

R. G. O'Hagan

*Department of Computer Science, The Australian National University, Canberra,
ACT 0200, Australia*

A. Zelinsky, S. Rougeaux

*Department of Systems Engineering, Research School of Information Sciences &
Engineering, The Australian National University, Canberra, ACT 0200, Australia*

Contact Addresses:

R. G. O'Hagan

Department of Systems Engineering,

Research School of Information Sciences & Engineering,

The Australian National University

Canberra ACT 0200

Australia

Tel. +61 2 6125 9659

Fax. +61 2 6216 7111

Rochelle.OHagan@syseng.anu.edu.au

Prof. A. Zelinsky

Department of Systems Engineering

Research School of Information Sciences & Engineering,

The Australian National University

Canberra ACT 0200

Australia

Tel. +61 2 6125 8686

Fax. +61 2 6125 8660

Alex.Zelinsky@syseng.anu.edu.au

Dr. S. Rougeaux

Department of Systems Engineering

Research School of Information Sciences & Engineering,

The Australian National University

Canberra ACT 0200

Australia

Tel. +61 2 6125 8684

Fax. +61 2 6125 8688

Visual Gesture Interfaces for Virtual Environments^{*}

Abstract

Virtual environments provide a whole new way of viewing and manipulating 3D data. Current technology moves the images out of desktop monitors and into the space immediately surrounding the user. Users can literally put their hands on the virtual objects. Unfortunately techniques for interacting with such environments have yet to mature. Gloves and sensor-based trackers are unwieldy, constraining and uncomfortable to use. A natural, more intuitive method of interaction would be to allow the user to grasp objects with their hands and manipulate them as if they were real objects.

We are investigating the use of computer vision in implementing a natural interface based on hand gestures. A framework for a gesture recognition system is introduced along with results of experiments in colour segmentation, feature extraction and template matching for finger and hand tracking, and simple hand pose recognition. Implementation of a gesture interface for navigation and object manipulation in virtual environments is presented.

Key words: gesture recognition, virtual environments, computer vision, interaction

^{*} This work was supported by the Co-operative Research Centre for Advanced Computational Systems (ACSys).

1 Introduction

1.1 Overview

The advent of virtual environments allows the creation of 3D objects and worlds in which the user is immersed. Scientists, engineers, doctors and architects, among others, can now visualise complex structures and systems with high degrees of quality and realism. Shutter glasses provide a stereo or 3D view of the scene, which is no longer confined to a desktop monitor, but may be a large table, projection screen or room. Virtual environments also offer the opportunity to interact with data in ways unavailable using traditional computing or physical tools. Unfortunately, the full potential of this opportunity is currently unrealised with most systems available today providing what is effectively little more than a single-user “point and click” interface in three dimensions.

Virtual environments attempt to create a world where the interaction feels real. In many ways, the user’s ability to interact with the pictures is as important as the quality of the pictures themselves (Figueiredo and Boehm, 1993). While current mechanical, acoustic and magnetic input devices track the user and allow control of movement, selection and manipulation of objects in virtual scenes, these interactions are often limited and unintuitive and the devices awkward, unwieldy and prone to distortion from the physical environment. We are interested in developing an alternative, natural interface that more closely models the way we interact with the real world. The user should be able to reach out, grab, point and move 3D objects just as we do with real objects.

We are investigating techniques for vision-based gesture recognition and have developed a framework for a vision-based gesture interface to projection table-

based virtual environments. Our system allows the user to manipulate objects within the environment in a natural manner by allowing the user to point at and grasp virtual objects as they would real objects. Manipulations include selection, translation, rotation and resizing of objects and also changing the viewpoint of the scene (eg. zooming in or out). Additionally, the system allows the user to navigate or “fly through” 3D data. This paper reports on work in progress in the development of a vision based gesture interface for navigation and manipulation of 3D objects within a virtual environment.

1.2 Background

In order to achieve immersion within a virtual environment, the user must be able to interact effectively with the virtual world. By *effectively* we mean minimising cognitive load and maximising goal success. In order to achieve this, a successful 3D user interface should be natural, intuitive or at least easy to learn, and powerful enough to allow the user to accomplish the required tasks. In this definition, we use the terms *natural* and *intuitive* to imply that the interaction metaphor should inherently make sense to the user. This may mean that interaction mimics real world processes, or an alternative method could be used that utilises the extended capabilities offered by virtual environments.

To achieve such an interface, we are faced with a set of basic requirements. Firstly, interaction response should be fast. In order for the user to maintain a sense of “presence” or immersion, the system must respond to the user’s input in real-time. For a typical virtual environment system, the desired graphical refresh rate is 30Hz. When the refresh rate drops much lower than this, the sense of being immersed in the world reduces. To ensure that interaction doesn’t lag behind the display, the response to interaction should also be in the order of

30Hz to be classed as “real-time”.

Secondly, interaction should be precise so that what the user believes themselves to be interacting with, is what the interface believes they are interacting with. This becomes even more important with true direct manipulation interfaces where the graphics are co-located with the interaction space. If the mapping between the user’s interaction space and the virtual world is imprecise then the interaction process breaks down completely and the sense of immersion is lost.

Finally, cognitive load on the user should be minimised. One way this can be achieved is through direct manipulation interfaces. Direct manipulation interfaces feature a natural representation of objects and actions to hide the feeling of performing tasks through an intermediary (the computer). The underlying belief is that allowing the user to directly perceive and interact with the virtual objects will lead to a more natural and effective interface.

By identifying basic interaction tasks and implementing them in a natural way, improvements should be seen in the usability and effectiveness of interaction with the virtual environment. Hand (1997) and Bowman and Hodges (1999) identify similar sets of universal tasks for virtual environments; namely object selection, manipulation, and viewpoint control. Viewpoint control refers to users interactively positioning and orienting their viewpoint within the scene. Since head tracking is usually used to determine the orientation of the viewpoint, this task primarily concerns translation of the viewpoint or moving to various positions within the scene, including zooming in or out. Selection is the task of specifying an object or objects for some purpose including selection of items for application control (eg. selecting menu items). Manipulation refers to the positioning and/or orienting of a virtual object or objects. Se-

lected objects may be manipulated in space, or selected for another purpose such as alteration of object attributes (eg. colour, texture), resizing, deletion or editing.

Input into virtual environments typically consists of two components: a means of tracking the user's head, hands or whole body to allow the user to specify translations and rotations; and extra controls such as buttons for selection, mode changes and actions. Tracking systems are available using a variety of technologies ranging from mechanical devices to magnetic, acoustic or optical trackers. Tracking devices are usually limited to some working volume determined by the range of the device, and most require a physical connection between the user and the system, usually in the form of a sensor or receiver and associated cabling. The general principle is to have a transmitter fixed somewhere in the environment, and a receiver attached to the user. The receiver uses the signal to calculate its position and orientation relative to the transmitter. This information is input to the computer and used to compute the location in system coordinates.

The limitations of being physically connected to the system constrain the ability of the user to move naturally within the environment and often induce awkward motions to achieve the desired manipulation. With the exception of gloves, the devices are all suited to tracking a single object. In order to track the various parts of an articulated object such as the hand, multiple sensors are necessary. Glove-based trackers can provide information about the various joint positions which is useful in determining hand poses, however obtaining this data without requiring the glove or cabling would be more comfortable and would also allow multiple users to move in and out of the environment without having to share specific equipment.

In contrast to the above interaction tools, vision-based gesture recognition provides a natural interface to virtual environments that combines the advantages of glove-based devices with the unobtrusive nature of vision-based tracking. Users can use their hands to manipulate virtual objects without being connected to the system in any way. Additionally, gesture-based input allows a higher bandwidth connection between the user and the system than traditional tracking devices. Instead of being constrained by the “point and click” metaphor for interaction, new metaphors can be explored. Contextual task information can be obtained from the user’s hand pose allowing a modeless interface to be developed. For example, rather than choosing tools from a menu or palette onscreen, different gestures could be used to indicate the desired action. This would remove the necessity for menus and toolbars cluttering up the workspace, simplifying the display presented to the user. Of course careful consideration must be given to the choice of a gesture set to ensure that the cognitive load on the user is not *increased* by the need to remember specific gestures, but reduced through the use of intuitive hand movements.

1.2.1 *Related Work*

The idea of using hand gestures as a means of interacting with computers has been around for some time. The first notable system was Bolt’s “Put That There” multimodal interface (Bolt, 1980). Bolt combined speech recognition with pointing to move objects within the scene. Other systems have since been developed for applications such as sign language recognition, virtual mice and control of 2D interfaces. Few, however, have been applied to interaction with 3D objects. Although glove-based gestural interfaces have been developed, it was not until very recently that useable vision-based gesture recognition systems were implemented.

Sharma et al. (2000) report on a multimodal interface combining speech and gesture to manipulate 3D virtual objects. The interface allows gestures for selecting objects and to aid other basic manipulation tasks such as “forward”, “back”, “up” and “left”. Commands can also be spoken. Our system differs from this by providing a means of *directly* manipulating the objects themselves rather than through commands.

Azarbayejani et al. (1996) developed a human body tracking system. Using stereo cameras the system tracks gross motions of the user’s head, hands and feet at 10-30Hz. The system has been applied to sign language recognition and control of animations and avatars rather than 3D object manipulation tasks. In an extension to this work, Wren (2000) applies knowledge of human kinematics to improve the accuracy of the tracking and to handle occlusions.

Segen and Kumar (2000) have recently reported developments with a system for manipulating 3D objects. The system uses stereo cameras with a plain background and stable illumination to track the thumb and forefinger tips of a pointing hand. This can then be used to control navigation around a 3D scene. Three simple gestures are also recognised allowing translation and rotation of 3D objects.

Although similar, our system focuses on tracking within minimally structured environments where conditions are open to change. This includes cluttered backgrounds and changing illumination. In order to improve the robustness of the system to temporary or partial occlusions, and to obtain stable and accurate position and orientation estimates, we track multiple points on the hand. By combining feature-based tracking with an underlying model, we obtain a robust and reliable system for tracking.

2 Vision-based Gesture Recognition

An effective vision-based gesture recognition system for virtual environments must accomplish two main tasks. First, the position and orientation of the hand in 3D space must be determined in each frame. Second, the hand and its pose must be recognised and classified to provide the interface with information on actions required. In other words, the hand must be tracked within the work volume to give positioning information to the interface, and gestures must be recognised to present the meaning behind the movements to the interface. Due to the nature of the hand and its many degrees of freedom, these are not insignificant tasks. Additionally, these tasks must be executed as quickly as possible in order to obtain a system that runs at close to frame rate (30Hz). The system should also be robust so that tracking can be re-established automatically if lost or if the hand moves momentarily out of the working area.

2.1 Feature-based vs Model-based Systems

Computer vision systems for tracking and recognising objects typically take one of two approaches. Some systems rely on 3D models of the object to be tracked. In the case of the hand, this means having an underlying model of the structure of the hand as in Rehg and Kanade's DigitEyes (Rehg and Kanade, 1993), or the articulated hand model by Shimada et al. (1998). Joint angle and position information is extracted from images and used to update the model. The model is also used to constrain image interpretation and reduce the effects of noise in tracking. Other models have been developed using 3D Point Distribution Models as in work by Heap and Hogg (1996). Unfortunately, model-based approaches tend to high levels of complexity and computational

inefficiency and are therefore difficult to implement in real-time. They are also susceptible to variation and would generally need to be re-calibrated for each user.

The second approach to solving the gesture recognition problem is appearance-based techniques. These systems rely directly on information extracted from video images. The information extracted may be in the form of geometric properties of hand poses, silhouettes or boundaries. Alternatively, appearance models may be template based, where the image or parts of it, are correlated with previously stored reference images. Image or geometric moments are used in several systems to recognise and classify gestures. Pavlovic et al. (1996) use this method in a gesture interface to a 3D molecular biology visualisation in a virtual environment. The system allows the user to control a 3D pointer and recognises a set of gestures for giving commands. In similar work by Segen and Kumar (1998, 2000), boundary analysis is used to determine and track the position of fingertip and thumb. Appearance-based approaches have been quite successful in classification tasks and for providing basic 3D position information. They do, however, suffer from difficulties in tracking features that deform or are obscured.

We believe that a combination of these two approaches will enable the development of a robust and reliable real-time tracking and recognition system. In the following section we describe the framework we have developed for such a system and the results of applying this framework to the problem of interacting with a medium-scale virtual environment.

2.2 A Framework for Vision-based Gesture Recognition

As noted above, the gesture recognition task can be thought of as two problems: determining the position and orientation of the hand or its parts; and identifying particular gestures or movements within the application context. Solving these problems involves a process which, in each frame, identifies the hand, extracts the relevant features, estimates the position and orientation, classifies the pose and updates the application. In order to accomplish these tasks, our gesture interface system follows the framework shown in Figure 1. Detailed explanations of each step in the process are given below.

2.3 System Setup

The system setup is shown in Figure 2 and is based on a Barco Baron projection table. The Barco Baron projection table provides a virtual working environment of approximately 1.5m^3 , and can be configured as a horizontal table, a vertical wall display, or something in between. The user views images projected onto the screen in 3D through CrystalEyes stereo shutter glasses. Graphics for the display are generated by an SGI Onyx2.

A twin camera system mounted above the projection table is used to provide stereo images of the user and, more specifically, the user's hand(s). The camera system consists of two Sony DFW-VL500 digital colour video cameras placed 150mm either side of the origin, and verged on the optimum distance of 1500mm. The coordinate system is specified as a right-hand coordinate system with the cameras on the x axis, the y axis vertical and the z axis pointing out from the cameras into the scene. Video frames are sent by firewire to a dual Pentium III 750MHz computer where the image processing is carried out in

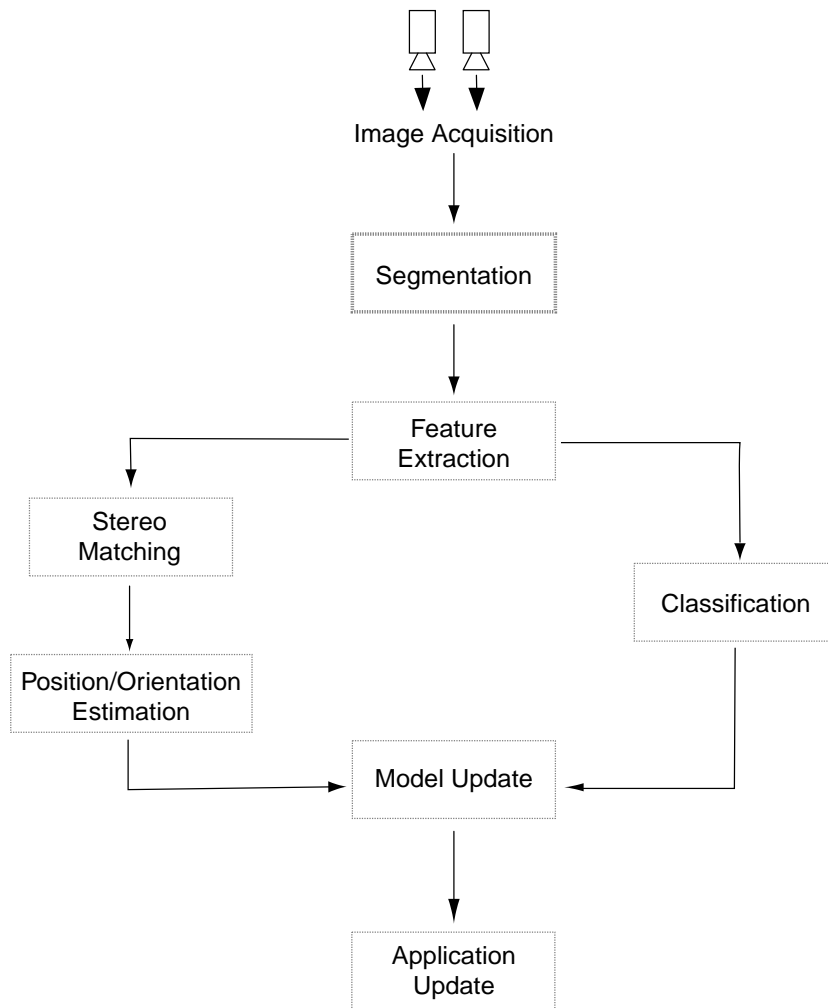


Fig. 1. Vision-based framework for gesture recognition software.

Gesture and positional information is passed between the recognition and graphical display systems via a socket-based data link. This connects the PC to the SGI and allows position, orientation and event information to be trans-

mitted to the application program.

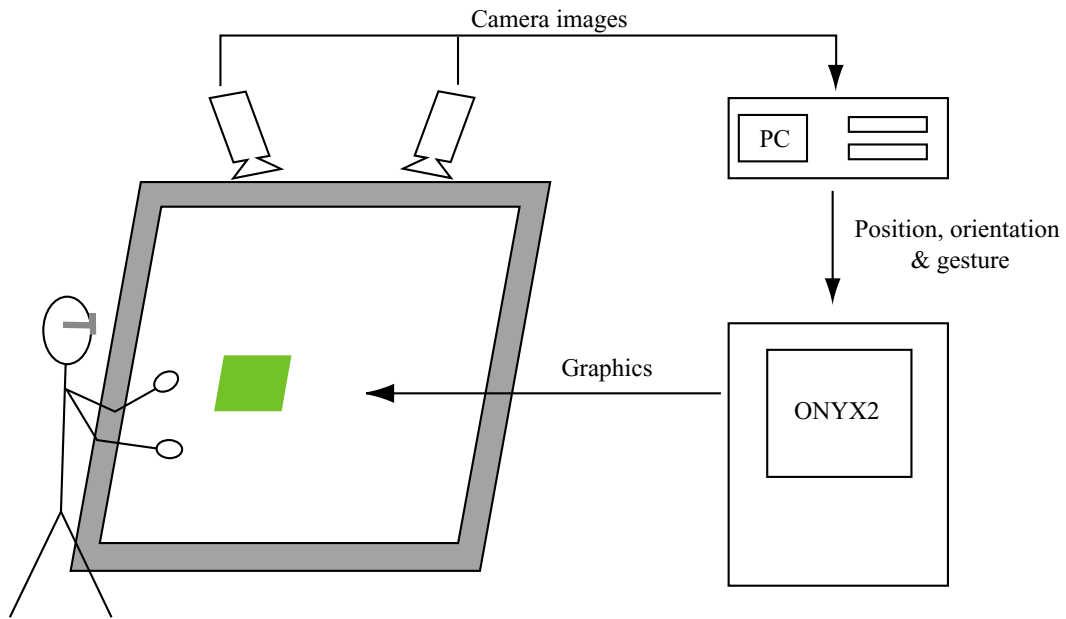


Fig. 2. System setup

2.4 Camera Calibration

In order to be able to accurately compute the 3D position of points from the video images, details of each camera's intrinsic and extrinsic parameters are required. Camera calibration is carried out to determine these parameters in terms of a fundamental matrix \mathbf{F} which provides a linear projective mapping between camera pixel coordinates and real-world coordinates:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = F \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

where x' and y' are the pixel coordinates of the 3D physical point (x, y, z) . The technique used for calibration is described in detail in Trucco and Verri (1998).

2.5 Model Acquisition

Having an underlying model of the hand in a particular pose assists in the feature extraction task. Knowledge of the location of features within the previous frame can be used to define a narrow search window for finding the feature in the current frame. This reduces the search area and hence the computation required to find the features. In a system where all processing for a frame must be completed within 33ms (in order to maintain a frame rate of 30Hz) minimising computation is essential. The model also enhances the robustness of the system. If some features are not tracking well (due to items obscuring part of the hand for example) the model allows their position to be estimated from the locations of the remaining features. This prevents the features from “wandering off” to track other parts of the image in error.

In the current system, model data is acquired manually. A pair of images is grabbed and the user asked to select appropriate feature points by clicking on them with the mouse. The 2D locations of the points in each image are then saved to a file. The camera images are also saved so that templates can be obtained for use in feature extraction. Previously we have investigated skin colour segmentation of the hand and image or geometric feature extraction as a means of identifying the location of parts of the hand and for classification of the pose. Details of these processes are included below. It is our intention to utilise our segmentation and feature detection results to automatically detect features so that models can be built on the fly and through classification,

switch between models as the gestures change.

When tracking is first started, the model is built. The 2D points are read from the file and used to compute the 3D location of each feature. Once all the features are specified, the centre of gravity of the model is computed, and the model translated to the origin of the world coordinate system and stored.

Template images are acquired at each feature location in the images and stored as a reference for correlation during the feature extraction stage. To assist in finding the hand within the image at startup and when tracking is lost, a low resolution template is taken of the area surrounding the model centre of gravity. This can then be used for quick searching within a shrunken version of the entire image when required.

Creation of a model for subsequent tracking can be accomplished in no more than a couple of minutes. New models only need to be generated if the environment or user changes dramatically.

2.6 Segmentation

Normalised skin colour detection can be used to find the hand within an image. Various algorithms have been developed for detecting skin coloured regions within RGB (Yang and Waibel, 1996) or YUV (Stark et al., 1995) images. We use a skin colour model developed in YUV colourspace. In each frame, those pixels within the volume of colourspace likely to be skin are extracted. Sensitivity to changes in lighting conditions is reduced by using a normalised model. Following detection, several filtering steps are applied to determine the hand region. The largest connected region of extracted pixels is chosen as the hand and any small holes within the region filled. This allows a mask to be

created and applied to the original intensity image for extraction of the hand. Using the extracted intensity image provides more information than straight segmentation. Typical results of the segmentation process are shown in Figure 3.

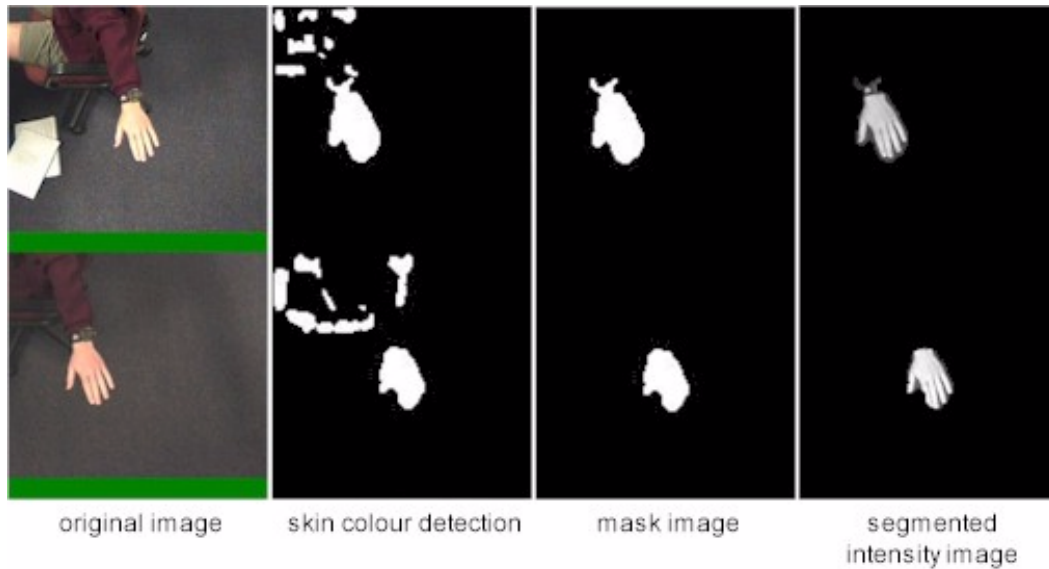


Fig. 3. Segmentation of the hand

2.7 Feature Extraction

We have investigated several different techniques for feature extraction including template matching, moment analyses and geometric properties analysis. A combination of these methods is used to determine the position of features such as the wrist, fingertips and base of fingers. Geometric properties such as areas of high curvature can be used to indicate regions of interest such as the probable location of fingertips or wrist-hand junction. By examining the outline of the hand region, points of local minima and maxima curvature can be found and matched to corresponding finger tips and valleys. Segmented hand images can be analysed to determine moment values, area and principal axes. These characteristics are then used to form a feature vector for classification.

For tracking, template matching is used to identify model features within the images. Template images are acquired when the model is built. In each frame, an area around the estimated location of the feature in the image is defined as a search window. The size of this window is increased when tracking is lost, and gradually reduced to a minimum size once tracking is regained. The template image is correlated against the current image within the search area. The point of highest correspondence marks the estimated location of the feature within the search area. A “confidence” value is returned by each feature being tracked indicating how well the template matched within the image. This confidence value is then used to weight the feature’s contribution to the calculation of the pose estimate.

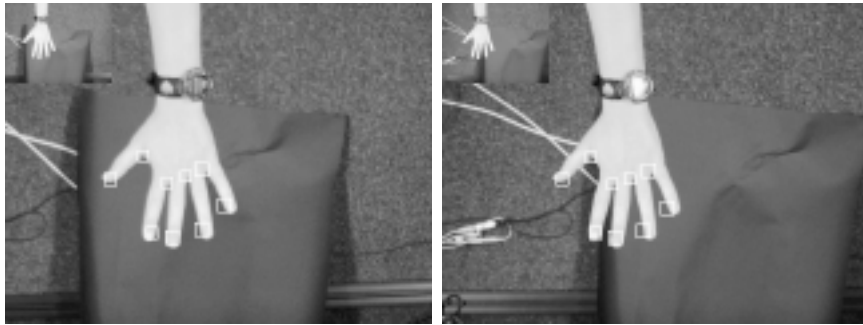


Fig. 4. Template tracking (left & right images)

Figure 4 shows template matching for a single frame. The overlaid boxes show the current estimated location of the model features within the left and right images. The size of the box is proportional to the confidence value with a large box representing a good match.

2.8 Model Update

Following the location of features within the camera images, an estimate of the pose of the hand model is calculated. Obtaining the optimal pose requires

determining the translation T and rotation R that would put the model at the observed position and orientation. Because of noise inherent in the observed locations of features, this becomes a minimisation problem. The goal is to minimise the error in T and R given a set of n model points \vec{m}_i and the observed points \vec{x}_i :

$$E = \sum_{i=0}^n w_i \|\vec{x}_i - R\vec{m}_i - T\|^2 \quad (1)$$

A weighting factor w_i is used to emphasise those features that matched best.

The translation, T , can be determined by setting the differential with respect to T equal to zero:

$$T = \frac{\sum_{i=0}^n w_i \vec{x}_i}{\sum_{i=0}^n w_i} - R \frac{\sum_{i=0}^n w_i \vec{m}_i}{\sum_{i=0}^n w_i} = \vec{\mathcal{X}} - R\vec{\mathcal{M}}$$

where $\vec{\mathcal{X}}$ and $\vec{\mathcal{M}}$ are weighted averages of the measurements and model points respectively.

The rotation matrix R is determined using the method in Horn (1987). This involves finding the eigenvectors of a matrix \mathcal{A} , with the eigenvector corresponding to the maximum eigenvalue minimising the rotation dependent terms in (1).

Each feature in the model is then updated by transforming the model point by the pose estimate, and projecting the 3D value back into the image plane to obtain the 2D location of the feature within the image.

2.9 *Gesture Classification*

Classification of hand gestures involves determining the particular pose of the hand and/or its motion, and deriving the meaning given a gesture set and the application context. In some situations, the pose of the hand along with its location in 2D or 3D space is sufficient information to determine the meaning and perform the appropriate action. In others, the temporal nature of gesture is required as well. In either case, a method of classifying various poses and movements is necessary. Several techniques have been used for classification in gesture recognition systems. These can be broken broadly into those directly using image features (e.g. (Clifton and Pang, 1997), Segen and Kumar (1998)), those using statistical or probabilistic methods (Rosales (1998), Stark et al. (1995)) and machine learning techniques (including Hidden Markov Models (Pavlovic et al., 1996; Becker, 1993), and neural networks (Väänänen and Böhm, 1992; Fels and Hinton, 1998)).

Although the classification step in our gesture recognition system is still under development, we have developed a classifier based on statistical methods that recognises a small number of gestures. By placing more emphasis on the context of an action, the number of distinct gestures can be reduced. This simplifies the interface presented to the user thus minimising the effort required to interact with the objects. The gestures chosen for object manipulation tasks are described below, and shown in Figure 5.

2.9.1 *Gesture Set*

Selection

Without doubt the most important task in interaction is the ability to select — an object, menu item, tool from a tool bar or a viewpoint. Selection is

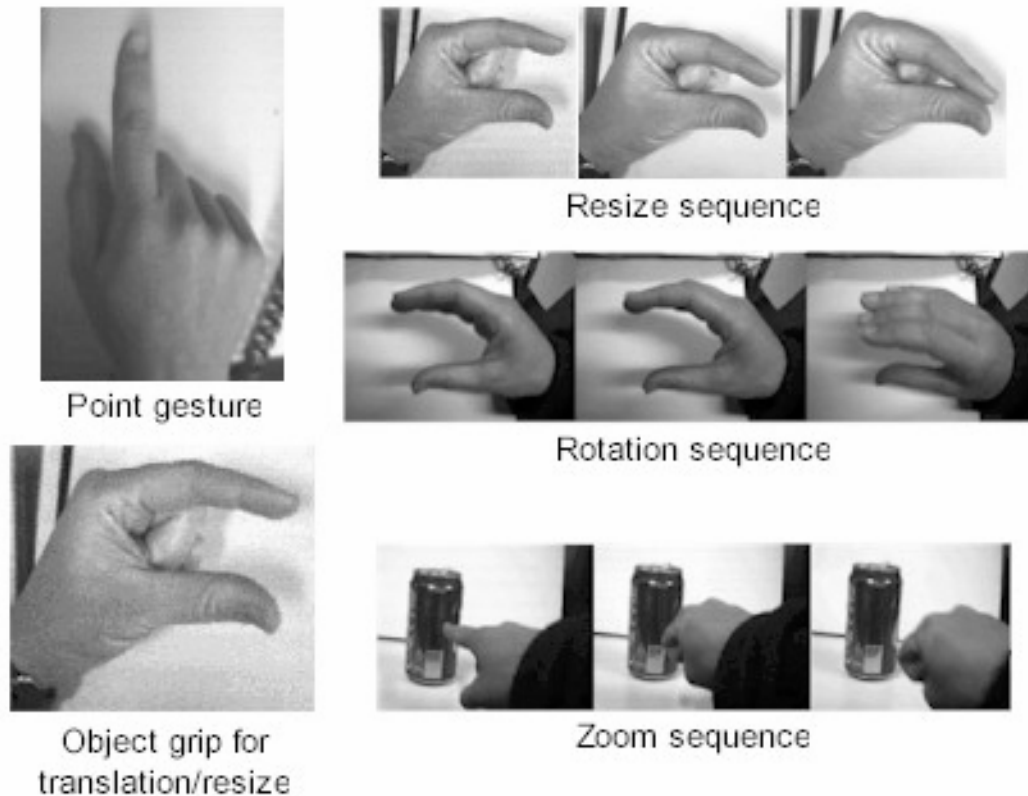


Fig. 5. Examples of the gestures to be used

also the task most demonstrated in gesture recognition systems. The gesture used for selection in these systems is typically that of a pointing finger, in the same way as we would naturally point to physical objects in the real world. Identifying the selection gesture is straightforward. The detection of a single finger outstretched, with the rest forming a fist clearly distinguishes this gesture from the others in the gesture set. The location of the “pointer” is defined to be the tip of the finger.

Object/scene translation

The standard “drag and drop” metaphor for moving objects can be extended to closely match the way we move objects in the real world. For example to move a wooden block we would pick it up, move our arm to the destination location and let the block go. In simulating this method of interaction we

chose to let the user grip the virtual object and perform the translation as if the object were real. The thumb and fore-fingertip can be recognised and located in 3D space. This gives the points of contact on the object in question. Tracking the thumb and finger until they *release* the object or close into a fist allows the destination to be determined.

Object/scene rotation

In a similar manner to object translation, object rotation is usually achieved by picking up the object and then rotating the wrist. The points of contact between the fingertips and the object determine the axes of rotation. With large rotations, the object is often rotated a small amount, released and then grasped again to rotate further. This prevents the wrist from twisting further than is comfortable.

Again mimicking the approach chosen with object translation, object rotation is carried out by grasping the object — either in the same grip as the translation, or by a larger grip as in the rotation sequence in Figure 5. The axes of rotation can then be determined and rotations achieved by rotating the wrist.

Object resizing

The question of how an object should be resized in a “natural” way is an interesting one as there are few directly equivalent tasks in the physical world. We cannot, for example, change the size of a wooden block. The closest we come to this is to stretch or squeeze objects of certain materials to change their shape.

However, in virtual worlds, it is a common task to change the size of an object. A method for resizing objects that seems intuitive would be to simulate stretching the object in question by either grasping the object with both hands and moving the hands away from each other to enlarge, or towards each other

to reduce. Alternatively, a one-handed approach could have the thumb and forefinger moving apart or towards each other as in the sequence in Figure 5. An option for non-uniform scaling of objects requiring two-handed interaction could use one hand to grasp the object at one end and the movement of the other hand would stretch or compress the object at the second hand’s point of contact.

Scene zoom

A common zooming technique for virtual environments mimics the action of reaching for an object and involves selecting a “zoom tool” (which effectively changes the mode of interaction) and then moving the input device towards or away from the screen to zoom in or out of the scene. This is easily replicated in a gesture interface by having a recognisable hand pose to set the mode to “zoom” and then tracking the motions of the hand until the mode (or hand pose) changes.

2.9.2 Classification Method

Statistical methods of classification are based on using a priori information to build a probabilistic model for classification (see Schalkoff (1992)). The classifier we have developed is based on a set of logistic regression models — one for each gesture in the gesture set. Logistic regression is a form of regression analysis where the response variable Y is qualitative rather than quantitative (see Chatterjee et al. (2000)). For each gesture, we model the probability that the response variable takes one of two values, 0 or 1 where 0 means the image being classified doesn’t belong to this class of gesture, and 1 means it does. The probability π is modelled as:

$$\pi = Pr(Y = 1 | X_1 = x_1, \dots, X_p = x_p)$$

$$= \frac{e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p}}{1 + e^{\beta_0 + \beta_1 x_1 + \dots + \beta_p x_p}} \quad (2)$$

where X_1, \dots, X_p are the predictor variables and β_0, \dots, β_p are model parameters. Applying the *logit transformation* allows (2) to be linearized such that:

$$\frac{\pi}{1 - \pi} = e^{\beta_0 + \beta_1 x_1 + \dots + \beta_p x_p} \quad (3)$$

where the ratio $\pi/(1 - \pi)$ is the *odds ratio* for the event. Taking the natural logarithm of both sides of (3) gives

$$\begin{aligned} g(x_1, \dots, x_p) &= \log\left(\frac{\pi}{1 - \pi}\right) \\ &= \beta_0 + \beta_1 x_1 + \dots + \beta_p x_p \end{aligned}$$

which is a linear combination of the predictor variables from which we can compute the probability that the image is a particular gesture.

Geometric moments computed during feature extraction are combined to form a feature vector of predictor variables. The feature vectors of a large training set were then used to model the parameters β_0, \dots, β_p above. When classifying, the values of the vector elements are passed to the model equation and the probability calculated to determine the likelihood that the image shows a particular gesture. This calculation is performed for each gesture equation, and the image is classified as that with the highest probability above a threshold. If the classifier returns low probabilities for all gestures, the image is determined to be unclassified.

2.9.3 Classification Results

We have used statistical classifiers to distinguish between a small set of gestures. A set of 5000 images was captured with 1000 images of each gesture. Ten percent of each gesture were randomly selected and kept as a testing set

while the remainder were used for training. The images were passed through the segmentation and feature extraction process to produce feature vectors. The feature vectors were then used to build five logistic regression models — one for each gesture. Images could then be grabbed, processed and passed through each of the models to obtain a set of probabilities indicating the likelihood of each gesture. The image was classified as being the gesture with the highest probability above a threshold (ie. winner takes all). If all probabilities fell below the threshold, the image was classified as “other”.

The images used in the training set tried to encompass the major variations in how each gesture could be presented. This was to enable the classifier to be robust to variations in rotation both in and out of the plane. Because of this, the classifier can distinguish gestures even when the hand is tilted to the side or up and down. This is important for our application to object manipulation as performing object transformations may result in the hand images being non-planar.

Results of classification of the testing set are given in Table 1. As can be seen, classification of flathand, fist and point gestures is very successful, however distinguishing between grasp and pinch gestures was more difficult. This is due to the high similarity between the gestures. We hope to overcome this problem by using different parameters in the feature vector.

2.10 Application Update

Having determined the position and orientation of the hand and classified the gesture, it remains only to notify the application program of the changes and allow appropriate actions to be taken. For the navigation interface, this involves converting the hand data into an equivalent change in the view di-






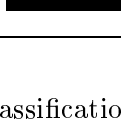
Gesture		N Rows	Classified as					% Misclassified
			Flathand	Fist	Grasp	Point	Pinch	
Flathand		100	100	0	0	0	0	0
Fist		100	0	100	0	0	0	0
Grasp		100	0	0	92	1	7	8
Point		100	0	2	0	98	0	2
Pinch		100	0	3	24	0	73	27

Table 1

Results of classification of test set images.

rection. Instead of using the angle of the hand to directly specify the angle of rotation, the viewpoint orientation can be thought of in terms of a continuous rotation, with the amount of angle indicating the rate of change in the orientation. This is similar to the method used to control flight simulators and results in an interface where the user can “fly” their hand over the terrain or dataset. Transforming the pitch, roll and yaw angles into a corresponding rotation of the viewpoint is achieved by adding the angles to the current viewpoint rotation angles. In future stages of development where multiple gestures are used, other information will be sent to the application to indicate specific events such as object selection or manipulation.

We use a simple TCP/IP socket-based connection to send data between the video processing and visualisation systems. The position, orientation and any event or gesture information is converted into a data package and sent by the server across the network to the client where it is unpacked. The position information must then be transformed from the camera coordinate system into the virtual world coordinate system before being used to update the position of objects within the scene.

3 Application — Object & Viewpoint Manipulation in a Virtual Environment

The first stage of our vision-based gesture interface is to provide a means for the user to navigate through the large datasets and scenes often encountered in virtual environments. In work by Segen (1993), navigation through a landscape was accomplished using the fingertip as an indicator of the direction of travel. We have chosen to utilise the flexibility of the wrist and the whole hand for navigation. By tilting the wrist up and down while holding the hand flat, the user can fly higher and lower with the model. Similarly, rolling the wrist and changing the direction of the hand initiates a turn. This is an intuitive set of movements which should be familiar to users. Figure 6 shows a snapshot of the system.

3.1 Experimental Setup

This application combines many of the components of our framework. A model is acquired and built from images of the user's hand. The features used are the tip of each finger and the valley between each finger as shown in Figure



Fig. 6. Viewpoint manipulation

4. This provides nine features to track. Position and orientation estimation requires a minimum of three features to be tracking, allowing several of the features to be obscured without losing tracking. The underlying model also restricts wandering of the templates when features are obscured or lost thus improving the robustness of the system.

During the tracking process, images are grabbed and the feature templates correlated within the search area defined by the previous frame feature location. Each feature returns a value indicating the confidence in the correlation result. These values are used to weight the position and orientation estimates, and to indicate when tracking is lost and searching should begin. The 3D location is calculated for each feature from the locations found in the template correlations. These 3D feature locations are then used to estimate the position and orientation of the hand model.

The current pose of the hand is computed by estimating the translation, T , and rotation, R , that best transforms the model to fit the measured feature locations.

Following pose computation, each feature in the model is updated by transforming the feature model point by the current pose and then projecting it back into the image plane of each camera. This updates the 2D feature locations and hence the search windows for correlation in the next frame.

The new pose is packaged and sent over the network to the application running on an SGI Onyx2 where it is used to update the position and orientation of the users viewpoint in the scene.

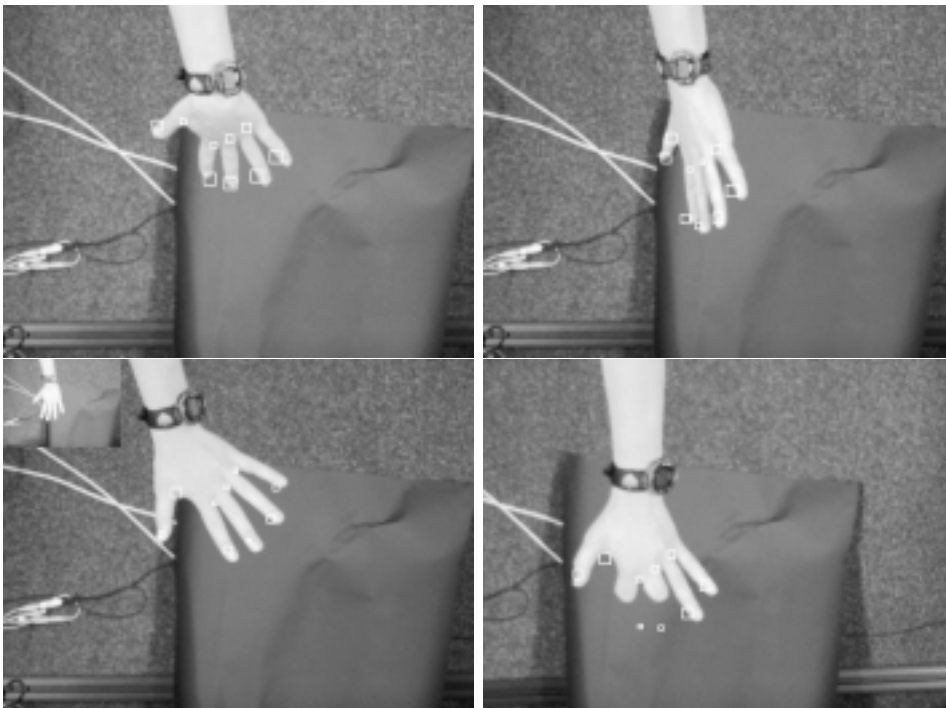


Fig. 7. Tracking the hand

3.2 Performance Results

The system tracks the user's hand at video frame rate (30Hz). This is fast enough that the user experiences little or no lag in response to movements. The work volume of the system is defined as the area in which tracking will generally succeed i.e. this is the area that is visible to the cameras. This volume is approximately 560mm in X , 420mm in Y and 1400mm in Z (in the camera world coordinate system).

Estimates of the error in position and orientation of the hand are difficult to obtain as it is difficult to measure the user's hand pose directly. In order to gather some measure of the accuracy of the system, we tracked a calibration pattern (a grid of squares) and measured the difference between the actual position and orientation of the plate and the estimates. Again, limitations in measuring ground truth restrict our ability to gather accurate performance data. From our measurements of rotations about the Y axis from 50° through to -15° in 5° steps, we have determined that the average error in orientation is approximately 1° .

An indication of the range of rotational motion is given by Figure 7. Tracking continued up to the angles shown in Table 2.

Positional accuracy was measured by computing the location of 120 points in a plane (the corners of the boxes on the calibration pattern) at 100mm intervals from 1300mm to 1700mm away from the cameras. The mean error overall was 0.08mm in X , 0.05mm in Y , and -0.19mm in Z with standard deviations of 0.44mm, 0.48mm and 1.47mm respectively. The error in position varied very little over this range of distance from the cameras. This is to be expected as the cameras were calibrated at 1500mm which is the expected distance of the

Axis	Rotation	Angle
	Direction	°
X	+ve	57
	-ve	82
Y	+ve	57
	-ve	81
Z	+ve	45
	-ve	48

Table 2

Angle at which tracking was lost
hand from the cameras.

Tracking can also cope with missing or obscured features, and is able to resume even if the hand is withdrawn from the work volume completely and then re-enters.

4 Gesture Interface Issues & Limitations

There are several issues concerning the use of gesture as an interface. One of these is the idea of feedback to the user. A gesture-based interface requires the user to interact with objects that have no physical representation. This is in contrast to traditional interfaces where the user interacts with the objects indirectly through a physical device. Recent developments in haptics (Salisbury, 1999) provide a solution to the feedback problem by providing force-feedback to the user when they *touch* a virtual object. In this way, the user can literally

feel the physical properties of the object. However, current systems are limited. Most provide only point contact to the objects, rather like feeling an object with a pen. Additionally, many systems are not co-located with the virtual objects, so the user's hands aren't in the same physical space as the objects they are feeling. Gesture, while not giving the user the benefit of touch, allows the user to interact with the objects in the exact space that they appear and to remain unconstrained by the physical restrictions of mechanical devices.

Another issue to consider in developing gesture-based interfaces is user fatigue. From observation, it is clear that manipulating objects in the volume in front of the user can quickly become tiring. This problem can be reduced by careful selection of the gesture set and understanding of the function of the hand. The effects of these issues cannot be properly evaluated until the system is fully implemented and useability testing conducted.

Vision-based gesture is also susceptible to lighting, particularly in the low-light environments often required with projection displays. The use of infra-red cameras can help to alleviate this problem, and may be utilised in the future. Template tracking of deformable models is another challenging problem. At present, our system assumes the model will remain effectively rigid. Difficulties in tracking arise when the hand is deformed as features no longer look the same as their template or may be no longer visible. We are investigating the use of multiple models switching on classification results to overcome this difficulty.

At present, our system tracks only one of the user's hands. In order to improve the quality of interaction when manipulating virtual objects it would be advantageous to track both hands. This would allow the user to grasp an object with one hand and perform some other action on it with the other as we often do in reality. Unfortunately the difficulties in using vision to track two very

similar objects that may occlude each other put this beyond the scope of the current project.

5 Conclusion & Further Work

We have investigated the use of gesture as an interface for virtual environments and in particular the use of computer vision techniques in developing such an interface. Gesture recognition offers a natural, unobtrusive method of interaction, especially if the gesture set is selected with some knowledge of the nature of human hand movements. Vision techniques eliminate the need for gloves or restraining cabling back to the computer and so provide an unobtrusive interface. Examination of previous work in this area provided a basis for the development of a framework for a vision-based gesture interface, as well as identification of promising techniques. The recognition process can be broken into steps; namely segmentation, feature extraction, stereo matching & 3D location determination, model update, classification and application update. Each of these steps can be implemented in many ways and experiments have been conducted to explore various techniques. A system for navigating in 3D worlds within virtual environments has been developed. The user can “fly” through the scene by tilting their flat hand up or down, rotating it left or right and by twisting the wrist. The hand is segmented from video images via normalised skin colour detection and is stable under varying lighting conditions. A combination of geometric properties and template correlation is used to find and match feature points in the stereo images. The 3D coordinates of these feature points are then calculated and the position and orientation of the hand estimated and passed to the application program to update the viewpoint of the user.

A set of gestures has been developed for object manipulations such as selection, translation, rotation, resizing and scene zoom. A classification system has been developed allowing multiple gestures to be recognised. Future work will extend the interface to include manipulation of objects within the environment. This will provide the user with an interface for navigating and manipulating objects in virtual environments in a new, natural and effective manner. In order to fully determine the benefits of the interface, rigorous useability studies should also be conducted in the future.

6 Acknowledgements

This work is supported by The Australian National University Department of Computer Science and Research School for Information Sciences and Engineering, and by the Co-operative Research Centre for Advanced Computational Systems. In particular, thanks goes to Yoshio Matsumoto and Tim Edwards for their technical knowledge and expertise.

References

- Azarbayejani, A., Wren, C., Pentland, A., May 1996. Real-time 3-d tracking of the human body. In: Proceedings of IMAGE'COM 96. Bordeaux, France.
- Becker, D. A., 1993. Sensei: A real-time recognition, feedback and training system for t'ai chi gestures. Master's thesis, School of Architecture and Planning, Massachusetts Institute of Technology.
- Bolt, R. A., 1980. Put that there. ACM SIGGRAPH Computer Graphics 14 (3).
- Bowman, D. A., Hodges, L. F., 1999. Formalizing the design, evaluation , and

- application of interaction techniques for immersive virtual environments. *The Journal of Visual Languages and Computing* 10 (1), 37–53.
- Chatterjee, S., Hadi, A. S., Price, B., 2000. *Regression Analysis by Example*, 3rd Edition. John Wiley & Sons, Inc.
- Clifton, M., Pang, A., September/October 1997. Cutting planes and beyond. *Computers & Graphics* 21 (5), 563–575.
- Fels, S., Hinton, G., January 1998. Glove-talkii: A neural network interface which maps gestures to parallel formant speech synthesizer controls. *IEEE Transactions on Neural Networks* 9 (1), 205–212.
- Figueiredo, M., Boehm, K., 1993. Advanced interaction techniques in virtual environments. *Computer & Graphics* 17 (6), 655–661.
- Hand, C., 1997. A survey of 3d interaction techniques. *Computer Graphics forum* 16 (5), 269–281.
- Heap, T., Hogg, D., 1996. 3d deformable hand models. In: *Gesture Workshop*.
- Horn, B. K. P., 1987. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America* 4 (4), 629–642.
- Pavlovic, V. I., Sharma, R., et al., October 1996. Gestural interface to a visual computing environment for molecular biologists. In: *2nd International Conference on Automatic Face and Gesture Recognition*. IEEE Computer Society Press, Killington, Vermont.
- Rehg, J. M., Kanade, T., December 1993. *Digiteyes: Vision-based human hand tracking*. Tech. rep., School of Computer Science, Carnegie Mellon University.
- Rosales, R., 1998. *Recognition of human action using moment-based features*. Tech. rep., Boston University.
- Salisbury, K., 1999. Making graphics physically tangible. *Communications of the ACM* 42 (8), 74–81.
- Schalkoff, R., 1992. *Pattern Recognition: Statistical, Structural and Neural*

- Approaches. John Wiley & Sons, Inc.
- Segen, J., 1993. Controlling computers with gloveless gestures. *Virtual Reality Systems* .
- Segen, J., Kumar, S., 1998. Gesturevr: Vision-based 3d hand interface for spatial interaction. In: *Proceedings of the ACM Multimedia Conference*.
- Segen, J., Kumar, S., July 2000. Look ma, no mouse! *Communications of the ACM* 43 (7), 103–109.
- Sharma, R., Zeller, M., Pavlovic, V. I., Huang, T. S., Lo, Z., Chu, S., Zhao, Y., Phillips, J. C., Schulten, K., March/April 2000. Speech/gesture interface to a visual-computing environment. *IEEE Computer Graphics and Applications* , 29–36.
- Shimada, N., Shirai, Y., Kuno, Y., Miura, J., April 1998. Hand gesture estimation and model refinement using monocular camera - ambiguity limitation by inequality constraints. In: *3rd International Conference on Automatic Face and Gesture Recognition*. Nara, Japan.
- Stark, M., Kohler, M., Zyklop, P. G., November 1995. Video based gesture recognition for human computer interaction.
- Trucco, E., Verri, A., 1998. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall.
- Väänänen, Böhm, 1992. *Virtual Reality Systems*. Academic Press Ltd, London, Ch. Gesture-driven interaction as a human factor in virtual environments - an approach with neural networks, pp. 93–106, given2.ps, paper.
- Wren, C. R., 2000. *Understanding expressive action*. Phd thesis, MIT Media Laboratory.
- Yang, J., Waibel, A., 1996. A real-time face tracker. In: *WACV 96*. Sarasota, Florida.