter Sturm comes very close to it in a paper that explains how to create a 3D estimation of planar surfaces taking sequences of images as input [31].

CONCLUSIONS AND LESSONS LEARNT

There are many approaches to 3D vision and tracking that could be of use for solving the problem stated in the first sections. In this survey we have barely scratched the surface of a field that is large and very complex, but this research makes us believe that with the application of several of the techniques found and the development of ad-hoc algorithms, a good solution can be found.

Although it is very difficult to figure out which of the techniques would actually be of use (due to lack of time, we couldn't delve into the details of most of the algorithms proposed in the literature), We are almost sure that using a 3D approach with mobile cameras, IR fiducial markers and predictive algorithms we can build a good, precise and relatively inexpensive PCursor implementation. This implementation would take some time to be developed, as the reader can probably tell from the discussions above.

REFERENCES

1. Aliakseyeu, D., Martens, J.B., Subramanian, S., Rauterberg, M., Interaction Techniques for Navigation through and Manipulation of 2D and 3D data. In *Proceedings of Eighth Eurographics Workshop on Virtual Environments* (2002)
2. Atracsys. <http://www.atracsys.com>.
3. Basu, S., Essa, I, Pentland, A., Motion Regularization of Model-Based Head Tracking. In *Proceedings of ICPR'96* (1996)
4. Burns, J.B., Hanson, A., Riseman, E.M., Extracting straight lines. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Vol 8, 4 pp. 424-455 (1986)
5. Duda, R.O., Hart, P.E. Use of the Hough transformation to detect lines and curves in pictures. In *Communications of the ACM*. Vol 15, Iss. 1. pp.11-15 (1972)
6. El-Melegy, M., Farag, A.A., Non-metric Lens Distortion Calibration: Closed-form Solutions, Robust Estimation and Model Selection. In *Proceedings of IEEE International Conference on Computer Vision (ICCV'03)* (2003)
7. Etemeyer. <http://www.etemeyer3d.com>
8. Favaro, P., Duci, A., Ma, Y., Soatto, S. On Exploiting Occlusions in Multiple-view Geometry. In *Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV'03)*. (2003)
9. Flocon, A. Curvilinear perspective: from visual space to the constructed image. *University of California Press* (1987).
10. Freeman, H., Machine vision for three-dimensional scenes. *Boston Academic Press*. (1990)
11. Hartley, R., Zisserman, A., Multiple View Geometry in Computer Vision. *Cambridge University Press*. (2000)
12. Harvey, A.C., Forecasting, Structural Time Series Models and the Kalman Filter. *Cambridge University Press* (1989)
13. Jurie, F., Dhome, M., A simple and efficient template matching algorithm. In *Proceedings of the International Conference on Computer Vision*. (2001)
14. Kato, H., Billinghurst, M. Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System. In *(IWAR '99) Proceedings of the 2nd IEEE and ACM International Workshop on Augmented reality 20-21 pp.85 – 94*. (1999)
15. Kato, H., Billinghurst, M., Poupyrev, I., Imamoto, K., Tachibana, K. Virtual object manipulation on a table-top AR environment. In *International Symposium on Augmented Reality pp. 111-119*. (2000)
16. Leavers, V.F. Shape Detection in Computer Vision using the Hough Transform. *Springer-Verlag*. (1992)
17. Lee, J., Hudson, S.E., Summer, J.W., Dietz, P.H., Moveable Interactive Projected Displays Using Projector Based Tracking. In *Proceedings of UIST'05* (2005)
18. Lepetit, V., Fua, P. Monocular Model-Based 3D Tracking of Rigid Objects: A Survey. In *Foundations and Trends in Computer Graphics and Vision*. Vol 1. No 1 (2005)
19. Lo, R., Tsai, W. Perspective-transformation-invariant generalized Hough transform for perspective planar shape detection and matching. In *Pattern Recognition vol.30 iss.4. p.383*. (1997)
20. Meer, P., Mintz, D., Rosenfeld, D.Y., Kim, D.Y. Robust Regression methods for computer vision: a review. In *International journal of computer vision*. Vol 5, Iss.1, pp-59 (1991)
21. Mundy, J.L., Zisserman, A. Geometric invariance in computer vision. *MIT Press, Cambridge Mass*. (1992)
22. Nacenta, M., Sallam, S., Champoux, B., Subramanian, S., Gutwin, C. Perspective Cursor: Perspective-based interaction for Multi-Display Environments. *Submitted to CHI'05*. (2005).
23. Norman, D. The Invisible Computer. *MIT Press*. (1998)
24. Pirenne, M.H., Optics, painting & photography. *Cambridge U.P.* (1970)
25. Polhemus. <http://www.polhemus.com>

26. Rekimoto, J. and Saitoh, M. Augmented Surfaces: A Spatially Continuous Work Space for Hybrid Computing Environments. In *Proc. of CHI '99*, pp. 378-385 (1999)
27. Ribo, M., Pinz, A., Fuhrmann, A.L., A new Optical Tracking System for Virtual and Augmented Reality Applications. In *IEEE Instrumentation and Measurement Technology Conference*. (2001)
28. Schilt, B., Adams, N., Want, R., Context-aware computing applications. In *Workshop on Mobile Computing Applications*. (1994)
29. Shpilman, R. Brailovsky, V. Fast and Robust Techniques for detecting Straight Line segments Using Local Models. In *Pattern Recognition Letters 20*. pp 865-877 (1999)
30. Sonka, M., Hlavac, V., Boyle, R., Image Processing, Analysis, and Machine Vision. *Brooks/Cole Publishing company*. (1999)
31. Sturm, P. Algorithms for Plane-Based Pose Estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2000)
32. Sukthankar, R., Stockton, R.G., Mullin, M.D., Smarter Presentations: Exploiting Homography in Camera-Projector Systems. In *Proceedings of International Conference on Computer Vision* (2001)
33. ViconPeak. <http://www.vicon.com/>
34. Weisser, M., The computer of the 21st century. *Scientific American* (1991)
35. Xu, L., Oja, E., Kultanen, P., A new curve detection method: Randomized Hough transform (RHT). In *Pattern Recognition Letters 11 (5)* pp. 331-338. (1990).
36. Zelnik-Manor, L., Peters, G., Perona, P., Squaring the Circle in Panoramas. In *Proceedings of ICCV'05*. (2005)
37. Zengyou, Z., 3D dynamic scene analysis: a stereo based approach. *Springer-Verlag* (1992)