ABSTRACT
Although the 2D desktop metaphor has been the dominating paradigm of user interfaces for over two decades, 3D models of interaction are becoming more feasible due to advances in computer output hardware and software technology. However, conventional input devices such as a mouse or track-pad generally restrict direct manipulation interaction to a 2D paradigm. More sophisticated 3D input devices such as data-gloves have been available for some time, but these tend to be expensive or restrictive in their use. In this paper we describe a simple and inexpensive single camera-based video input system which allows 3D interaction with existing computer applications using bare hands.

Keywords
Bare hands input, gesture input, computer vision.

ACM Classification Keywords
H5.2 [Information Interfaces and Presentation]: Graphical User Interfaces (GUI), Input Devices and Strategies, Interaction Styles.

INTRODUCTION
Since the appearance of first direct manipulation interfaces in the 1980s, human computer interaction has been generally limited to working on a 2D “desktop” surface. Although this flat surface metaphor has been useful in providing the users with some direct manipulation capabilities, this metaphor doesn’t completely match the real-world desktops that are 3D environments in which objects can be picked up, moved around and put on other objects.

Over the years some attempts have been made at moving the human computer interaction into the 3D world. Various experimental interfaces have been designed which utilize the third dimension. These systems have, however, not yet seen widespread adoption due to the limitations of the input and output technologies.

In recent years rapid progress has been made in improving the size and quality of computer displays. Large vertical interactive displays, as well as, tabletop displays have become more common due to the falling cost of data projectors and other technologies such as plasma screens. The speed of computer and display hardware have also increased, thus allowing fast rendering of 3D interaction environments.

Development of computer input technology, on the other hand, has unfortunately been much less rapid. Almost all conventional input devices such as mice, keyboards, trackballs, touch-pads, joysticks, electronic pens, and digitizers are designed for two dimensional input along the X and Y coordinates. Although the Z coordinate input can sometimes be achieved using projection on a 2D plane, or simulated using for instance the middle mouse scroll button, this type of input is not really natural and can be confusing.

There are specific 3D input devices such as data-gloves and 3D-mice in the market. These, however, tend to be either costly, or are cumbersome to use.

A better range of solutions to the 3D input problem is the use of computer vision techniques to recognize hand movements in 3D space and use these for real 3D input to computer. Such experimental systems have existed for some time, but in the past their use has not been viable due to the limited speed of the available computer technology. This has now changed, and with the increasing computing power of current systems it is possible to perform complex image processing algorithms in real-time, while running other applications at the same time on a single system.

In this paper we describe a simple and inexpensive solution to providing real 3D input mechanism using bare hands. A single camera is used over a desktop to capture and recognize hand gestures which are then used to control computer applications directly. The system is used to control two example applications, the Microsoft® Solitaire game and Windows Picture Viewer™. Although these applications do not have real 3D interfaces, they are used to demonstrate the capabilities of a 3D input mechanism even in a 2D interaction environment.
RELATED WORK
A number of researchers have explored the use of hand gestures as a means of computer input, using a variety of technologies. Methods that have been used include one or more of data gloves [8, 6, 1, 4], bats or wands [2, 4, 6], and video camera(s) [7, 10, 5].

Implementation issues in gesture systems include intention/immersion, recognition and segmentation of gestures. Intention is the problem of deciding whether the user is genuinely gesturing or not; avoiding false positives from recognizing casual motions as gestures. The term immersion [1] is used to describe the problem of the user’s hand being always under analysis. This is a feature of gesture recognition systems that is not experienced in conventional mouse and keyboard interaction, where the user can simply let go, completely disengaging themselves from interaction. Recognition is identifying the user’s intended gestures, while segmentation is determining where one gesture ends and another begins in a command sequence.

In 1989 Sturman et al [8] used a data glove, capable of measuring hand position, hand orientation and flex angles for the thumb and each finger, to allow manipulation of objects in a virtual world. They considered three ways in which gesture input might be used. The first is to directly manipulate objects by ‘reaching into’ an environment and holding and moving. They observed that this is, or might be, the most natural and intuitive way of interacting with 3D objects. They, however, found out that the absence of tactile feedback caused some difficulties. The second use of gestures by Sturman et al was to operate abstract input devices: buttons, valuators and locators, positioned in a three dimensional volume. In their system hand static postures and motion onset were used as buttons to trigger actions. Experiments with finger flexing found that people did not readily issue static gestures depending on accurate finger angles. They found it practicable only to distinguish extreme positions: fully extended or tightly clenched, despite being able to measure finger flex to within two degrees. Sturman’s third use of gesture was to issue a sequence of abstract tokens, as occurs in human sign languages.

The VOES system by Lee et al [4] adopts the third of Sturman’s interaction types: issuing a sequence of commands. Using a data glove and Polhemus bat device the user controls the movement of an avatar. This style of interaction makes the most demand on gesture segmentation, and the authors separate gestures using a finite state automaton, tracking the variation in hand speed and acceleration. Gestures are characterised by a period of sharp motion corresponding to a circular track in acceleration/speed space. An example is the ‘stop’ gesture, with the hand held up, palm forward. This is issued by a sharp movement into the gesture position, with rapid deceleration to the finished gesture. Surprisingly, they report reliable acceleration measurements with a 10 Hz sampling rate.

The Charade system [1] used a data glove to control a slide presentation. They described a gesture set with 16 commands (some are extensions or variations of others). They observed that human gestures associate tension with intent. For instance, the use of a clenched fist, or pointing with the index finger; postures in which the muscles of the hand are in tension. The 16 commands were chosen to start with a tense posture which was then relaxed. The tension/relaxation sequence also provided a pattern that can be used to segment gesture sequences. They also used the idea of an active zone to limit immersion. When a user wasn’t pointing somewhere near the presentation screen, their gestures were ignored. In the specialist Charade application, this allowed users the freedom to gesture to their audience without the danger of having gestures interpreted as presentation commands.

Despite these interesting systems, researchers generally report a preference for untethered operation. It is inconvenient, and sometimes uncomfortable to wear devices with trailing wires. A better option for users is to at least wear wireless positioning aids, e.g. tracking devices, LEDs or coloured marks. However, the ideal solution is free movement of the bare hands, tracked by video or other means. In our intended context, of co-located collaboration, the need to either have large numbers of positioning devices and the complexity that adds, or the social inconvenience of sharing devices support the argument for bare hand tracking.

Segen et al [7] describe one such system, which uses two fast cameras (60Hz frame rate) capturing stereo images of the hand to build a bare-hand gesture-based system for virtual object manipulation and 3D navigation (flying over virtual terrain). In their system the hand moves over a table with uniform colour and illumination. Only three gestures are used: point, reach (all fingers extended) and click (index finger curled as in pulling a trigger) – all high tension gestures. Expectation of tactile feedback is avoided by interacting indirectly with objects. The user points at something and then gestures to indicate action. There is no pretence of contact. Even with the fast cameras, Segen notes that motion blur can compromise image analysis. The chosen gesture set copes because blurred images can be ignored with little impact on commands.

Hardenberg et al [10] also describe single camera-based systems which robustly detect hand gestures against a varied but generally light background. They use image differencing with a slowly updating reference image to allow static gestures, whilst still reacting to changing illumination. In one system the hand can be used as a mouse replacement, using a finger to point and using a one second pause as a mouse click.

Visual Touchpad [5] is a two-handed gestural interface in which stereo cameras track a hand moving on and over a
pad (or clipboard) type surface. The surface is black with white edges, allowing calibration by image analysis. Interestingly, the black background means that image analysis does not have to contend with shadows of the hands. As well as allowing height estimates, the stereo images are used to decide whether or not the hand is in contact with the pad surface. The system is sensitive to a 1cm lift above the pad.

In summary, intention has been dealt with in a number of ways. Allowing a user to easily withdraw from the system (active zone), and using high tension gestures to reduce the likelihood of accidental triggering seem the most appropriate in our application. Segmentation seems to be most easily managed by choosing gestures such that adjacent commands in sequences will be orthogonal. Finally we adopt the idea of a controlled background (desk surface) to ease the gesture detection (image analysis) problem. The desk surface also provides a base for precise manipulation and lessens user fatigue. Our system is novel in that it detects gestures in 3D with a single camera.

HAND3D PROTOTYPE

Our main objective in developing a 3D bare-hands gesture recognition system is to provide easy and natural control of shared documents in co-located collaborative work environments. In such environments it is important to allow multiple users to interact with the system seamlessly without requiring them to put on special gloves or markers, or pass around input devices every time they want to use the system. Requirements, such as passing an input device around, often interrupt the natural flow of the group work, and therefore are undesirable.

To provide for such a seamless group interaction mechanism with a computer system we have developed a prototype system called HAND3D. Although HAND3D is intended to be used in a collaborative work setting, perhaps along with a shared tabletop display, for development purposes we have mainly focused on a single-user work environment; however the system is readily usable within a shared group work setting.

Figure 1 shows a diagrammatic view of HAND3D. The system consists of a single standard web-cam hanging roughly 50cm above a white gesture area, approximately the size of an A2 sheet of paper. The setup does not require any calibration, other than the need for the camera view to contain the gesture surface area. The camera view is homographically mapped to the computer display by identifying the four corners of the (camera view of) the gesture surface with the four corners of the display[5].

It should be noted that although the use of a single camera, instead of two cameras, reduces the precision of the height detection along the Z-axis, this reduction in precision outweighs the cost of setting up and calibrating multiple camera systems. Furthermore, the intended use of this system is in shared collaborative environments, where most activities such as selecting, pointing, and manipulation of documents do not require very high precision.

SKIN DETECTION

HAND3D uses skin detection and background subtraction techniques to isolate the image of hand on each video frame, which is then used for gesture recognition. The skin detection algorithm used is a modified version of that discussed in [9]. Skin is identified by looking for pixels in which:

\[
(R > 40) \land (G > 20) \land (B > 10) \land \\
(max(R, G, B) - \min(R, G, B) > 10) \land \\
(abs(R - G) > 10 \land (R > G) \land (R > B)
\]

where \(R, G,\) and \(B\) are the 8 bit red, green, and blue colour intensities respectively.

All the non-skin coloured pixels are masked with the white colour, as shown in Figure 2. The white coloured pixels can then be made translucent when the input video frames are superimposed on application windows being displayed on the computer screen.

Figure 1. HAND3D prototype

Figure 2. Isolated hand and its various measurements

Skin detection on its own is not enough to isolate the hand, as sometimes the bare arm area is also detected by the skin detection algorithm (e.g. when the user has short sleeve clothing).
We have, therefore, devised mechanisms for isolating the hand based on reasonably standard proportions of human hands. The basic rules that we apply are:

- when the hand is open and flat and the thumb is closed (see Figure 2), the length of the hand, from wrist to the middle fingertip, is roughly about twice the width of the hand in the middle section (the palm).
- when the hand is in fist position, the length of the hand from the knuckles to the wrist is almost the same as the width of the hand (see Figure 7).
- when the hand is open and held sideways (see Figure 5), the length of the hand is about four times the width of the hand.
- when the hand is open and held vertically (see Figure 6), the wrist is identified when the hand width becomes less than 0.7 of the maximum width of the hand (around the palm).

These rules are used in conjunction with other rules to decide if:

- the thumb is on the right side of the hand (left hand with palm facing down, or right hand with palm facing up).
- the thumb is on the left side of the hand (right hand with palm facing down, or left hand with palm facing up).
- the thumb is closed (near the palm) or open (away from the palm).

Some fluctuations occur in showing the visible hand, particularly when the hand is flipping or when the thumb is moving in and out (see Figure 8), but these changes are minimal and do not affect gesture recognition algorithms we have developed.

3D GESTURE RECOGNITION

Current version of HAND3D is capable of recognizing several gestures. These gestures are demonstrative of a range of gestures which can be performed in our 3D video-based recognition environment, and we are in the process of developing algorithms for recognizing other gestures.

As with our hand isolation techniques, our gesture recognition algorithms utilize the stable proportions of human hands, particularly in terms of movements along the Z axis, where the size of the hand changes as it gets closer or further away from the camera. We utilize the coordinates of three points for most gestures:

- the tip of the middle finger: (firstX, firstY)
• the left-most pixel of the hand when the thumb is on the right, or vice-versa (handX, handY)
• the middle of the hand, which is halfway between the tip of the middle finger and the wrist: (avgX, avgY)
These points are used when the hand is open (shown in Figure 2), and similar points are calculated when the hand is closed in a fist or flipped.

2D Movements
Movements along the X and Y axes when there are no noticeable changes along the Z axis (hand moving up and down) are considered as 2D movements (Figure 3). 2D movements, regardless of their vertical distance from the camera, are translated to cursor movements along the X and Y axes on the computer display. In our system we use the position of the middle of the hand (avgX, avgY) as the cursor position, as this is more sensible for grabbing and moving objects, etc.

3D Movements
Movements along the Z axis (see Figure 4) are perhaps the most difficult ones to measure accurately using a single video camera. Although one could assume that measuring the size of the hand as it gets closer or further away from the camera is a good way of measuring distance, in reality people tend to change their hand orientation as they move it up and down, by for instance not holding their hand flat, moving their thumbs and fingers, or flipping the hand.

To measure the movements along the Z axis, we measure the size of the hand, as well as using the proportions of the hand through two measurements (shown in Figure 2):
• distXO which is the difference between handX and avgX
• distYO which is the difference between avgY and firstY
Changes in the values of distXO and distYO, while their ratio remains reasonably constant, are then used to calculate the movements along the Z axis, i.e. if the hand is moving towards the camera or away from it. If both distXO and distYO are increasing then the hand is moving up, and if they are both decreasing then the hand is moving down. Of course changes to distXO and distYO could also mean that the hand is flipping, or changing to fist, but in these cases the ratio distYO/distXO changes radically, a fact which we also use to recognize other gestures. Currently our system assumes that if the ratio of distYO/distXO remains between 1.3 and 2.5 the hand is not flipping.

Horizontal flip
Recognizing the horizontal flip, as shown in Figure 5, is very simple and is based on the ratio of distYO/distXO. If the hand is not a fist, and the ratio of distYO/distXO becomes greater than 2.5 then the hand is taken to be flipped horizontally.

Fist
To recognize the fist gesture, as shown in Figure 7, the narrowest part of the hand at the top is compared with the widest part of the hand. If the hand is open (unfolded) then the narrowest part at the top is the fingertips, while the widest part is around the palm; in this case the difference between the two is large. For a fist, the difference between the narrowest and widest part is small.

Thumb Movements
Thumb movements (Figure 8) are recognized if the position of the thumb changes over at least 10 frames of the video input, while the hand remains stationary.

Left and right hands
Our system works well regardless of whether the left or right hand is used for gesturing, or whether the palm of the hand is facing towards the camera or away from it. This is achieved by the system automatically identifying the position of the thumb in relation to the rest of the hand. Thumb is simply identified as a small section of skin either on the right or left of the main section of skin (i.e. a narrow section of skin versus a wide section of skin).

RECOGNITION RATES
We have done some testing of HAND3D in a standard office environment, with normal lighting conditions. 50 samples were taken for each gesture by a single user. The results of these tests are shown in Table 1.
We consider these recognition rates acceptable for our intended use of the system – to control shared applications in a co-located collaborative environment in which large displays are used to interact with documents in a less-than-precise manner.
Furthermore, although hand orientation recognition rates are low, these are only used when the hand is initially

<table>
<thead>
<tr>
<th>Gesture</th>
<th>Recognition rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D movements</td>
<td>100</td>
</tr>
<tr>
<td>3D movements</td>
<td>98</td>
</tr>
<tr>
<td>Horizontal flip</td>
<td>98</td>
</tr>
<tr>
<td>Vertical flip</td>
<td>100</td>
</tr>
<tr>
<td>No Flip</td>
<td>100</td>
</tr>
<tr>
<td>Hand fisted</td>
<td>100</td>
</tr>
<tr>
<td>Hand open</td>
<td>100</td>
</tr>
<tr>
<td>Hand orientation (left)</td>
<td>96</td>
</tr>
<tr>
<td>Hand orientation (right)</td>
<td>90</td>
</tr>
<tr>
<td>Thumb moving</td>
<td>94</td>
</tr>
<tr>
<td>Thumb stationary</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 1. Gesture recognition rates
placed under the camera and are quickly stabilized within a first few frames of the input video.

**PERFORMANCE**

On a standard desktop computer (2.6 GHz P4, 512MB) the system processes images 30±1 fps. We accept gestures that are recognised for successive 10 frames, giving a recognition time of 0.33 sec. Actual gesture recognition is very fast after initial image processing, so we anticipate no difficulty in scaling to a larger gesture set.

**PROTOTYPE APPLICATIONS**

We have developed an interface for HAND3D which allows it to be used for controlling other applications by generating mouse events and sending them to the interested applications. We have also used this interface to control two example applications. Although both of these applications have conventional 2D interfaces, we have been able to experiment with the usability of 3D gestures for controlling these applications.

![Figure 9. Use of gestures in Solitaire a) no gesture b) move over a card c) grab and move d) move to the right place e) release f) no gesture g) move over a card h) flip the card](image)
Figure 10. Use of gestures in Image Viewer a) no gesture b) zoom in c) grab d) move to the right part e) horizontal flip f) move left to previous image g) move right to the next image h) move thumb to move over toolbox items i) vertical flip to select toolbox item j) toolbox item selected
**Microsoft® Solitaire**

Figure 9 shows various screenshots of HAND3D being used to control the Microsoft® Solitaire game. HAND3D video outputs have been overlaid on the Solitaire window. The video outputs currently include debugging data at the bottom left-hand side of the screen, as well as the input video frame rate at the top right-hand corner.

Figure 9a shows a game in progress, where the hand is open and no gesture is being done. Figure 9b shows the hand moving over the second column of cards, which are then picked using the fist gesture (9c), moved over the empty space in the fifth column (9d) and placed there by opening the hand (9e). The hand is then moved over the card remaining in the second column (9f, 9g) and then the flip horizontal gesture is used to turn the card to face up position (9h).

To control Solitaire we only use the 2D movements (mouse location), open hand to fist for picking cards (mouse down and hold), fist to open hand for placing cards (mouse up and release), and horizontal flip for dealing cards or turning a card (mouse click). For performing these gestures the hand does not need to be placed on the gesture surface, and can be moved up and down naturally (e.g. in picking cards).

**Windows Picture Viewer™**

Figure 10 shows the screenshots of HAND3D being used to control the Windows Picture Viewer™. We use a wider range of gestures for controlling this application. In particular motion in the Z direction is used to control zoom.

Figure 10a shows the hand in an open position, and as the hand moves up the image is zoomed in (10b). The hand can then be closed into a fist to grab the image (10c) and move it around (10d). The horizontal flip followed by move to the left is used to go to the previous image (10e, 10f), while horizontal flip followed by move right is used to go to the next image (10f, 10g). The thumb move is used to go through the toolbox menu items at the bottom of the screen (10h). Once a toolbox menu item has been reached, the vertical flip is used to select it (10i, 10j).

**CONCLUSIONS**

In this paper we have described a 3D bare-hand gesture recognition system using a single camera. We have used the system to demonstrate the capabilities of 3D input in interactions with a 2D environment. However, the value of 3D gestures will be more apparent in true 3D interfaces, and we are planning to work on example 3D applications as well as formally user testing our existing applications.

We are also currently working on extending the range of 3D gestures that our system recognizes, including those related to hand rotation and orientation, as well as folding and unfolding the individual fingers. Other intended future work includes recognition of two handed gestures tracked by using two individual cameras over two separate gesture surfaces.

**REFERENCES**


