Proceedings of ICVS 2003, pp. 257-267
April 2003, Graz, Austria

VICs: A Modular Vision-Based HCI Framework

Guangqi Ye, Jason Corso, Darius Burschka, and Gregory D. Hager

The Johns Hopkins University
Computational Interaction and Robotics Laboratory
cips@cs.jhu.edu

Abstract. Many Vision-Based Human-Computer Interaction (VB-HCI) systems are based on the tracking of user actions. Examples include gaze-tracking, head-tracking, finger-tracking, and so forth. In this paper, we present a framework that employs no user-tracking; instead, all interface components continuously observe and react to changes within a local image neighborhood. More specifically, components expect a pre-defined sequence of visual events called Visual Interface Cues (VICs). VICs include color, texture, motion and geometric elements, arranged to maximize the veridicality of the resulting interface element. A component is *executed* when this stream of cues has been satisfied.
We present a general architecture for an interface system operating under the VIC-Based HCI paradigm, and then focus specifically on an appearance-based system in which a Hidden Markov Model (HMM) is employed to learn the gesture dynamics. Our implementation of the system successfully recognizes a button-push with a 96% success rate. The system operates at frame-rate on standard PCs.

1 Introduction

The promise of computer vision for human-computer interaction (HCI) is great: vision-based interfaces would allow unencumbered, large-scale spatial motion. They could make use of hand gestures, movements or other similar input means; and video itself is passive, (now) cheap, and (soon) nearly universally available. In the simplest case, tracked hand motion and gesture recognition could replace the mouse in traditional applications. But, computer vision offers the additional possibility of defining new forms of interaction that make use of whole body motion, for example, interaction with a virtual character [17].

A brief survey of the literature (see Section 1.1) reveals that most reported work on vision-based HCI relies heavily on *visual tracking* and *visual template recognition algorithms* as its core technology. While tracking and recognition are, in some sense, sufficient for developing general vision-based HCI, one might ask if they are always necessary and if so, in what form. For example, complete, constant tracking of human body motion, while difficult because of complex kinematics [21], might be a convenient abstraction for detecting that a user's hand has touched a virtual "button," but what if that contact can be detected using simple motion or color segmentation? What if the user is not in a state where he or she is interacting at all? Clearly, we don't want to perform these operations except when needed, and then hopefully within a context that renders them reliable.

1.1 Related Work

The Pfinder system [29] and related applications [17] is a commonly cited example of a vision-based interface. Pfinder uses a statistically-based segmentation technique to detect and track a human user as a set of connected "blobs." A variety of filtering and estimation algorithms use the information from these blobs to produce a running state estimate of body configuration and motion [28]. Most applications make use of body motion estimates to animate a character or allow a user to interact with virtual objects.

More broadly, from the point of view of vision, there has been a great deal of interest in tracking of human body motion, faces, facial expression, and gesture, e.g. [2,10,23,5,30,8,3,19,16,7,4], with the general goal of supporting human-computer interaction.

From the HCI perspective, there have also been a wide class of "demonstration systems" that make use of vision as their input. The ZombiBoard [18] and BrightBoard [24] are examples of extensions of classical 2-D "point-and-click" style user interfaces to desktop/blackboard style interactions. They allow, for example, the selection, capture, or manipulation of items viewed by a video camera on a whiteboard or desktop. Input is usually via special written tokens; vision processing is based on simple background subtraction or thresholding followed by binary image processing, much as with Pfinder. More extensive proposals for mixing virtual and physical documents on the desktop include work on the Digital Desk [27] and on the "office of the future" [20]. A good example of a gesture-based interface is GestureVR [23].

It is clear that general-purpose vision tools for HCI is a nascent technology – systems are quite limited in scope, slow, or lack flexibility and robustness. In our work, we present a general architecture for an interface system operating under the VIC-Based HCI paradigm (Section 2). To the best of our knowledge, it is the first proposed general-purpose framework for vision-based HCI. We then focus on an appearance-based system using a Hidden Markov Model to learn user-input dynamics. This system operates under the VIC paradigm.

2 The VIC Paradigm

2.1 Modeling Interaction

Current interface technology, Windows-Icons-Menus-Pointers (WIMP) [26], is modeled with a simple state-machine (Figure 1). The dominant interface component in these *third-generation interfaces* is the *icon*. Typically, these icons have one pre-defined action associated with them that is triggered upon a mouse click.

We extend the functionality of a traditional icon by increasing its number of associated actions that can be triggered by the user. For standard WIMP

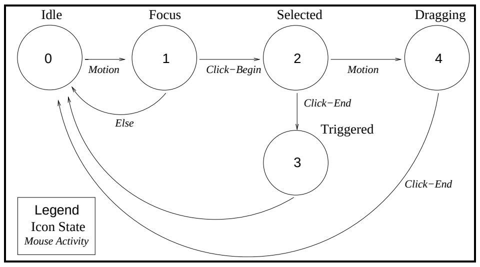

Fig. 1. The icon state model for a WIMP interface.

interfaces the size of this set is 1: point-and-click. For *super*-WIMP[1] interfaces, the size of this set is larger, but still relatively small; it is limited by the coarse nature of mouse input. Our vision-based extension greatly increases the set of possible user inputs. To allow for such an extension, the notion of the icon must change: we define a VIC-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 replaces the current point-and-click nature of third-generation interfaces. Second, it has the ability to display itself to the user, and lastly, it has some application specific functionality.

As mentioned earlier in Section 1, the VICon does not rely on 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. For instance, if we model a simple push-button, the VICon might watch for something that resembles a human-finger in its ROI.

The obvious approach to detect user interaction is one of template-matching in which the VICon is aware of a set of possible gestures and uses image processing techniques to analyze the ROI in every frame of video. However, in practice, such a method is prone to false-positives by spurious template matches. Also, a template matching approach, alone, is potentially wasteful because it is more expensive than other simpler tasks like motion detection and color segmentation that may easily indicate a negative match.

If one observes the sequence of cues that precede a button-push, for instance, one notices that there are distinct stages preceding the actual button push: motion, color-blob, rough edges. This sequence of cues, ordered from simple to complex, can be used to facilitate efficient, accurate user-input detection. 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 [11]. 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. Thus, at

[1] 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 [31].

its core, the visual processing engine of the VICon is a sequence of selectors: we call it a *visual interaction cue parser* or just a parser.

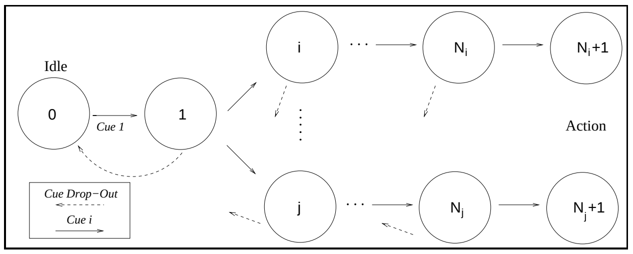

Fig. 2. The state model for a VIC-based interface component.

Formally, we define a visual interaction cue parser (Figure 2). It is a component with the following structure:

1 A finite set of discrete states $s_1, s_2, ...s_n$.
2 A distinguished initial state s_1.
3 Associated with each state s_i, a function f_i on the incoming input stream that defines a continuous state variable x.
4 For each state s_i, a set of transition rules that associates an event $e_{i,j},\ j = 1 \ldots m \leq n$ (informally, the output of a selector) with either a state of higher index, or s_1. By convention, the first transition event to fire defines the transition for that state.

We return to the example of a button push from above. Using a parser, we create a possible sequence of selectors: (1) a simple motion selector, (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. More powerful and sophisticated parsing models are plausible under this paradigm: an example showing the use of a Hidden Markov Model is presented in Section 3.

The intent of the framework is that a parser will not only accept certain input, but might return other relevant information: location, duration. The key factor differentiating the VIC paradigm from traditional interface components is that there may be multiple exit cases for a given VICon determined by different streams through the parser each triggering a different event. The lexicon of possible triggers is an order of magnitude larger than WIMP and *super*-WIMP interfaces.

2.2 Interaction Modes

The notion of a VIC-based interface is broad and extensible to varying application domains. In this section we enumerate the set of interaction modes in which a VICon may be used.

1. **2D-2D Projection** - Here, one camera is pointed at a workspace, e.g. tabletop. One or many projectors are used to project interface components onto this surface while the video-stream is processed under the VIC paradigm. This mode has been proposed in [32]. We feel incorporating VIC-based interface components will increase is effectiveness and broaden the domain of applications.
2. **2D-2D Mirror** - In this mode of interaction, one camera is aimed directly at the user and the image stream is displayed in the background of the user-interface for the user. Interface components are then composited into the video stream and presented to the user. This interface mode could also be used in a projection style display to allow for a group to collaborate in the shared space.
3. **3D-2D Projection** - This mode is similar to the first (2D-2D Projection) except that 2 or more cameras will be aimed at the workspace and the set of possible selectors is increased to include more robust 3d geometry.
4. **2.5D Augmented Reality** - Both video-see-through and optical-see-through augmented reality are possible if the user(s) wear stereo head-mounted displays (HMD) [1]. With stereo cameras mounted atop the HMD, knowledge of a governing surface can be extracted from the view, e.g. planar surface [6]. All VICons can then be defined to rest on this governing surface and interaction is defined with respect to this surface. One possible application is a piano where each key is a separate VICon.
5. **3D Augmented Reality** - In this case, we remove the constraint that the interface is tied to one governing surface and allow the VICons to be fully 3D. An example application would be a motor-function training program for young-children in which they would have to organize a set of blocks whose shapes and colors differ according to some rubric.

2.3 Prior State of the Art in VIC Technology

In this section we show a small set of example interfaces built under the VIC paradigm: interaction through a stream of local-based selectors. First, we show a simple button-based VICon in a calculator setting (Figure 3-left). In this case, the VICon used a motion-based cue, a color-segmentation cue, and enforced that the color remain present for a static time-interval. Next, we show multiple triggers based on user-input (Figure 3-middle). Here, the user can select the ball, drag it, and release. The parser incorporates a simple-gesture recognition stage; it's state-model follows Figure 2. As mentioned earlier, motion and dynamics can be added to the VICons. Figure 3-right shows a *Breakout*^{TM} like program where the ball is a VICon. During play, the ball, the VICon, travels through the workspace. The user attempts to prevent the ball from falling through the bottom of the workspace while deflecting it toward the colored bricks at the top of the workspace; notice the VICon is not anchored.

The previous three VIC-based interface examples employ the 2D-2D Mirror mode of operation. Our current focus is the 2.5D Augmented Reality mode of operation. We have developed a set of fast surface recovery techniques [6]

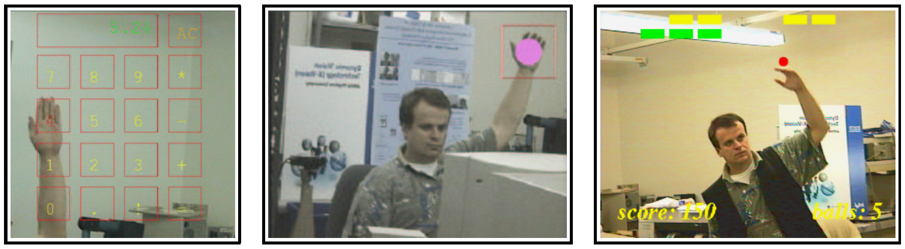

Fig. 3. (left) A VIC-based calculator using a motion-color parser. (middle) Gesture-based demonstration of multiple interaction triggers for a single VICon. (right) VIC-Based 2D-2D mirror mode interface for a *Breakout*^{TM} style game.

allowing us to anchor the interface to a planar surface. In the next section, we present an extension of the parsers presented above through the incorporation of background-foreground modeling and stochastic parsing.

3 Focus: A Stochastic VICon via HMM

We designed and implemented a real-time VICs-based interaction system to identify a button-pushing action. This module can be easily incorporated into a larger system that allows the user to interact with the computer through gesture and finger movement. We use a static camera to supervise a virtual button, which is represented by a graphical icon. The user is expected to move his finger toward the button and stay on the button for a short period of time to trigger it. The system will decide whether the user has triggered the button. Thus, fast and robust foreground segmentation and action recognition are two key elements of our system.

3.1 Background Modeling and Image Segmentation Based on Hue Histogram

Background subtraction, gray-scale background modeling [12], color appearance modeling [25], color histogram [15] and combining of multiple cues [22] are among the most widely used methods to model the background and perform foreground segmentation. We propose to use a hue histogram for two reasons: speed and relative color invariance. This scheme employs a very fast on-line learning process, which is an advantage for this specific application since the area surrounding the button may change between sessions. Furthermore, hue is a good color invariant model that is relatively invariable to translation and rotation about the viewing axis, and changes slowly under change of angle of view, scale and occlusion [9].

We assume that the background is static for a given session. We split the background image into an array of equal-sized sub-images. For each sub-image, we build a hue histogram to model it. We process the foreground image in a similar way and perform pairwise histogram matching between background and foreground image histograms. Here, we employ histogram intersection [25] as the comparison criterion.

$$H(I, M) = \frac{\sum_{j=1}^{n} min(I_j, M_j)}{\sum_{j=1}^{n} M_j} \tag{1}$$

Here I and M refer to model and measure histogram respectively. If the matching value is below the threshold, which is determined empirically, the corresponding image region is classified as foreground; otherwise, it is background. Our experiments show that combining hue color model and histogram intersection can achieve relative invariance to illumination changes and obtain good segmentation results. After employing a median filter on this binary image to reduce possible noise, we perform the segmentation on the original image according to the identity of each region. Figure 1 shows an example.

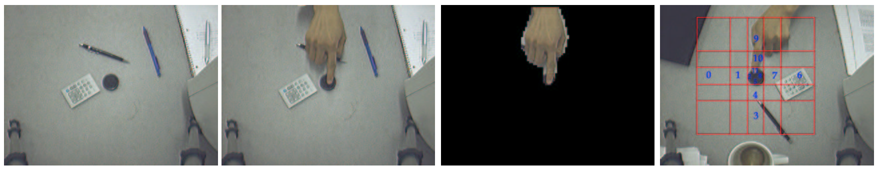

Fig. 4. An example of background image, unsegmented image and segmentation result. The leftmost image is the background. The second image shows when the hand has entered the scene. The final segmentation result is shown in the third image. The last image demonstrates our feature space definition.

3.2 HMM-based Human Activity Recognition

In our experiment, we employ a simple HMM [13] [14] to train and recognize the button-pushing action. The basic idea is to define a finite feature space onto which the image is mapped. Then based on captured training image sequences, we can construct HMMs for each class of actions and train them using the Baum-Welch[14] algorithm. The probability that each HMM generates the given feature sequence is the criterion of recognition.

We propose a computationally efficient and robust feature extraction scheme. This feature indicates the direction and distance of the finger from the center of the button. In principle, we split the contiguous region of the button into a 5 by 5 grid. According to the segmentation result, we can tell whether a certain cell is foreground or background. By comparing the number of cells touched by the hand in each direction, we know from which direction the hand is coming. It's also easy to tell the distance of the finger to the button by checking the nearest cell covered by the hand. Combination of all possible direction and distance forms our feature space.

For each feature state, we define a basic HMM to represent it. And for each of the four classes of actions (i.e., pushing from up, down, left and right, respectively), we will find a representative sequence. Based on this standard sequence, we build the HMM for this class by concatenating, with null transitions, all the basic HMMs corresponding to each symbol in the sequence.

Since it is difficult to capture all possible patterns of non-pushing actions, we use a threshold on the highest possibility of the classes to perform rejection. However, the duration of the action may vary significantly and thus the possibilities that each class generates such a sequence, even though the action pattern is still the same. To overcome this time variation, we perform sequence aligning in training and recognition. That is, we choose a fixed length, for example, 20 frames, to be the standard duration. For any sequence longer than this, we resample the sequence to get a new sequence with standard length. We will discard those sequences that are shorter than standard length.

4 Experiment Results

In our current experiment, we use a color camera with image size of 640×480 as the imaging sensor. The system can achieve a frame rate of about 10 fps on a Pentium III PC. If we reduce the resolution to 320×240, the system can run at over 20 fps.

4.1 Background Modeling and Segmentation Result

To test our segmentation scheme, we captured image pairs of the background and foreground. By comparing the segmentation result and the ground-truth classification image, which is generated by manually marking the foreground part of the scene, we are able to evaluate this algorithm. We captured more than 20 pairs of background/foreground images with different background scenes and carried out the experiment on these images. The test set also includes 6 pairs of images that undergo illumination changes. As a result, the average correct ratio is 98.16%, with average false positive ratio of 1.55% and false negative ratio of 0.29%.

We also compare the segmentation result with different sub-window size and with different number of bins of the hue histogram. The result shows that histograms with at least 8 bins perform better than those with less, while increasing the bins to 16 or more does not bring any performance enhancement. Figure 5 shows the relationship between segmentation result and size of sub-images. It can be seen that for a histogram with only 4 bins, the more samples, the better the result. While with 8 bins, the correct ratio and false positive doesn't change much. For both cases, false negative ratio increases with the tile size.

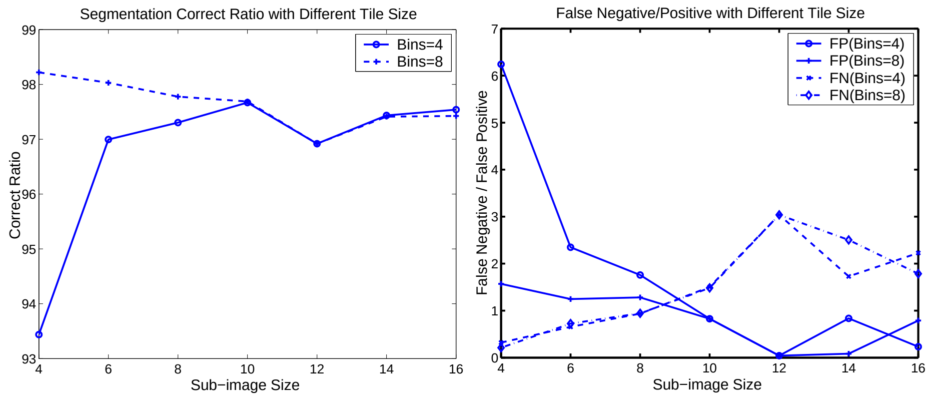

Fig. 5. Segmentation correct ratio/false positive/false negative with different sub-image size.

4.2 Action Recognition Result

For training and testing of our HMMs, we recorded over 300 action sequences by 6 different people, 76 of them used for training. An offline procedure is carried out to find the best characteristic sequence for each class. After training, the system can achieve a correct ratio of 100% on the training set. We tested the system on a set of 277 well-segmented sequences, including both valid and invalid button-triggering actions. The length of these sequences varies significantly, ranging from 30 to over 220. Our test set includes some sequences with illumination changes, which are also segmented successfully. The overall correct ratio on this test set is 96.8%. The result demonstrates the robustness and correctness of our system.

The standard length of the category characteristic sequence will influence the system performance and speed. Along with the increase of the size of primary sequence, the time needed to carry out the recognition will also grow linearly. However, since a longer sequence contains more information and thus, has a larger HMM, the total system performance will improve. The following table shows the experimental results with category primary sequences of different sizes.

Table 1. Experiment results with different length of characteristic sequence

L	Average fps	Accuracy of Training Set	Accuracy of Test Set
10	10.3	100.0%	86.8%
20	10.0	100.0%	94.2%
30	9.8	100.0%	96.8%

5 Conclusion

We have introduced the VICs approach to vision-based interaction. VICs stems from our experience using locally activated iconic cues to develop simple vision-driven interfaces. In particular, we have identified two central problems to be solved: developing reliable foreground-background disambiguation, and incorporating dynamics into gestures. We have shown that, given good solutions to the former problem, the latter can be addressed using standard HMM techniques.

Our immediate goal for the VICs project is to create 2.5D surface-anchored interfaces. To this end, we have developed a set of fast surface recovery techniques to place two rectified images in correspondence [6], and we are currently extending the results reported in this paper to a two-camera system. In the latter case, the HMM input will be data from both images, and the goal will be to recognize that the user is pressing a button as if it appears on the underlying surface.

References

1. R. Azuma. A survey of augmented reality. *Presence: Teleoperators and Virtual Environments 6*, pages 355–385, 1997.
2. S. Basu, I. Essa, and A. Pentland. Motion regularization for model-based head tracking. In *Proc. Int. Conf. Pattern Recognition*, 1996.
3. M.J. Black and Y. Yacoob. Tracking and recognizing rigid and non-rigid facial motions using local parametric models of image motion. *Int. J. Computer Vision*, 25(1):23–48, 1997.
4. G. Bradski. Computer vision face tracking for use in a perceptual user interface. *Intel Technology Journal*, April 1998.
5. C. Bregler and J. Malik. Tracking people with twists and exponential maps. In *Proc. Computer Vision and Pattern Recognition*, pages 8–15, 1998.
6. Jason Corso and Gregory D. Hager. Planar surface tracking using direct stereo. Technical report, The Johns Hopkins University, 2002. CIRL Lab Technical Report.
7. Y. Cui and J. Weng. View-based hand segmentation and hand-sequence recognition with complex backgrounds. In *ICPR96*, page C8A.4, 1996.
8. 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.
9. Theo Gevers. Color based object recognition. *Pattern Recognition*, 32(3):453–464, 1999.
10. L. Goncalves, E. Di Bernardo, E. Ursella, and P. Perona. Monocular tracking of the human arm in 3-d. In *Proc. Int. Conf. Computer Vision*, pages 764–770, 1995.
11. G. Hager and K. Toyama. Incremental focus of attention for robust visual tracking. *International Journal of Computer Vision*, 35(1):45–63, November 1999.
12. Thanarat Horprasert, David Harwood, and Larry S. Davis. A robust background substraction and shadow detection. In *Proc. ACCV'2000, Taipei, Taiwan*, January 2000.
13. K. Ishii J. Yamota, J. Ohya. Recognizing human actions in time-sequential images using hidden markov model. In *IEEE Proc. CVPR 1992, Champaign, IL*, pages 379–385, 1992.

14. Frederick Jelinek. In *Statistical Methods for Speech Recognition*, MIT Press, 1999.
15. Michael J. Jones and James M. Rehg. Statistical color models with application to skin detection. *International Journal of Computer Vision*, 46(1):81–96, 2002.
16. R. Kjeldsen and J.R. Kender. Interaction with on-screen objects using visual gesture recognition. In *CVPR97*, pages 788–793, 1997.
17. P. Maes, T.J. Darrell, B. Blumberg, and A.P. Pentland. The alive system: Wireless, full-body interaction with autonomous agents. *MultSys*, 5(2):105–112, March 1997.
18. T. Moran, E. Saund, W. van Melle, A. Gujar, K. Fishkin, and B. Harrison. Design and technology for collaborage: Collaborative collages of information on physical walls. In *Proc. ACM Symposium on User Interface Software and Technology*, 1999.
19. 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.
20. R. Raskar, G. Welch, M. Cutts, A. Lake, L. Stesin, and H. Fuchs. The office of the future: A unified approach to image-based modeling and spatially immersive displays. In *Proc. SIGGRAPH*, 1998.
21. J.M. Rehg and T. Kanade. Visual tracking of high DOF articulated structures: An application to human hand tracking. In *Computer Vision – ECCV '94*, volume B, pages 35–46, 1994.
22. Christopher Richard Rwen, Ali Azarbayejani, Trevor Darrell, and Alex Paul Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence 19(7)*, 19(7):780–784, 1997.
23. J. Segen and S. Kumar. Fast and accurate 3d gesture recognition interface. In *ICPR98*, page SA11, 1998.
24. Quentin Stafford-Fraser and Peter Robinson. Brightboard: A video-augmented environment papers: Virtual and computer-augmented environments. In *Proceedings of ACM CHI 96 Conference on Human Factors in Computing Systems*, pages 134–141, 1996.
25. M. J. Swain and D. H. Ballard. Color indexing. *International Journal of Computer Vision 7(1)*, pages 11–32, 1991.
26. Andries van Dam. Post-wimp user interfaces. *Communications Of The ACM*, 40(2):63–67, 1997.
27. Pierre Welner. Interacting with paper on the digital desk. *Communications of the ACM*, 36(7):87–96, 1993.
28. C. Wren and A. Pentland. Dynamic modeling of human motion. In *Proc. Int. Conf. Automatic Face and Gesture Recognition*, 1998.
29. 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.
30. M. Yamamoto, A. Sato, and S. Kawada. Incremental tracking of human actions from multiple views. In *Proc. Computer Vision and Pattern Recognition*, pages 2–7, 1998.
31. 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.
32. Zhengyou Zhang, Ying Wu, Ying Shan, and Steven Shafer. Visual panel: Virtual mouse keyboard and 3d controller with an ordinary piece of paper. In *Workshop on Perceptive User Interfaces.* ACM Digital Library, November 2001. ISBN 1-58113-448-7.