**Proceedings of CVPRHCI 2003**
June 2003, Madison, Wisconsin, to appear

# The 4D Touchpad: Unencumbered HCI With VICs

Jason J. Corso, Darius Burschka and Gregory D. Hager
Computational Interaction and Robotics Laboratory
The Johns Hopkins University
Baltimore, MD 21218
{jcorso|burschka|hager}@cs.jhu.edu

***Abstract*— We present a platform for human-machine interfaces that provides functionality for robust, unencumbered interaction: the 4D Touchpad (4DT). The goal is direct interaction with interface components through intuitive actions and gestures.**

**The 4DT is based on the 3D-2D Projection-based mode of the VICs framework. The fundamental idea behind VICs is that expensive global image processing with user modeling and tracking is not necessary in general vision-based HCI. Instead, interface components operating under simple-to-complex rules in local image regions provide more robust and less costly functionality with 3 spatial dimensions and 1 temporal dimension. A prototype realization of the 4DT platform is presented; it operates through a set of planar homographies with uncalibrated cameras.**

Fig. 1. A picture of our first prototype 4DT.

## I. Introduction

In typical human-machine interfaces, we find many components — icons, buttons, scrollbars, etc. — with which we can interact. A conventional interface usually requires additional hardware devices to represent user's actions in a form comprehensible to a computer system. These devices restrict the pool of possible interactions to a subset that can be registered by the specific hardware device, e.g. two-dimensional movements of a computer mouse. The user does not interact directly with the visualized interface, but he/she observes a representation of his/her actions on the screen instead.

The goal of the 4D Touchpad (4DT) is the direct interaction with interface components through intuitive actions and gestures without any artificial tools or features. In our approach, a vision system passively monitors the scene while the user interacts directly with the interface. The vision system monitors the video-stream for a sequence of *visual interaction cues* (VICs): simple metrics when sequenced together provide more complex interaction capabilities (Sections II and III). A typical VICs-based HCI interface can be implemented as a 2D, 2.5D or 3D device [19]; in this paper, we demonstrate a *3D-2D Projection-based* system.

### A. Related Work

There exist many novel attempts to integrate computer vision into interface design. There has been a large effort to employ the tracking of human body motion, faces, and gestures with the general goal of supporting human-computer interaction [1], [2], [5], [11], [12], [16]. The Pfinder system [18] is a commonly cited example of a vision-based interface based on human-body tracking. These approaches typically require intensive computation that prohibits their use in a general interface setting.

There is also a broad range of table/board based systems that attempt to augment traditional physical environments with electronic means. The ZombiBoard [9] and BrightBoard [13] are examples of extensions of classical 2D "point-and-click" style user interfaces to desktop/blackboard style interactions. [4], [8], [17] are also examples of projects motivated by natural interaction with digital environments. Zhang et al's Visual Panel [21] enables the use of arbitrary planar surfaces as 3D input devices. In their work, they track the plane and a *tip pointer* (e.g. a fingertip) enabling its use in both conventional and new interaction modi. Kjeldsen et al's work on the Everywhere Display project [7] is similar to this work; it can also be considered a *3D-2D Projection-based* system. They have developed an approach that allows dynamic reconfiguration of vision-based interfaces.

In this paper we present a platform for human-machine interfaces that provides functionality for robust, unencumbered interaction. Our approach has similar goals to [17]: to integrate a desktop computer with unencumbered, intuitive interaction techniques (e.g. finger pointing). We call this platform the 4D Touchpad because it supplements 3D physical interaction with an additional temporal dimen-

sion. A picture of the first prototype system is shown in Figure 1.

In Sections II and III we review the VICs framework and introduce the stratified nature of robust interface design based on the VICs paradigm. In Section IV we present the 4DT and demonstrate an application built on this framework. Section V concludes and discusses possible extensions and enhancements to the VICs framework.

## II. The VICs Framework

The fundamental idea behind VICs is that expensive global image processing with user modeling and tracking is not necessary in general vision-based HCI settings. The basic interface component in the VIC framework is the VICon. A VICon is bound to a specific region in the interface and is restricted to process the image-stream in this local region; it has no knowledge of any information outside of its local region-of-interest (ROI). Additionally, each VICon defines a function-specific set of image processing components. The components are ordered in a *simple-to-complex* fashion such that each consecutive level of processing introduces a higher detection precision that is accompanied by an increased computational cost. Each level is only executed if the previous levels have validated the probable existence of an expected object in this ROI.

### A. The VICon

We define a VICs-based interface component (VICon) to be composed of three parts. First, it contains a visual processing engine. This engine is the core of the VICon as it defines the set of possible interaction behaviors associated with this VICon. Second, it has the ability to display itself to the user, and last, it generates some output triggers which are passed to higher level applications that use this interface.

The VICon does not rely on global tracking algorithms to monitor the user and detect actions. Instead, the VICon watches a region-of-interest (ROI) in the video stream and waits for recognizable user-input; i.e. interaction is site-centric. For instance, if we model a simple push-button, the VICon might watch for the sequence of cues the precede a button-push: motion, color-blob, and shape verification. This sequence of cues, ordered from simple-to-complex, is used to facilitate efficient, accurate user-input detection.

We define a *selector* to be a vision component that computes some measure on a local region of an image, and returns either nothing, indicating the absence of a cue or feature, or values describing a detected feature [6]. For example, a motion selector might return nothing if there is no apparent image motion or a description of the size and magnitude of a region of detected motion.

We define a *parser specification* to be a sequence of selector actions. Multiple parsers may exist for similar user-actions, but each may operate optimally under different circumstances: for example, in the presence of abundant lighting, a button-press parser may be defined as above: motion, color-blob, and shape verification. Yet, if the lighting is inadequate, a second button-press parser may be defined as motion, coarse edges, and shape verification. Thus, these two parsers provide similar functionality under different conditions thereby increasing the overall robustness of the system.

At its core, the visual processing engine of a VICon is a set of parsers. The composition of VICon's parsers is described in Section III.

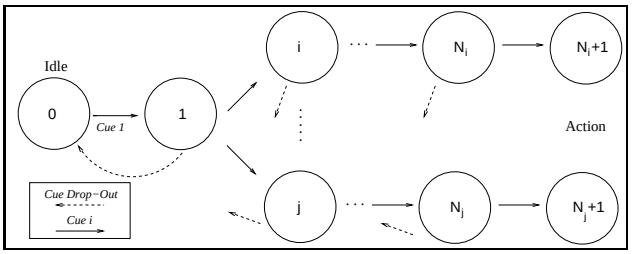


Fig. 2. The state model for a VICs-based interface component.

### B. Parser Modeling

Formally, we define a parser. It is modeled by a state machine with the following structure (Figure 2):

1 A finite set of discrete states $s_1, s_2, ...s_n$.
2 A distinguished initial state $s_1$ representing the idle condition.
3 Associated with each state $s_i$, a function $f_i$ comprised of a *bank* of selectors $b_i$ that defines a state variable $x$.
4 For each state $s_i$, a set of transition rules that associate an event $e_{i,j},\ j = 1 \ldots m \leq n$ (informally, the output of one or more selectors in the bank $b_i$) with either a state of a different index, or $s_i$ (the null-transition). By convention, the first transition event to fire defines the transition for that state.

We explain the parser modeling with the example of a button press VICon from above. We create a possible sequence of selectors: (1) a simple motion selector defines the trigger condition to switch from the distinguished initial state $s_1$, (2) a coarse color and motion selector, (3) a selector for color and cessation of motion, and (4) gesture recognition. It is easy to see that processing under this framework is efficient because of the selector ordering from simple-to-complex wherein parsing halts as soon as one selector in the sequence is not satisfied. We discuss the possibility of employing powerful parsing models in Section V.

### C. Modeling of Dynamics in VICons

The intent of the framework is that a parser will not only accept certain input, but it may return other relevant information: duration, location, etc. We define a VICon

history as a set $\{h_0 \dots h_m\}$, where the set length $m$ implicitly represents the duration of the current interaction and $h_{j\in\{1\dots m\}}$ is a snapshot of the behavior's current state and any additional relevant information. When a behavior enters its distinguished initial state, its history is reset: $m \leftarrow 0$. The history begins to *track* user's actions when the behavior leaves its initial state. A snapshot is created for every subsequent frame and concatenated into the history thereby adding another dimension that can be employed by the VICon during parsing.

The key factor differentiating the VICs paradigm from traditional interfaces is that there may be multiple exit conditions for a given VICon determined by different parsing streams each triggering a different output signal. The lexicon of possible output signals is an order of magnitude larger than WIMP[1] [15] and *super*-WIMP interfaces. We call a *super*-WIMP interface any interface that extends the traditional functionality of the mouse to include multi-button input or mouse-gesture input. One such example is the SKETCH framework [20].

## III. Stratified VICs

The components of human-machine interfaces typically require varying complexity in their specification. In the VICs framework, a VICon's parsing may be dependent on conditions exclusive to its own selectors' output. For instance, in a calculator type application, there may exist a function key that changes the parser of every VICon when triggered, or there may be a VICon measuring light conditions in the environment whose output affects the visual processing of other VICons.

Theoretically, these additional conditions could be incorporated into the parsers of a VICon, but this would increase the complexity of the state machines and render them difficult to create and maintain. We mentioned in Section II-A that a specific VICon may have several possible implementations with differing degrees of robustness that may respond to different conditions in the interface or the surrounding environment. Instead of overloading a VICon with these extra conditions, we extrapolate a stratification of VICons.

Such a stratification is realized through a control layer of VICons that manage system-wide variables like light-conditions and function-button type VICons. This does not exclude the application itself from setting conditions by which lower level VICons will be affected: for example, if a conventional keyboard is used, VICon processing may be affected by the state of the CAPS-LOCK key but the control layer VICons are not responsible for this.

In our approach, each VICon is comprised of a set of parsers (shown in gray in Figure 3), only one of which is *on* at any given instance, and each parser is modeled as described in Section II-B. The interface in Figure 3 consists of two application VICons (VICon2, VICon3) implemented with two alternative behaviors that are selected by the control-VICon (VICon1) or the application itself.

[1]WIMP is an acronym that means Windows, Icons, Menus, and Pointers.

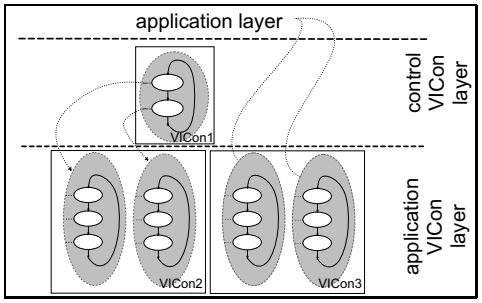


Fig. 3. Control hierarchy in a VICs system: *parser specification* of application VICons is switched based on conditions in higher layers.

The difference between a control VICon and an application VICon is how the output signals are used. While the output signals of the application VICons are returned to the application that uses the interface, the signals from the control VICons are used directly to switch parsers of the application VICons (gray ellipses in Figure 3). This functionality encapsulates the visual processing from the application: in normal situations, the application itself does not care about the light-conditions of the environment, but the application VICons do. Thus, this additional control layer simplifies the implementation of the top-level application by moving standard functionality, like adaptation to light condition, within the interface.

## IV. The 4D Touchpad: A VICs Platform

In this section, we introduce a VICs platform that has been constructed based on the *3D-2D Projection-based* interaction mode [19]. Here, a pair of cameras with a wide-baseline and a projector are directed at a table. The projector is placed underneath the table while the cameras are positioned above to remove user-occlusion in the projected images. This setup is shown in Figures 1 and 4.

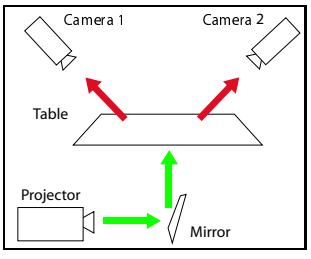


Fig. 4. The schematics of the 4DT.

### A. Image Rectification

In order for the visual processing and the graphical rendering of the interface to be simplified, the cameras

and the projector must be calibrated relative to the table. Since each VICon is processing only input in its local ROI, the optimal placement of a camera is parallel to the plane of the interface above the center of the table, while the projector is placed ideally below the table (removing all keystone distortion) to project exactly the same extent as the cameras are capturing. In practice, this is not possible because of physical imperfections and misalignments.

The solution to this problem lies in the use of a well known homography that maps points on a plane to their image [3]. Similar to [14], the assumptions made in the system are that the intrinsic and extrinsic parameters of the cameras are unknown and that the camera and projector optics can be modeled by perspective projection. The projection of a point in space on the image can be modeled with standard perspective projection:

$$\begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \quad (1)$$

Without loss of generality, we can say that the plane lies at $Z = 0$ yielding the following homography:

$$\begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = \begin{bmatrix} a & b & d \\ e & f & h \\ i & j & l \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}. \quad (2)$$

Numerous techniques exist for recovering this homography from points [14], lines [10], and other image features. The homography is used to solve the three problems mentioned above: (*i*) rectification of the two image streams thereby virtually placing the cameras directly above the table, (*ii*) keystone correction of the projected image, and (*iii*) projector-camera imaging area alignment. Thus, we have a two-step process that is now formalized; problems (*i*) and (*ii*) are solved in a manner that also satisfies (*iii*).

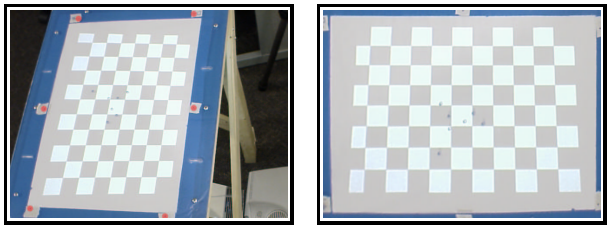

Fig. 5. (left) The original image from one of the cameras. Calibration points are shown as highlighted dots around the active area. (right) The same image after it has been rectified by $H_i$.

First, we compute $H_{i\in\{1,2\}}$ that rectifies a camera image into model space (Figure 5). Let $\hat{p}_{j\in\{1\ldots n\}}$ be an imaged point (shown as dots around the active area in Figure 5-left) and let $b_j$ be the corresponding model point in rectified space. In our context, we assume that the model is constructed by a set of known points that form a rectangle atop the table. For each camera, we can write

$$b_j = H_i \hat{p}_j, \quad i \in \{1,2\} \wedge j \in \{1 \ldots n\} \quad (3)$$

Then, we compute the projector homography $H_p$ that will keystone-correct the projected image and ensure projector-camera alignment. Let $q_{j\in\{1\ldots n\}}$ be a point in the projector image, and $\hat{q}_j^i$ be the corresponding point after it has been projected onto the table, imaged by camera $i$, and rectified by $H_i$. Thus, we define $H_p$ in a way that it maps $\hat{q}_j$ to the corresponding model point $b_j$.

$$b_j = H_p \hat{q}_j^i, \quad i \in \{1,2\} \wedge j \in \{1 \ldots n\} \quad (4)$$

This homography corrects any keystone distortion and aligns the projected image with the rectified camera images:

$$H_p \hat{q}_j^i = b_j = H_i \hat{p}_j, \quad i \in \{1,2\} \wedge j \in \{1 \ldots n\} \quad (5)$$

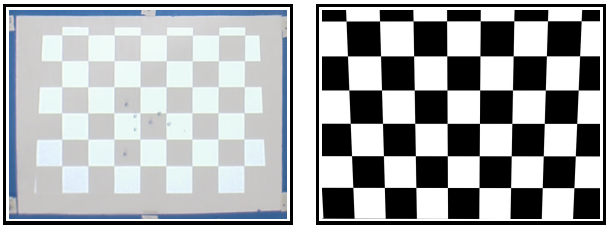

Fig. 6. (left) The projected image with the keystone correction applied (the distorted one is Figure 5-right). (right) The pre-warped image with the keystone correction applied before it has been projected.

### *B. Stereo Properties*

The vision system is responsible for the correct detection of press actions on the table and other gestures related to the VICons. Since the projection onto a camera plane results in the loss of one dimension, we use two cameras to verify the contact of the object with the surface of the table. The rectification process described in Section IV-A corrects both camera images in a way that all points in the plane of the table appear at the same position in both camera images. This can be used for simple stereo calculation. In practice, a small region above the surface is considered (Figure 7). Because of the rectification, a simple button press detector can be built just by segmenting color regions with a color skin detector and subtracting the resulting color blobs in both cameras from each other. The common region between the two images represents the part of the object that has actual contact with the plane of the interface. In Figure 9 we show a graph of the depth resolution of our system. The high depth discrimination is due to the wide baseline of the stereo system.

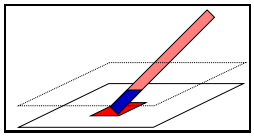

Fig. 7. Simple contact detection based on disparity.

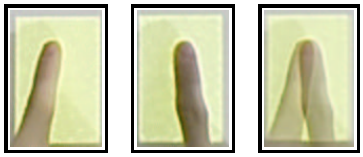

Fig. 8. Disparity for a typical press action: (left) rectified image 1, (middle) rectified image 2, (right) overlayed images of the finger.

Segmentation of objects above the plane is even easier. For both cameras, we can subtract the current frame from a stored background frame yielding mask of *modified* regions. Then, we can take the difference between two *modified* masks to find all pixels not on the plane and use it in more intensive computations like 3D gesture recognition. This method reduces also the influence of shadows that appear as part of the interface plane and get removed.

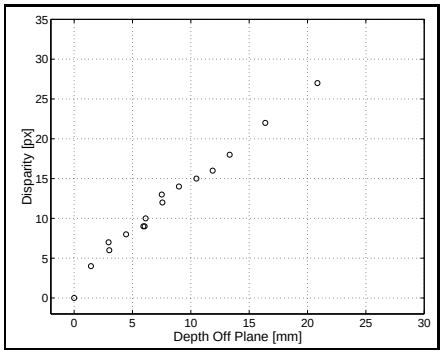


Fig. 9. Graph showing the depth resolution of the system.

### C. Experiments

To demonstrate the effectiveness of the VICs framework in our implementation of the 4DT, we include some experiments. The system runs on a LinuxPC with two CPUs operating at 1.4Ghz and 1GB of memory. The cameras are IEEE1394 Sony X-700; they are limited to 15Hz. However, we run the system in an asynchronous mode such that the VICons repeatedly process the same image until a new one is available. This allows multiple state transitions through the behaviors during one image-frame. Thus, our average processing rate is much higher than the input rate of the video-stream.

As discussed in Section II-A, each VICon processes a local region of the image in a simple-to-complex fashion. This efficient manner of processing allows many VICons to be incorporated into the interface without using too much computation. Table I shows the processor utilization for selectors of differing complexity (in this experiment, each selector was run independently and given full system resources).

| Complexity | Type | Rate [Hz] |
|---|---|---|
| Coarse | Motion Detection | 8300 |
| Medium | Hue Segmentation | 1000 |
| Fine | Hue Blob Intersection | 530 |
| Fine | Shape Template | 525 |

TABLE I — Simple-to-complex processing minimizes unnecessary computation. Rates for a 75x75 pixel region.

We built a virtual piano-keyboard application to demonstrate the capabilities of the 4DT and the VICs framework. Figure 10-left shows the system in use. We include 8 keys in the piano that are green while off and blue when pressed. When the system is set to a camera image size of 320x240, the asynchronous visual processing operates at a mean rate of about 250 Hz with nearly 100% processor utilization and it operates at a rate of about 45 Hz with 100% utilization when using 640x480 images. Experimentation showed that 320x240 images were sufficient for robust interaction detection.

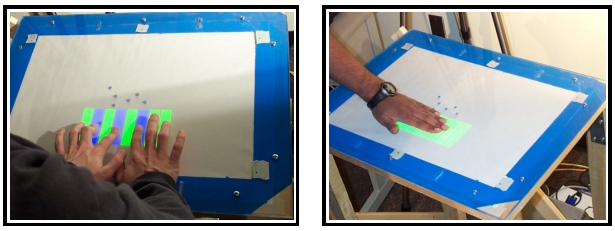

Fig. 10. (left) The 8 key VICs piano-keyboard. The user is pressing-down the 3 blue keys. (right)The 8 key VICs piano-keyboard employs a fingertip shape matcher at its finest detection resolution.*

The piano key VICons were modeled with the following 4 states: (1) motion detection in both images, (2) hue-blob presence in both image, (3) stereo shape-template matching, and (4) hue-blob presence. The piano key is considered pressed from the transition between states 3 and 4 until the transition from 4 to 1. Table II shows the accuracy rates of the system in usage; an informal experiment was performed with an unbiased onlooker judging user intention versus system response. Figure 10-right demonstrates the successful reduction of false-positives resulting from the implemented parser specification — the processing does not pass the shape-template stage (a fingertip pressing the key).

*The dots in the center of the work-surface are artifacts of the piece of Plexiglas used for the first prototype–they are in no way intended or used by the system.

| | Percent |
|---|---|
| **Correct** | 86.5 |
| **False Negative** | 11.45 |
| **False Positive** | 2.05 |

TABLE II — Accuracy of the piano keys to normal user input. Each figure is the mean of a set of users playing for a half minute. The camera image size was 640x480.

## V. Conclusion

We presented a new interface platform based on our previously proposed VICs framework [19]: the 4D Touchpad. The robustness of the processing and the marginal load on the host system for the idle processing of the VICs interface proves the usefulness of the simple-to-complex processing in a local ROI of each VICon. The framework is designed to maximize processing efficiency in order to allow the incorporation of vision-based HCI components into general interface designs.

The newly developed hardware setup with the projection from beneath the table-surface opens a variety of possible interfaces that would provide robust interaction without encumbering the user with equipment of any sort. A re-implementation of conventional interfaces like the keyboard and graphical tools would come first and move to new classes of interfaces exploiting the full capabilities of a true 4D interface framework. The advantage of the platform is obvious: the user directly interacts with the interface in an intuitive manner instead of indirectly interacting with the interface through their projection on the screen (e.g. a mouse pointer).

We intend to extend our work on behavior modeling to include more complex gestures not only in the plane of the surface but in the layers above: e.g. throwing an object into the trashcan by a simple movement of the hand above the VICon. We are also investigating more complex statistical-based methods to improve the robustness of the state-transitions in VICons. One possible technique is to train a Hidden Markov Model. In [19] we demonstrated a VICs-based system that employed an HMM to learn gesture dynamics. However, the basic tenet of HMMs is that the entire input vector is available for processing — i.e. the entire set of selectors in a VICon is processing every frame. This is completely opposite the simple-to-complex processing formulation of VICs. We are investigating some HMM modifications that would incorporate feedback from the current state to determine what processing must be performed.

## Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 0112882. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

The authors would like to thank Nicholas Ramey for performing the physical construction of the 4DT prototype.

## VI. References

[1] G. Bradski. Computer vision face tracking for use in a perceptual user interface. *Intel Technology Journal*, April 1998.

[2] Y. Cui and J. Weng. View-based hand segmentation and hand-sequence recognition with complex backgrounds. In *ICPR96*, page C8A.4, 1996.

[3] O. Faugeras and Q. Luong. *The Geometry of Multiple Images*. The MIT Press, 2001.

[4] Christophe Le Gal, Jérôme Martin, Augustin Lux, and James L. Crowley. Smartoffice: Design of an intelligent environment. *IEEE Intelligent Systems*, 16(4):60–66, 2001.

[5] D. Gavrila and L. Davis. Towards 3-d model-based tracking and recognition of human movement: A multi-view approach. In *Proc. Int. Conf. Automatic Face and Gesture Recognition*, 1995.

[6] G. Hager and K. Toyama. Incremental focus of attention for robust visual tracking. *International Journal of Computer Vision*, 35(1):45–63, November 1999.

[7] Rick Kjeldsen, Anthony Levas, and Claudio Pinhanez. Dynamically reconfigurable vision-based user interfaces. In *Proceedings of 3rd International Conference on Computer Vision Systems*, pages 323–332, 2003.

[8] M.S. Lee, D. Weinshall, E. Cohen-Solal, A. Colmenarez, and D. Lyons. A computer vision system for on-screen item selection by finger pointing. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1026–1033, 2001.

[9] Thomas P. Moran, Eric Saund, William Van Melle, Anuj U. Gujar, Kenneth P. Fishkin, and Beverly L. Harrison. Design and technology for collaborage: collaborative collages of information on physical walls. In *Proceedings of the 12th annual ACM symposium on User interface software and technology*, pages 197–206. ACM Press, 1999.

[10] Nassir Navab and Amnon Shashua. Algebraic derivations of relative affine structure and applications to 3d reconstruction from 2d views. Technical Report 270, M.I.T. Media Laboratory Perceptual Computing Section, March 1994.

[11] V.I. Pavlovic, R. Sharma, and T.S. Huang. Visual interpretation of hand gestures for human-computer interaction: A review. *PAMI*, 19(7):677–695, July 1997.

[12] J. Segen and S. Kumar. Fast and accurate 3d gesture recognition interface. In *ICPR98*, page SA11, 1998.

[13] Quentin Stafford-Fraser and Peter Robinson. Brightboard: a video-augmented environment. In *Conference proceedings on Human factors in computing systems*, pages 134–141. ACM Press, 1996.

[14] R. Sukthankar, R.G. Stockton, and M.D. Mullin. Smarter Presentations: Exploiting Homography in Camera-Projector Systems. In *Proceedings of Eighth International Conference on Computer Vision (ICCV)*, volume 1, pages 247–253, 2001.

[15] Andries van Dam. Post-wimp user interfaces. *Communications Of The ACM*, 40(2):63–67, 1997.

[16] Christian von Hardenberg and Franois Brard. Bare-hand human-computer interaction. In *Proceedings of Perceptual User Interfaces*, 2001.

[17] Pierre Wellner. Interacting with paper on the digitaldesk. *Communications of the ACM*, 36(7):87–96, 1993.

[18] C.R. Wren, A. Azarbayejani, T.J. Darrell, and A.P. Pentland. Pfinder: Real-time tracking of the human body. *PAMI*, 19(7):780–785, July 1997.

[19] Guangqi Ye, Jason Corso, Darius Burschka, and Gregory D. Hager. Vics: A modular vision-based hci framework. In *Proceedings of 3rd International Conference on Computer Vision Systems*, pages 257–267, 2003.

[20] Robert C. Zeleznik, Kenneth P. Herndon, and John F. Hughes. Sketch: an interface for sketching 3d scenes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 163–170. ACM Press, 1996.

[21] Zhengyou Zhang, Ying Wu, Ying Shan, and Steven Shafer. Visual panel: Virtual mouse keyboard and 3d controller with an ordinary piece of paper. In *Proceedings of Perceptual User Interfaces*, 2001.