Visual Touchpad: A Two-handed Gestural Input Device

Shahzad Malik, Joe Laszlo
Department of Computer Science
University of Toronto
smalik | jflaszlo @ dgp.toronto.edu
http://www.dgp.toronto.edu

ABSTRACT

This paper presents the Visual Touchpad, a low-cost vision-based input device that allows for fluid two-handed interactions with desktop PCs, laptops, public kiosks, or large wall displays. Two downward-pointing cameras are attached above a planar surface, and a stereo hand tracking system provides the 3D positions of a user's fingertips on and above the plane. Thus the planar surface can be used as a multi-point touch-sensitive device, but with the added ability to also detect hand gestures hovering above the surface. Additionally, the hand tracker not only provides positional information for the fingertips but also finger orientations. A variety of one and two-handed multi-finger gestural interaction techniques are then presented that exploit the affordances of the hand tracker. Further, by segmenting the hand regions from the video images and then augmenting them transparently into a graphical interface, our system provides a compelling direct manipulation experience without the need for more expensive tabletop displays or touch-screens, and with significantly less self-occlusion.

Categories and Subject Descriptors

H.5.2 [**User Interfaces**]: Graphical user interfaces, Interaction styles. I.4.m [**Image Processing and Computer Vision**]: Miscellaneous.

General Terms

Algorithms, Design, Human Factors.

Keywords

direct manipulation, gestures, perceptual user interface, hand tracking, fluid interaction, two hand, visual touchpad, virtual mouse, virtual keyboard, augmented reality, computer vision.

1. INTRODUCTION

Recently, a number of input devices have made it possible to directly manipulate user interface components using natural hand gestures, such as tabletop displays [2][14][15][23][24], large wall displays [6][10][16], and Tablet PCs equipped with touch sensors [25]. Users typically find that interacting with such devices is much more enjoyable and efficient than using a mouse and keyboard, largely due to the increased degrees of control as well as the comfort and intuitiveness of the input. Additionally, the user interfaces for such devices typically allow a user to focus more on the task at hand, rather than diverting attention between different input devices and visual elements.

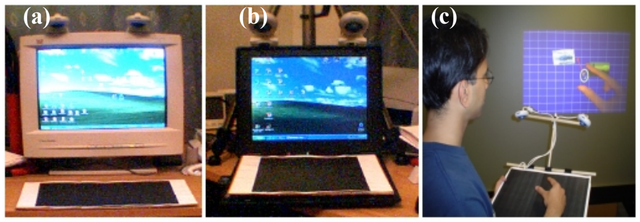

Figure 1. Example configurations for the Visual Touchpad: (a) Desktop setup; (b) Laptop setup; (c) Hand-held setup.

However, a major drawback with such touch sensitive input devices is the frequent occlusion of the display. For example, many long-time stylus users that have become accustomed to using an external tablet surface find that using a Tablet PC (which requires a user to touch the stylus directly to the display) is actually more difficult due to their hand frequently occluding their work. This becomes even more apparent with systems that recognize entire hand gestures directly over the display surface, such as tabletops and interactive walls. Another shortcoming with standard touch sensitive devices is that they usually only recognize hand gestures on or close to the surface. Rekimoto's Smartskin technology [15] can detect hand proximity to some extent, but it is still difficult to determine specific feature points for gestures too far above the surface. Finally, another problem with touch sensitive surfaces is the lack of robust finger orientation information, which is useful for certain types of operations. In other words, while accurate position information can be determined for the tip of a finger touching the surface, it is very difficult to determine in which direction the finger is pointing without requiring the whole finger to be placed flat on the surface.

In this paper, we explore the idea of using computer vision techniques to track a user's bare, unmarked hands along a planar region that simulates a touch-sensitive surface. By using stereo vision we can not only determine contact information, but also the distance of a fingertip from this Visual Touchpad surface for additional types of input. We can also use vision techniques to extract finger orientation information for other more advanced interactions. Finally, we can extract the hand regions from the video images in real-time, and then transparently augment them over top of the graphical interface as a visual proxy for the user's actual hands. This allows for the directness of tabletops and touch-screens, but with the added ability to still see items that are beneath the hand area. One and two-handed gestures are then recognized over the touchpad in order to manipulate 2D graphical elements in an intuitive and fluid manner. Such a device allows for direct two-handed gestural interactions on desktop and laptop PCs, public kiosks, or large wall displays from afar.

2. RELATED WORK

Some of the earliest work demonstrating computer vision-based hand tracking for interaction without the use of gloves or markers was Krueger's VIDEOPLACE [10], where silhouettes of hands could be used to generate 2D line drawings on large projection screens.

Mysliwiec's FingerMouse [13] demonstrated a single-camera vision system that could track a pointing finger above the keyboard, allowing mouse control without explicitly having to move the hand over to another device. This increases the efficiency of tasks which require constant switching between mouse manipulations and text entry. However, the system as presented only simulates a mouse with a single button.

The Wearable Virtual Tablet [22] allows any planar rectangular object such as a magazine to be used as a touch-sensitive tablet via an infrared camera attached to a head-mount display. The system can recognize single finger pointing gestures to simulate mouse cursor movement, while contact with the tablet surface is determined by analyzing the depth-dependent grayscale pixels around the fingertip area.

The tabletop display community has also been using computer vision finger and hand tracking recently. Wellner's DigitalDesk [23] demonstrated a number of interesting single and multi-finger interaction techniques in order to integrate real and virtual data, using microphones to capture "tapping" sounds for selection operations. Similarly, the EnhancedDesk project [2] [14] uses infrared cameras to detect the 2D positions of all the fingertips of each hand for such tasks as two-handed drawing and GUI navigation, but their single camera setup cannot determine whether a finger is touching the table surface. Corso et al. [3] presented the 4D Touchpad, a bottom-projected tabletop system that uses a stereo camera setup to extract the 3D position of fingers above the table surface. Rather than tracking hands globally in each video image, they instead passively monitor regions of interest in the image for sequences of "visual interaction cues". For example, a region representing a pushbutton would watch for cues such as motion, skin color blobs, and finger shape. While their system has significant potential in terms of rich interactions, they only demonstrate a simple button press detector by implementing a virtual piano that allows a user to simulate pressing and releasing piano keys. Finally, MacCormick and Isard [12] presented a vision-based hand tracker using a particle-filtering approach that provides 2D position and orientation information for the thumb and index finger. They demonstrate the speed and robustness of their system by implementing a 2D drawing application.

The tabletop community has also investigated non-vision based solutions by using special touch-sensitive hardware. For example, Wu and Balakrishnan [24] present a room planning system using a touch sensitive tabletop that can detect multiple points of input from multiple users. Similarly, Rekimoto's SmartSkin [15] allows the detection of multiple contact points for tabletop displays, allowing full hand gestures to be recognized.

Yee [25] describes a modification for Tablet PCs that allows two-handed interaction. Using a touch-sensitive overlay, the Tablet PC can detect single finger contact information in addition to the Tablet PC's original stylus device. A number of interesting asymmetric two-handed tasks are described in order to leverage this additional mode of input.

A number of researchers investigating interaction techniques for large wall displays have also considered using hands directly as input. For example, the system by Hardenberg [6] detects and tracks unmarked hands using computer vision techniques in order to select and move objects in 2D. In essence, the system allows a pointing finger to control 2D mouse cursor movement, with a single second delay to simulate button clicks. Similarly, the BareHands system [16] describes a method to interact with a large touch screen by mapping various hand postures to commands such as copy and paste, thereby saving the user from having to select these operations from a menu.

In most of the above mentioned systems that use back-projected displays [16] [25], the main drawback is the frequent occlusion of the screen area by the hands. As a result, a user frequently tries to peer around the occluding hand or moves it away from screen, thereby disrupting the focus of attention. Clever placement of widgets and menus can remedy the situation somewhat [24], but at the expense of lost screen real estate or more complicated menu layouts.

Our work largely builds upon the Visual Panel system described by Zhang et al. in [26]. In their system they track a quadrangle shaped piece of paper using single-view computer vision techniques, then extract the position of a fingertip over the panel in order to position the mouse cursor in a Windows desktop. Since the panel is not equipped with any buttons or touch sensors, mouse clicks are simulated by holding the fingertip position steady for one second. Text entry is achieved by way of a virtual on-screen keyboard. Due to the one second delay, text entry and interface navigation can be quite slow. Additionally, the single fingertip detector only allows for two degrees of freedom, thereby limiting the input to single cursor mouse control. However, by extracting the X and Y orientation of the actual panel from some base pose, they are able to simulate a joystick which is useful for another two degrees of freedom.

Using the Visual Panel as a starting point, we present a variety of new interaction techniques that are possible when we combine it with stereo cameras and a more sophisticated gesture recognition system that can detect more than a single hand or finger as well as fingertip contact with the panel surface. Also, by augmenting the live images of a user's actual hands directly into the graphical interface, our Visual Touchpad begins to provide a more compelling "hands-on" experience similar to tabletops or touch-screens while the use of transparency during augmentation avoids the occlusion problems associated with other hand-based interaction schemes such as tabletops or Tablet PCs. Figure 1 shows some example configurations of our system.

Various researchers have recognized the value in augmenting displays with overlaid live video proxies of the body for compelling visual feedback. Tang's Videowhiteboard [20] and Ishii & Kobayashi's ClearBoard [7] display overlaid video of a collaborator working on a shared planar workspace. Buxton [1] presents a good discussion of these and related earlier work, which lies primarily in the area of shared workspaces and collaboration. Roussel's VideoPointer [17] proposes the overlay of a user's hand as an expressive remote pointing device. In their Video FaceTop [19], Stotts, Smith & Jen overlay the desktop with a live video reflection of the user, which can be used to manipulate onscreen widgets.

With the exception of the latter, these works use overlaid live video primarily for remote awareness in applications such as teleconferencing, and do not make use of the video proxy as an active user input. In contrast, we make use of the self-image both to provide visual feedback without occlusion, and as a direct user input mechanism in a similar spirit to that of [19].

3. SYSTEM OVERVIEW

3.1 Hardware

Similar to the Visual Panel [26], the Visual Touchpad is a simple quadrangle panel such as a piece of paper with a rigid backing, over which hand gestures can be recognized for interaction purposes. In our system, we use a piece of paper with a large black rectangle in the centre, surrounded by a thin white border. This black region defines the active "touchpad", while the white border facilitates the vision algorithms described later. The size of the paper can be any size as long as we can place both hands comfortably over the touchpad. Additionally, the touchpad should ideally have the same aspect ratio as the display that will be used for visualization. Section 3.4 discusses this in more detail.

Two off-the-shelf web cameras are then placed in a convenient location such that the black rectangular region of the touchpad is fully visible to both cameras, and the cameras are placed with a sufficiently wide baseline for accurate depth estimation. The cameras can capture 320x240 images at 30 frames per second on a standard Pentium 4 PC. For desktops, laptops, and kiosk configurations, it is sufficient to fix the location of the touchpad in front of the display, and then place the cameras on top of the display facing downward (Figure 1a and Figure 1b). For interacting with large wall displays from afar, we propose attaching the cameras directly to the panel (Figure 1c) and using the panel as a handheld input device, instead of hanging the cameras from the ceiling as in the original Visual Panel work. The advantage with our fixed panel approach is that the system is easier to set up, and the pattern will always be visible to the cameras. The disadvantage is that we lose the ability to extract the panel orientation, but we make up for these lost degrees of freedom with a more sophisticated hand gesture recognition system.

3.2 Homography Computation

To simulate a touch-sensitive surface, we assume that the corners of the touchpad map to the corners of the display (Figure 2). In order to determine this mapping, we use a homography [4], which defines a plane-projective mapping between two planes. In order to compute a homography we require the positions of at least four points on one plane and the corresponding four points on the other plane.

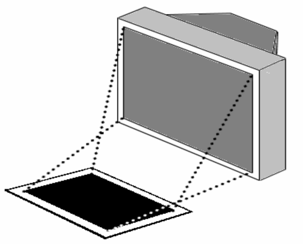
Figure 2. Touchpad to screen mapping.

Therefore, for each of our cameras, we detect the four corners of the touchpad in a captured video frame and then compute $H_{i \in \{1,2\}}$, which represents the homography that maps camera *i*'s view of the touchpad into display coordinates.

To find the corners of the touchpad in a frame of video, we use simple binary image processing operations. First we threshold a grayscale version of the video frame into a binary image in order to segment out the high contrast black rectangle that is surrounded by the thin white border (Figures 3a and 3b). We currently use a fixed value of 128 for our 8-bit grayscale image threshold, which works well in most situations. A flood-fill technique is then used to extract the largest black connected component in the video frame, and for this black blob we extract the four strongest corner features (Figure 3c). A homography is then computed using these four touchpad corners and the corresponding corners of the display. The assumption here is that the Visual Touchpad fills most of the image such that the largest black blob region will correspond to the black rectangular region of the touchpad.

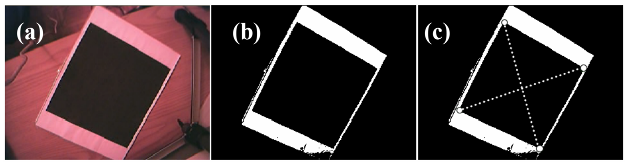

Figure 3. Touchpad detection: (a) Original frame; (b) Thresholded binary image; (c) Corners detected.

3.3 Hand Tracking

In this section we describe the details of the hand tracker, which applies low-level image processing operations to each frame of video in order to detect the locations of the fingertips. While a model-based approach that uses temporal information could provide more robustness to situations such as complex backgrounds or overlapping hands, the image processing approach is straightforward to implement and can run in real-time with low cost PCs and cameras.

3.3.1 Image Rectification

Using H_i defining the mapping from the touchpad in camera *i* to screen space, our hand tracker first warps each frame of live video so that the panel (and any hand over top of it) is in screen space. Let p_j represent a pixel in screen space, and q_j represent the corresponding pixel from touchpad space. Therefore we have

$$\vec{q}_j = H_i^{-1} \vec{p}_j \tag{1}$$

Figure 4a and 4b show the result of creating a warped (screen space) image of the touchpad with a hand over top of it

3.3.2 Background Subtraction

Since we assume a black rectangular region for our panel, it is easy to segment out the hand from the warped image by using a simple background subtraction operation, where our background is simply a black rectangle covering the whole screen (Figure 4c). By using a black region as our known background, the system is quite robust to shadows cast onto the touchpad by foreground objects such as hands. Additionally, the system can reliably detect foreground objects in a wide variety of lighting conditions as long as they are different from the black background.

3.3.3 Hand Blob Detection

A flood-fill technique is then applied to the foreground objects, and the two largest connected blobs above some threshold size are assumed to be the hand blobs. Assuming that hands will not cross over during interaction, we simply label the left-most blob as the left hand, and the right-most blob as the right hand. In the case of only a single blob, we consider it to be either the left or right hand depending on a software setting that defines a user's dominant hand preference.

3.3.4 Fingertip Detection

The contours of each blob are then detected in a clockwise order and potential fingertips are found by finding strong peaks along the blob perimeters. We first use an approach similar to [18], where the vectors from a contour point *k* to *k*+*n* and *k*-*n* are computed (for some fixed *n*). If the angle between these vectors is below some threshold (we currently use 30 degrees) then we mark that contour point as a potential fingertip. To avoid detecting valleys (such as between fingers) we verify that the determinant of the 2x2 matrix consisting of the two vectors is negative. Non-maximal suppression is then used to avoid detecting strong fingertips too close to one another. Finally, orientation is determined by computing a line from the midpoint between contour points *k*+*n* and *k*-*n* to the fingertip point *k*. Figure 4d shows the result of fingertip position and orientation detection.

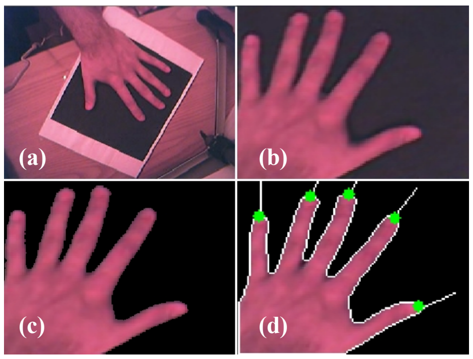

Figure 4. Hand detection in the warped image: (a) Original image; (b) Warped image; (c) After background subtraction; (d) Finger tip positions and orientations detected.

3.3.5 Fingertip Labeling

If a single fingertip is detected in the contour, it is always labeled as the index finger. If two fingers are detected, the system assumes they are the thumb and index finger, using the distance between each fingertip along the contour to differentiate between the two. For example, for the right hand, the distance from the index finger to the thumb is larger in the clockwise contour direction than the distance from the thumb to index finger. For three, four, and five finger arrangements we use a similar contour-distance heuristic for the labeling, with label priority in the following order: index finger, thumb, middle finger, ring finger, little finger.

3.3.6 Detecting Contact with the Visual Touchpad

For each camera, the hand detector gives us the *(x,y)* position of fingertips in screen space, as well as the orientation angle *Θ* of the finger. For fingertips directly on the surface of the touchpad, the positions will be the same regardless of whether we use the pose information from the warped image from camera 1 or the warped image from camera 2. However, for fingertips *above* the touchpad surface the positions of corresponding points will be different since the homography only provides a planar mapping (Figure 5). This disparity of corresponding points can thus be used to determine the distance of feature points above the touchpad surface [21]. To determine a binary touch state, we define a disparity threshold below which we consider a point to be in contact with the touchpad. For a given camera configuration, a threshold can be easily determined by holding the finger at a height above the touchpad which should be considered "off" the surface. The disparity of the fingertip position can then be used as the disparity threshold. In our experiments we have found that holding the finger approximately 1cm above the surface works well. The final output from our hand tracker is a set of *(x,y,z,Θ)* values for each detected fingertip, where *z* is a boolean value representing whether the finger is touching the surface. Note that we set one of our cameras to be the reference camera, and thus the *(x,y)* values for each fingertip are extracted from the hand contour associated with that camera. Additionally, the tracker can also provide temporal information, resulting in five parameters for each fingertip. The advantage of using disparity instead of 3D triangulation is that we do not need to perform camera calibration of any sort, which makes the system extremely simple to set up.

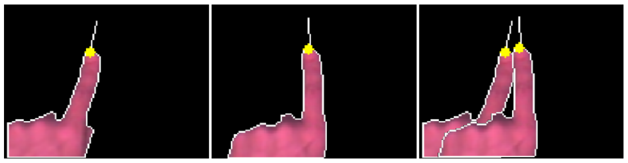

Figure 5. Using disparity for sensing height of raised fingers: (left) Rectified camera 1 view; (middle) Rectified camera 2 view; (right) Images overlaid together show corresponding points for raised fingers are not in same position.

3.3.7 Postures and Gestures

Given the output of the hand tracker, it is extremely simple to detect the four static postures depicted in Figure 6. The pointing posture is simply the index finger held straight out in some direction. The pinching posture involves setting the thumb and index finger as if something is being held between them, with the thumb and index finger pointing in relatively the same direction. The L-posture is a variation of the pinching posture, where the thumb and index finger are pointing in approximately orthogonal directions. For both the pinch posture and L-posture we can overload the recognition system with variations such as both fingers touching the touchpad surface, both fingers not touching the surface, or one finger on the surface and one finger off the surface. Finally the five-finger posture is simply holding out all fingers so that the hand detector can clearly identify all fingertips.

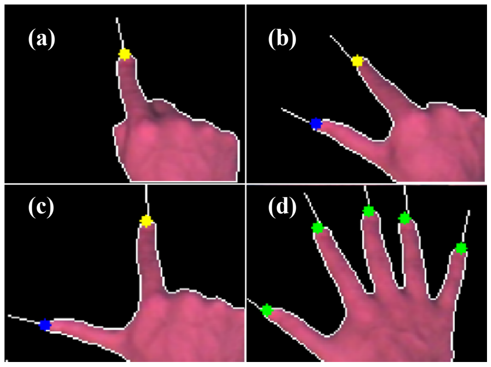

Figure 6. Posture set: (a) Pointing; (b) Pinching; (c) L-posture; (d) Five-finger posture.

Along with the static postures, our system can also detect gestures using temporal information. To demonstrate this capability, we currently detect a holding gesture (for all postures), a double-tap gesture (for the pointing posture), and an X shape gesture (also for the pointing posture). While these gestures are fairly simple there is nothing preventing our system from recognizing more complicated gestures, since all of the required information is available.

3.4 Hand Augmentation

The ability to use your hand for direct manipulations is one of the main advantages of devices such as tabletop displays or touch-screens. Roussel's VideoPointer system [17] proposed using a live video stream of a user's hand as a better telepointer for real-time groupware. Building on this idea, we propose augmenting the user's hand directly into the graphical interface, using the live video of the segmented hand region from the reference camera as a visual proxy for direct manipulations. The advantage of this approach is that a user feels more connected to the interface in a manner similar to tabletops or touch-screens, but while using an external display such as a monitor. The other advantage is that by rendering the hand as an image, we can apply other special effects such as transparency to help overcome the occlusion problem, as well as visual annotations onto the hand such as mode or state information. Figure 7 shows an example of a hand being augmented onto a graphical interface.

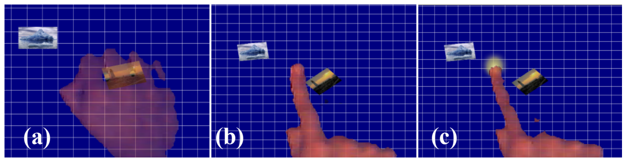

Figure 7. Hand augmentation: (a) No fingers; (b) Finger above surface; (c) Finger contacting touchpad.

Note that the size of the hand on the screen is dependent upon the size of the touchpad, due to the touchpad-screen homography; the larger the touchpad, the smaller the hand appears on the screen, and vice-versa. Thus the size of the panel should be proportional to the size of the display. As mentioned earlier, it is also best to have similar aspect ratios for the display and the touchpad so that the hand is rendered realistically.

When no fingers are detected by the hand tracker, any hand blobs are rendered with 50% opacity (Figure 7a). As soon as any fingers are detected, each fingertip is drawn at 85% opacity with gradual falloff to 50% opacity using a fixed falloff radius (Figure 7b). This allows the hand to come into focus when the user is performing some action. Additionally, when a fingertip is determined to be in contact with the touchpad, a yellow highlight is rendered beneath it for visual touch feedback (Figure 7c).

4. INTERACTION TECHNIQUES

To demonstrate the capabilities of the Visual Touchpad, a simple picture manipulation application has been implemented. A number of images are scattered around a canvas, and using hand gestures the user is able to move/rotate/scale the images, query object properties, pan/rotate/zoom the view, etc. Using some of the postures and gestures described earlier we show that the Visual Touchpad can be used to perform a variety of common GUI operations in a fluid manner.

4.1 One-handed Techniques

4.1.1 Object Selection/Translating/Rotating/Query

To select an image on the canvas, a single hand in a pointing posture can be positioned so that the fingertip is within the bounds of the object, with the fingertip touching the surface of the Visual Touchpad. When the finger makes contact with the touchpad a yellow glow appears around the fingertip. Additionally, the borders of the selected image become green to signify that it has been selected. To deselect the object, the user simply raises the fingertip up from the touchpad surface until the yellow glow disappears.

Once an object has been selected, it can be simultaneously translated and rotated. Translation is controlled by simply moving the finger in the appropriate direction. The image then remains attached to the fingertip. Similarly, the orientation of the finger controls the rotation of the object, with the centre of rotation being the fingertip position. Figure 8 shows an image being translated and rotated using the pointing gesture.

To query an object for information (such as file name, image dimensions, etc) we use an approach similar to tooltips found in graphical interfaces such as Windows. By simply holding a pointing posture for one second inside the boundaries of an image, but without touching the touchpad surface, a small query box is activated. Moving the finger out of the image dismisses the query box.

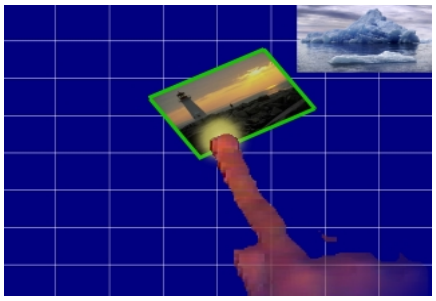

Figure 8. Image translation and rotation.

4.1.2 Group Selection/Copy/Paste/Delete

To select a group of images for operations such as copying or deleting we can make use of the double-tap gesture. By double-tapping on an image a yellow highlight appears around it signifying that it has been added to the current group. To remove a selected image from the group we simply double-tap it again. A single tap in any empty canvas location causes the entire group of objects to be deselected.

The selected group is always the set of objects in the clipboard so there is no need to explicitly perform a *copy* operation. To paste the selected group of images we use the L-posture with both fingers above the touchpad surface. The index finger position defines the centre of the selected group, and translation or rotation of the L-posture can be used to place the group in the desired location. To finalize the positioning the user simply places both the index finger and thumb onto the touchpad surface. After the paste operation, the new set of images becomes the active group selection. Note that the second hand can be used to navigate the canvas viewpoint simultaneously (as described in the next section). To *cancel* the paste operation the user can touch the thumb and index finger together without touching the touchpad surface. To *delete* a selected group of images a user draws an X in an empty part of the canvas.

4.1.3 Canvas Panning/Rotating/Zooming

To control the canvas viewpoint we use an approach similar to the SmartSkin map viewer [15]. Using a pinching posture, where the thumb and index finger are in contact with the surface of the touchpad, the user can simultaneously control the position, orientation, and zoom level of the window into the canvas. The idea is that as soon as two fingers make contact with the touchpad, they become "attached" to the corresponding positions within the canvas. Moving the hand around the canvas while maintaining the pinch posture causes the window into the canvas to move in a similar direction. To rotate the entire canvas, the hand can be rotated while the pinch posture is maintained. The centre of rotation is thus defined as the midpoint between the tips of the thumb and index finger. Finally, bringing the fingers closer together while still touching the surface causes the view to be zoomed out, while moving the fingers further apart causes the view to be zoomed in. The centre of zoom is defined as the midpoint between the thumb and index finger. In all cases, when translation, rotation or zooming becomes difficult due to the hand ending up in an awkward pose, the operation can be continued by simply raising the fingers off the touchpad surface, adjusting the hand to a comfortable position again, and then continuing the viewpoint control. Figure 9 shows an example of a pinch posture controlling the zoom level.

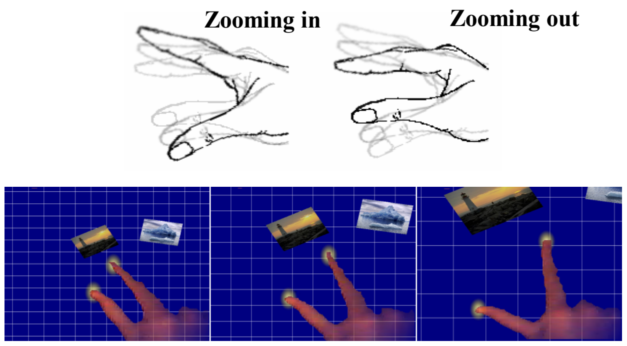

Figure 9. Canvas zoom control.

4.1.4 Navigation Widget

While the canvas viewpoint control described above works well for small adjustments of the canvas, it is inefficient when large-scale viewpoint changes are required. Since we are able to recognize postures and gestures above the touchpad surface, we propose a navigation widget that can be used for continuous scrolling of the viewpoint. To activate the widget the user holds a pinch posture steady for one whole second above the surface of the touchpad. Once activated, the system captures the midpoint between the thumb and index finger as the "centre" position. A navigation arrow then appears between the thumb and index finger, with a line connecting the current midpoint between the thumb and index finger to the "centre" position (Figure 10).

The widget then acts much like a joystick, where translation in any direction away from the "centre" causes the viewpoint to translate in that direction, with scrolling speed dependent upon the distance of the widget from the "centre". Canvas zooming can also be performed, by treating the navigation widget as a dial, where the "zero" rotation is the finger orientation at the "centre" pose. Therefore, rotation of the fingers in a clockwise direction causes the view to be zoomed in, while a counter-clockwise rotation causes the view to be zoomed out. The amount of rotation from the "zero" defines the speed of the zoom. To deactivate the widget, the user can simply pinch the fingers together completely.

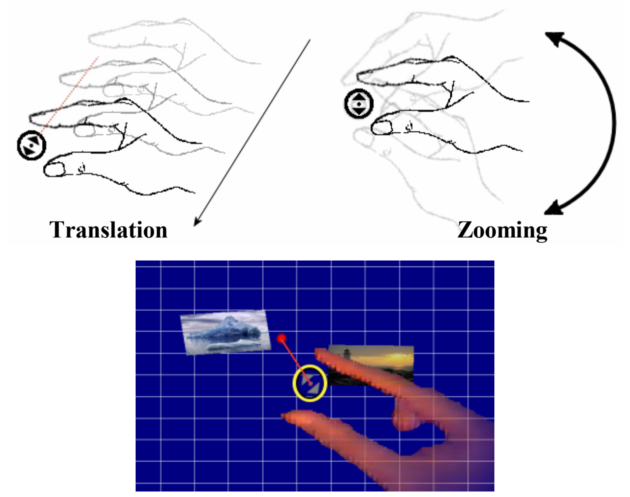

Figure 10. Navigation widget.

4.2 Two-handed Techniques

4.2.1 Pie Menu

Asymmetric-dependent tasks, as proposed by Guiard [5], are those in which the dominant (D) hand moves within a frame of reference that has been set by the non-dominant (ND) hand. Therefore, the ND hand will engage in coarse and less frequent actions, while the D hand will be used for faster, more frequent actions that require more precision. Kabbash [8] showed that such asymmetric-dependent interaction techniques, where the action of the D hand depends on that of the ND hand, give rise to the best performance since they most closely resemble the bimanual tasks that we perform in everyday life.

We follow such an asymmetric-dependent approach for our pie menu system that is used to select various options. To activate the pie menu the user performs a double-tap gesture using the ND hand. The pie menu (with a small hollow centre) is then displayed, centered at the ND hand's index finger. If the user maintains contact with the touchpad surface, the pie menu will remain centered at the index finger. If the index finger is raised from the surface, the pie menu will remain at the previous position, thereby allowing the user to select menu options with a single-tap. Another double-tap in the hollow centre is used to deactivate the pie menu. To illustrate the functionality of the pie menu, we implemented a simple drawing tool that allows the user to "finger-paint" onto the canvas. The pie menu consists of the following options: *drawing mode*, *draw color*, *draw size*, *draw shape*.

The *drawing mode* option acts as a toggle switch. When selected, the D hand's fingertip becomes a paintbrush, with an appropriate cursor drawn at its tip. The user can then paint strokes with the D finger when it is in contact with the touchpad surface.

By selecting the *draw color* option, a color palette is presented to the user. Moving the ND fingertip within the palette (while making contact with the touchpad surface) sets the color of the D hand's fingertip. To deactivate the color palette the user simply moves the ND fingertip out of the palette area.

The *draw size* menu option allows the size of the paintbrush tip to be modified. A slider appears when the option is selected, which can be modified by "dragging" the slider handle using the ND finger much like many 2D GUIs. The slider is deactivated by moving the ND finger outside of the slider's rectangular border.

Finally, the *draw shape* menu option allows the user to change the shape of the paintbrush tip. Four simple shapes are currently implemented as shown in Figure 11. Unlike traditional painting tools, ours allows for simultaneous control of not only the brush tip's position but also the tip orientation, thereby allowing for interesting calligraphic effects.

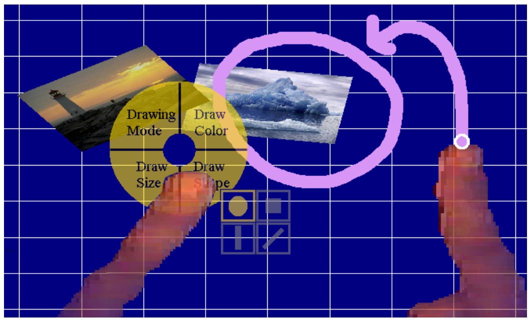

Figure 11. Pie menu for finger-painting.

4.2.2 Image Stretchies

Kurtenbach et al [11] introduced an interaction technique called "two handed stretchies" that allow primitive shapes to be simultaneously translated, rotated and scaled using two rotation-sensitive pucks on a tablet surface. The Visual Touchpad is also capable of such techniques using two-handed postures instead of pucks.

One hand with a pointing posture selects an object as usual. The position of the fingertip is then "locked" onto the selected image. The second hand then selects another position within the same image, and that position becomes "locked". Translating both fingers at the same rate and in the same direction allows for the image to be moved. However, translating the fingers in different directions or at different speeds will cause rotation and scale changes. The idea is that the two "locked" finger positions will always represent the same pixel in the image. While we currently do not use the finger orientations for this stretch operation, we plan to integrate it into the system in the future to allow for a simultaneous image warp as well.

4.2.3 Two-handed Virtual Keyboard

Many applications such as presentation tools, drawing tools, or web browsers require frequent switching between text entry (keyboard) and navigation (mouse). Virtual keyboards [9] are one approach to making text entry and navigation more fluid. By rendering a graphical layout of a keyboard on the screen, a user does not have to switch between input devices and can instead focus more on the desired task. Additionally, virtual keyboards can be reconfigured to different layouts based on a user's personal preferences. The downfall with most virtual keyboards is that they rely on single mouse clicks to simulate key presses, resulting in slow text entry.

Motivated by the familiarity and reconfigurability of virtual keyboards, we have implemented an onscreen *QWERTY* keyboard for the Visual Touchpad that can be used to make textual annotations on our image canvas. To activate the virtual keyboard, a user makes a five-finger gesture with both hands over the touchpad (Figure 12). This gesture simulates putting the hands onto "home-row" on a real keyboard. The virtual keyboard is then rendered transparently on the screen, with the hands rendered over top. By default, the text entry cursor is placed at the canvas location corresponding to the middle of the screen, above the virtual keyboard. Letters can then be entered by simply touching the appropriate virtual keys with the fingertips. The virtual keyboard is deactivated by pressing the virtual "Escape" key. Note that the mapping between the touchpad and the virtual keyboard is not dependent on the canvas window settings. Instead, the size and position of the virtual keyboard is fixed to some predetermined absolute region of the touchpad and a corresponding region of the screen so that the spatial layout of the keys remains constant.

By rendering the hands and keyboard together on the display, users do not have to divert their visual attention away from the onscreen task. Additionally, by using the same input surface for both text-entry and GUI navigation, the experience is much more fluid compared to traditional keyboard-mouse configurations. It is worth mentioning, however, that the current implementation does not allow for extremely fast text entry, largely due to the limited speed of our camera capture and image processing operations.

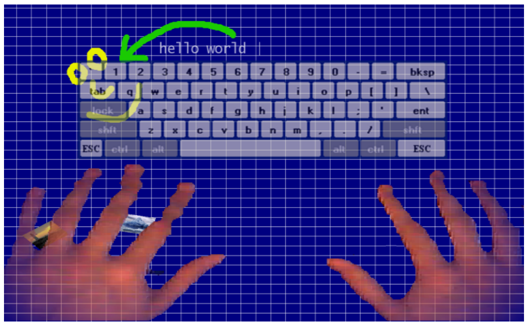

Figure 12. Virtual keyboard.

5. DISCUSSION

While detailed user experiments have not yet been performed, informal user feedback from students in our research lab has been very positive. Each user was given a brief introduction to the posture and gesture set, and then they were free to explore the system on their own for 10 to 15 minutes.

All users found the posture and gesture based manipulations to be extremely intuitive, with descriptions such as "cool", "neat", and "fun" to describe the overall system. One of the first things many people were impressed with was the ability to see their own hands on the screen, and as a result they found the direct manipulation techniques to be very compelling.

The asymmetric two-handed pie menu required a quick introduction in most cases, but afterwards all users found the pie menu to be easy to use. Although our pie menu only has four options on it, we tried a quick experiment to see if hand transparency makes any difference when portions of the menu end up beneath the hand. Some users were given a version with a fully opaque hand, while others were given a transparent hand. It was observed that a few of the opaque hand users would frequently move their hand off to the side of the pie menu if an option's title was occluded by the hand, while we did not see this with the transparent hand users. A more extensive study is required to accurately determine how effective our transparent hands are against occlusion, but these preliminary observations are encouraging.

While many users liked the idea of fluidly switching between navigation and text-entry modes, most felt that the virtual keyboard has some drawbacks. Most notably, it was felt that the lack of tactile feedback during keypresses made text entry awkward and prone to errors, since it was difficult to determine key boundaries. One user suggested using some more cues to signify which key was about to be pressed, such as highlighting the most likely key based on the trajectory of the fingertip, or generating audible key clicking sounds. Another complaint with the virtual keyboard was that it occupied a significant amount of screen real estate. An interesting suggestion was to gradually increase or decrease the transparency of the virtual keyboard based on a user's typing speed or idle time, under the assumption that a fast typist has memorized the spatial arrangement of the keys and does not need to see the keyboard as much.

In terms of system limitations, there are a few things worth mentioning. Since the cameras are assumed to be above the touchpad surface facing downward, gestures that require fingers to point straight down cannot be recognized by the hand tracker. While such gestures are simple to detect on devices such as SmartSkin [15], the vision system still affords other features such as multi-layer gesture recognition and accurate finger orientations. We are currently attempting to combine an actual touch sensitive surface (a Mitsubishi DiamondTouch) with our Visual Touchpad to leverage the benefits of both devices, and preliminary observations are promising.

For large wall displays we mentioned that a hand-held Visual Touchpad would be appropriate when a direct manipulation interface is desired. Our current two-handed interaction techniques do not work well for such a setup since the ND hand is required to hold the actual touchpad. Instead, for such large wall interaction it would be more appropriate to investigate two-handed techniques that allow the ND hand's four fingers to hold the panel from behind while the thumb rests on the touchpad surface from the front. The thumb could then be "twiddled" much like a gamepad on a video game console, but in an asymmetric-dependent manner with the D hand.

6. CONCLUSION

In this paper we presented the Visual Touchpad, a low-cost vision-based input device that allows for fluid two-handed gestural interactions much like those available on more expensive tabletop displays or touch-screens. While we demonstrated the capabilities of the Visual Touchpad in a simple image manipulation application, there are still many possibilities for new two-handed interaction techniques that further exploit the strengths of the system.

7. ACKNOWLEDGEMENTS

We thank Ravin Balakrishnan, Allan Jepson, and Abhishek Ranjan from the University of Toronto for valuable discussions.

8. REFERENCES

[1] Buxton, W. (1992). Telepresence: integrating shared task and person spaces. In *Proceedings of Graphics Interface*, pp. 123-129.

[2] Chen, X., Koike, H., Nakanishi, Y., Oka, K., Sato, Y. (2002). Two-handed drawing on augmented desk system. In *Proceedings of Advanced Visual Interfaces (AVI)*. pp. 219-222.

[3] Corso, J., Burschka, D., Hager, G. (2003). The 4D Touchpad: Unencumbered HCI with VICs. *In Proceedings of CVPR-HCI.*

[4] Faugeras, O., Luong, Q. (2001*). The Geometry of Multiple Images*. The MIT Press.

[5] Guiard, Y. (1987). Asymmetric Divison of Labor in Human Skilled Bimanual Action: The Kinematic Chain as a Model. Journal of Motor Behavior, 19(4). pp. 486-517.

[6] Hardenberg, C., Berard, F. (2001). Bare-hand Human-computer Interaction. In *Proceedings of ACM Workshop on Perceptive User Interfaces (PUI).*

[7] Ishii, H., Kobayashi, M. (1992). ClearBoard: a seamless medium for shared drawing and conversation with eye contact. In *Proceedings of ACM CHI Conference*. pp. 525-532.

[8] Kabbash, P., Buxton, W., Sellen, A. (1994). Two-Handed Input in a Compound Task. In *Proceedings of ACM CHI Conference*. pp. 417-423.

[9] Kolsch, M., Turk, M. (2002). Keyboards without Keyboards: A Survey of Virtual Keyboards. Technical Report 2002-21. University of California, Santa Barbara.

[10] Krueger, M. (1991). VIDEOPLACE and the Interface of the Future. In *The Art of Human Computer Interface Design*. Addison Wesley, Menlo Park, CA. pp. 417-422.

[11] Kurtenbach, G., Fitzmaurice, G., Baudel, T., Buxton, B. (1997). The Design of a GUI Paradigm based on Tablets, Two-hands, and Transparency. In *Proceedings of ACM CHI Conference.* pp. 35-42.

[12] MacCormick, J., Isard, M. (2000). Partitioned sampling, articulated objects, and interface-quality hand tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, Volume 2. pp. 3-19.

[13] Mysliwiec, T. (1994). FingerMouse: A Freehand Computer Pointing Interface. Technical Report VISLab-94-001, University of Illinois Chicago.

[14] Oka, K., Sato, Y., & Koike, H. (2002). Real-time tracking of multiple fingertips and gesture recognition for augmented desk interface systems. In *Proceedings of IEEE Conference on Automatic Face and Gesture Recognition (FG).* pp. 429-434.

[15] Rekimoto, J. (2002). SmartSkin: An Infrastructure for Freehand Manipulation on Interactive Surfaces. In *Proceedings of ACM CHI Conference*. pp. 113-120.

[16] Ringel, M., Berg, H., Jin, Y., and Winograd, T. (2001). Barehands: Implement-Free Interaction with a Wall-Mounted Display. In *Proceedings of ACM CHI Conference Extended Abstracts*. pp. 367-368.

[17] Roussel, N. (2001). Exploring new uses of video with videoSpace. In *Proceedings of the IFIP Conference on Engineering for HCI*, Volume 2254 of Lecture Notes in Computer Science, Springer. pp. 73-90.

[18] Segen, J., & Kumar, S. (1998). GestureVR: Vision-based 3D Hand Interface for Spatial Interaction. *In Proceedings of the Sixth ACM International Conference on Multimedia.* pp. 455-464.

[19] Stotts, D., Smith, J., and Jen, D. (2003). *The Vis-a-Vid Transparent Video FaceTop*. In *Proceedings of ACM UIST.* pp. 57-58.

[20] Tang, J. & Minneman, S. (1991). Videowhiteboard: video shadows to support remote collaboration. In *Proceedings of ACM CHI Conference*. pp. 315-322.

[21] Trucco, E., Verri, A. (1998). *Introductory Techniques for 3-D Computer Vision*. Prentice-Hall.

[22] Ukita, N., Kidode, M. (2004). Wearable Virtual Tablet: Fingertip Drawing on a Portable Plane-Object using an Active-Infrared Camera. In *Proceedings of the International Conference on Intelligent User Interfaces*. pp. 169-176.

[23] Wellner, P. (1993). Interacting with Paper on the DigitalDesk. *Communications of the ACM*, 36(7), July 1993. pp. 86-96.

[24] Wu, M., Balakrishnan, R. (2003). Multi-finger and Whole Hand Gestural Interaction Techniques for Multi-User Tabletop Displays. In *Proceedings ACM UIST.* pp. 193-202.

[25] Yee, K. (2004). Two-Handed Interaction on a Tablet Display. In *Proceedings of ACM CHI Extended Abstracts*. pp. 1493-1496.

[26] Zhang, Z., Wu, Y., Shan, Y., & Shafer, S. (2001) Visual Panel: Virtual Mouse, Keyboard, and 3D Controller with an Ordinary Piece of Paper. In *Proceedings of ACM Workshop on Perceptive User Interfaces (PUI).*