A Virtual Touchscreen with Depth Recognition

Gabriel Hartmann **Burkhard C. Wünsche**

Graphics Group, Department of Computer Science
University of Auckland, New Zealand,
Email: gabriel.hartmann@gmail.com, burkhard@cs.auckland.ac.nz

Abstract

While touch interfaces have become more popular, they are still mostly confined to mobile platforms such as smart phones and notebooks. Mouse interfaces still dominate desktop platforms due to their portability, ergonomic design and large number of possible interactions. In this paper we present a prototype for a new interface based on cheap consumer-level hardware, which combines advantages of the mouse and touch interface, but additionally allows the detection of 3D depth values. This is achieved by using a web cam and point light source and detecting hand and shadow gestures in order to compute 3D finger tip positions. Our evaluation shows that the concept is feasible and more powerful interactions than with traditional interfaces can be achieved. Limitations include a reduced input precision, insufficient stability of the utilised computer vision algorithms, and the requirement of a stable lighting environment.

Keywords: touch interface, gesture recognition, stereo vision, shadow detection

1 Introduction

The past decade has brought a remarkable shift in the way humans interact with computers. For the first time the traditional keyboard and mouse interfaces have a real alternative in touchscreen and motion based controls. This shift has coincided with the explosion in popularity and unprecedented affordability of mobile devices and MEMS accelerometers, gyroscopes and magnetometers.

So far, touch and motion based interfaces have been almost universally confined to mobile devices, public displays, entertainment systems, and commercial devices. Touchscreens have been present on commercial devices in the form of point of sale devices, automatic teller machines, and similar equipment for some time. In consumer entertainment systems, the Nintendo Wii was the first device to embrace motion based input and see wide spread adoption. It was quickly followed by competing products from Microsoft with its Kinect, and Sony with its Move. In mobile devices Apple's iPhone was the first widely adopted device which employed only touch and motion based controls. The concepts have been quickly adapted by a slew of competitors, and a similar development can be observed for the iPad, the first commercially successful notebook using these interface technologies.

Although the above devices are used widely, mostly by non-specialist users, the employed interfaces have not found their way back to the personal computing desktop platform. One reason for this lack of feature migration has to do with the excellence of the current system. The mouse and keyboard combination is an extremely versatile system that is well understood and has been used happily by millions of people for decades. The mouse in particular holds some serious advantages over the touchscreen interface on the desktop:

- The mouse cursor obscures less screen space than a finger or stylus.
- Resting the hand on a mouse is more comfortable for complex interactions than lifting the hand and touching a screen.
- Mouse motions can be scaled in order to cover large screen space with small physical movements.
- The screen is not soiled by dirt and oil from the user's fingers.
- A mouse is highly portable and can be used with different computers.
- Traditional display devices with a mouse are usually more affordable than touch screens.

The mouse, however, is not a perfect input device. One major shortcoming is its limitation to a single point of interaction. The use of two mice has been explored in academia since more than a decade (Latulipe et al. 2006, Gonzalez & Latulipe 2011). In contrast to multi-touch input this functionality is not widely supported and few real-world applications exist. One of them is virtual surgery simulation, where the two mice control different instruments used by the surgeon (SIMTICS Ltd. 2011).

Mice are also limited in their repertoire of actions. The single-click, double-click, and drag operations represent the main functionalities of a mouse. Additional functionalities have been added in the form of multiple mouse buttons and scroll wheels, and by combining mouse interactions with keyboard interactions.

Some operating systems and applications have begun to emulate touch and motion screen interactions via the mouse. For example Windows 7 will minimise all other windows if another window is dragged back and forth repeatedly, emulating a shake. The Opera web browser employs a large variety of optional mouse gestures which can control the web browsing experience. The attempts to incorporate these sorts of interaction into the desktop experience highlight the fact, that there are interactions that touch and motion interfaces provide that are desirable on the desktop, but cannot be accomplished with the mouse.

In this paper we explore whether it is possible to produce an interface which combines some of the advantages of the mouse and touch screen, is portable, uses cheap consumer-level hardware, and has additional capabilities neither the mouse nor touch screen offers.

Section 2 analyses the problem, proposes a solution concept, and derives design requirements for it. Section 3 reviews the literature in this field and discusses relevant technologies. The design of our solution and implementation details are explained in sections 4 and 5. Section 6 evaluates our solution, compares it with our original goals, points out limitations, and presents an informal user study. We conclude this paper in section 7 and section 8 gives a short overview of future work.

2 Problem Analysis and Design Requirements

2.1 Problem Analysis and Solution Proposal

Two major disadvantages of touchscreens are their cost and that interface and display device can not be separated, i.e., reuse and portability is limited. We therefore propose to use as interaction space any flat surface, e.g., the user's desk. In order to mimimise hardware requirements the interaction device is the user's hand. Hand tracking is a well researched subject, but is still technologically challenging, and the best results are achieved by using stereo vision (two cameras) or markers. Since many computers already have a single inbuilt web cam, we will only require one such camera, which is oriented to view the interaction space, e.g., the desk. In order to achieve more information about the users hand gesture and position we use a point light source, which casts a shadow onto the interaction surface.

We propose to track the hand and its shadow in order to obtain 3D locations. This corresponds to the mouse drag operation, but additionally adds a third dimension given by the hand's height over the interaction surface (e.g., desk). Different interaction modes, defined for the mouse by different mouse button and simultaneously pressed keyboard keys, are achieved by performing a limited gesture recognition.

2.2 Design Requirements

We assume a low specification web-cam with a minimum resolution of 640x480 pixels. Calibration of both the desktop surface and the camera is achieved using a calibration grid, which can be printed out by any printer (although a laser printer is preferable). Our solution should have a low computational complexity, such that it can be used even on entry-level desktop machines without advanced CPU/GPU. Finally we require a single point light source which illuminates the interaction surface. For illumination a lamp with a light bulb is sufficient, as long as it is positioned above the users and preferably on the side. Examples are most ceiling lights (depending on the desk position) or a clip-on desk lamp.

In terms of functional requirements the aim of our research is to provide device level support for touch and motion controls. The interpretation of device output is not a concern. For example, we are interested in creating 3D coordinates reflecting the users interactions, but we leave the interpretation of these coordinates, e.g., as mouse click or mouse drag, up to the application developer.

Hence we can summarise the goals of our research as follows: We want to create a system using a web cam, point light source and flat illuminated surface in order to determine 3D positions for hands and fingertips above and on the desktop surface. We want to be able to detect touches on the surface, but we do not try to emulate a multi-touch interface. The goal of the research is to investigate the feasibility of the described configuration as an inexpensive 3D interaction interface for a desktop platform.

3 Literature Review

Hand based computer interaction is a popular research area due to its intuitiveness and promise to enable true 3D interactions.

In situations where the accuracy and speed of hand tracking is important data gloves are considered the best choice. The devices usually employ electro-mechanical, electro-optical or magnetic sensing technologies. They are application independent and single purpose, devoted entirely to hand tracking (Erol et al. 2007). Data gloves produce real-time results and are able to capture all the degrees of freedom a human hand has to offer. However, they are expensive, require precise calibration, and might hinder natural hand motion to some degree.

Computer vision based hand tracking approaches can overcome the expense, inconvenience, and unnaturalness of data gloves. Cameras for computers are now widely available and inexpensive, and the contact-free image-based sensing does not interfere with hand motions. However the difficulty of 3D hand pose estimation has so far resisted attempts to produce results which are comparable to those produced by data gloves.

The main difficulties of computer vision based hand pose estimation are high-dimensionality, self-occlusion, high computational requirements, uncontrolled environments, and rapid hand motion (Erol et al. 2007). High-dimensionality refers to the fact that the human hand has over 20 degrees of freedom without taking into account the position and orientation of the hand as a whole (Erol et al. 2005). Self-occlusion occurs when fingers and/or palm overlap on the two-dimensional image of a camera. Uncontrolled environments can result in arbitrarily complex backgrounds and unpredictable lighting scenarios. Rapid hand motion refers to the hardware and software limitations that constrain the ability of applications to track rapidly moving hands. All the above points make complex algorithms necessary to detect, track and disambiguate hand configurations, which leads to a high computational complexity.

Each of the difficulties described is non-trivial to solve individually. Taken together it is unsurprising that no general solution for computer vision based 3D hand pose estimation and tracking exists. All attempts so far make assumptions which eliminate or mitigate some of the difficulties.

3.1 Two Dimensional Fingertip Tracking

Hand tracking can be simplified by only determining certain key features such as the finger tips. Hardenberg & Bérard (2001) use the position of fingertips and the number of detected fingers, in order to define pointer positions and different commands akin to mouse clicks. Fingertips are detected by employing an image segmentation algorithm and fitting circles to foreground image sections. The problems of self-occlusion are ignored with the tacit assumption that as long as fingertips are sufficiently visible to be identifiable, operations can continue. For fast hand motions finger tip positions are not individually identified but estimated from previous frames.

The authors present four proof-of-concept applications. Three of the applications replicate mouse functionalities, with the fourth allowing for multiple input positions. The applications highlight some difficulties of the system. As it is entirely a 2D system no touching of the interacting surfaces is detected. Instead, the authors' web browsing and painting application represents mouse clicks by maintaining a hand position for one second. The system also supports simple finger counting gestures for a slide presentation application. These gestures have no movement component or intuitive relation to their function. For example, displaying two fingers indicates a need to move to the next slide, and three fingers indicate a move to the previous slide. This requires memorization of the slide application commands by the user. No attempt is made to actively deal with the problem of misclassifying shadows of fingertips as actual fingertips. The authors' experimental results showed that this error was not uncommon.

Song et al. (2007) improve Hardenberg and Bérard's algorithm by additionally taking the finger shape into consideration, which is done by segmenting and classifying the pixel region connecting the finger tip to the palm. Laptev & Lindeberg (2001) use particle filtering and multi-scale image features for finger tip detection and tracking. Terajima et al. (2009) use a template approach, which also gives some 3D information. Hsieh et al. (2008) use finger tip detection for handwriting recognition. The finger tip motion is obtained from frame differences. The search for a new finger position is sped up by predicting its position using a Kalman filter.

3.2 Surface Touch Detection from Camera Images

Malik & Laszlo (2004) use stereo cameras to detect touch interactions with the underlying surface and this way construct a virtual touchpad. An important constraint is, that the background must be black and rectangular. This improves calibration and avoids problems with shadows.

Detecting 3D motion with a single camera is more difficult. Wang et al. (2007) capture hand motion with a single camera in real-time by combining a model-based and appearance-based method. Interference among fingers is resolved using k-means clustering and particle filters.

Segen & Kumar (1999) simulate touch interactions by using additionally a single point light source, whose position is determined in a calibration step. The surface upon which the light source casts shadows must also be defined, although this step is not explicitly described in the paper. Instead the camera's position is defined in terms of the surface plane which is defined as the $Z = 0$ plane. Segen and Kumar do not attempt to detect surface touches and indicate use of a uniform background. They find fingertips and finger orientation with exactly the same method employed by Malik & Laszlo (2004). The orientation of fingers is defined by the line which extends through the fingertip and the midpoint of the end points of the two vectors which designated it as such.

Much of this 3D information is not employed in their proof-of-concept applications. Actions are almost entirely defined by static gestures, with the exception of a "clicking" gesture which employs a bent finger motion. This gesture requires a stationary hand at the time of its execution, which is perhaps problematic in the suspended hand positions indicated by the authors. Once a gesture is detected, the pose of the hand in space is used to modify the application of the gesture's command. For example, a two finger gripping gesture can be used to manipulate a virtual robot arm. The arm is oriented according to the lines defining the fingers. The distance between the gripping portions of the robot hand depends on the distance between the two fingers in the gesture. Using three fingers to grab a virtual object is however impossible as no three finger gesture is defined. Likewise, if two fingers are used, but they are not straight, the gesture will be unrecognised and no input will occur.

3.3 3D Hand Pose Estimation

A large variety of hand tracking algorithms has been proposed. A good survey is given in (Mahmoudi & Parviz 2006). Two important categories are marker-based and marker-less methods.

Marker-based hand tracking algorithms require the user to wear point or area markers (Park & Yoon 2006, Lee & Woo 2003, Wang & Popović 2009). Robustness and interactive speed are achieved by employing simplified hand models in order to resolve ambiguities in the tracked marker positions. However, the need for auxiliary devices (markers, gloves) can be inconvenient for the user and often requires some type of calibration, e.g., to align marker positions with positions on the underlying hand model.

Without markers the easiest way to identify (potential) hand shapes is by using a skin colour classifier (Kakumanu et al. 2007, Vassili et al. 2003). Sensitivity to changes in the illumination can be reduced by using a perception-based colour space (Chong et al. 2008). 3D position and orientation of the hand shape can be obtained by using a 3D hand model and searching for a mapping between it and the perceived hand shape subject to the model's inherent constraints (e.g., joint constraints and rigidity of bones) (Stenger et al. 2001, 2006). A different approach is to perform the matching between hand features rather than the entire shape (Chen et al. 2007).

Single camera systems suffer from the high computational complexity and a limited robustness. They work best if the range of possible hand motions is constrained (Liu 2010, Liu et al. 2011). Tracking can be simplified by using depth information obtained using multi-camera vision or stereographic systems (Argyros & Lourakis 2006).

4 Design

In order to simplify the design we assume that the background is stable (e.g., a desk surface rather than an office with people moving around) and that only one hand and its shadow must be tracked. The techniques presented in this section will also work for two hands, as long as the hand images and shadows don't overlap. However, this extension was deemed unnecessary for our proof-of-concept application. The tracking of the hand shape and its shadow requires the following three tasks to be performed for each frame: segmentation, feature detection, and feature position estimation.

4.1 Segmentation

Segmentation determines the objects of interest and separates them from the background. We assume a static background, such as a desk surface, and a static lighting environment using a single point light source, e.g., a desk or ceiling lamp.

4.1.1 Background Classification

Segmentation is achieved by first computing a statistical model of the background. Since the user has control over the environment we can assume that no foreground objects, such as skin coloured objects and shadows, are in the field of view. We compute for each pixel an average value and average difference, which approximates the standard deviation but is faster to compute (Bradski & Kaehler 2008). For the computation of both statistics the three RGB channels remain separated. The statistics are used to define for each pixel a range of colours considered to represent the background. The threshold values representing valid background pixels were determined experimentally in order to optimise correct classifications (see section 5). All pixels outside this range are considered foreground. The statistical model is necessary since pixel colours change between frames even for static environments. Causes are pulsed light sources such as energy saving (fluorescent) lights, slight vibrations, and camera specific issues such as noise.

4.1.2 Hand and Shadow Identification

The second step of the segmentation process divides the foreground pixel colour histogram into regions representing the hand and its shadow (see figure 1). This is achieved by observing that shadows change predominantly the "Value" of the HSV colour of the object onto which they are cast, whereas the hue and saturation show little variation. Since we restrict ourselves to one-handed interactions, the shadow is always cast onto the interaction plane. We hence compare for each pixel of the foreground region its colour with the pixel's learned average background colour. This comparison is performed in the HSV colour space. If the difference occurs largely for the value (V) component of the pixel, and is accompanied by a slight decrease in the saturation (S) channel, then the pixel is considered to be part of the hand shadow. All other foreground pixels are considered to be part of the hand region. All thresholds for these test were determined experimentally.

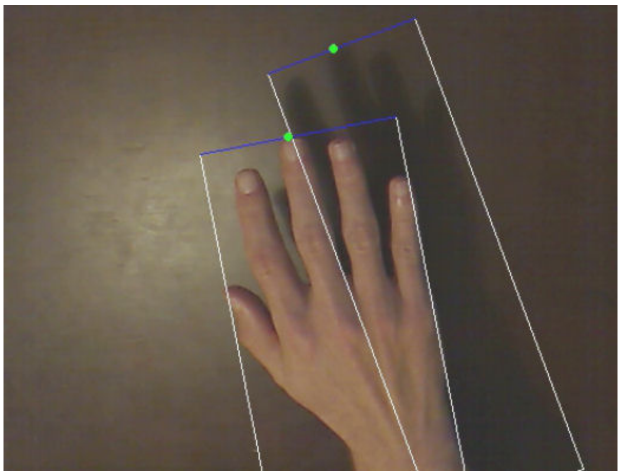


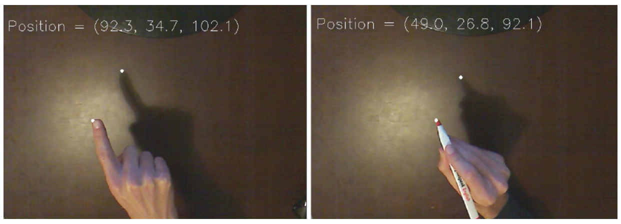
Figure 1: The segmentation process separates foreground pixels into hand regions and shadow regions. The tips of each region are determined by computing an oriented bounding box for the largest connected component of each region, and selecting the point closest to its top-most edge.

4.2 Finger Tip Detection

In order to define meaningful interactions finger tips must be detected. The finger tip location can be used to select points, draw shapes, or define different interaction modes according to the relative position of multiple finger tips.

4.2.1 Single Feature Point Detection

A single feature point for pointing and selection operations is defined by computing the minimum oriented bounding box of the contour of the hand region. We then determine the edge nearest to the tip of the hand. Since the user usually sits opposite to the web cam, and since the hand and wrist usually have a smaller width than length, we use the short edge of the bounding box furthest from the bottom of the camera image. The feature point for selection operations is then given by the point on the contour closest to this edge. This algorithm gives a meaningful result for whole hand gestures (in which case the tip of the middle finger would be the feature point), as well as for single finger gestures, such as using the index finger for selection, or when holding a pointing device such as a pen. Figure 1 and 2 illustrate these three cases both for the hand and the shadow region.



Position = (92.3, 34.7, 102.1)

Position = (49.0, 26.8, 92.1)

Figure 2: A selection/drawing operation using a single feature point defined as tip of the index finger (left) or tip of an external pointing device such as a pen (right).

4.2.2 Multiple Finger Tip Detection

Detecting multiple fingertips and their shadows is a three-step process. The first two steps of the process correspond to the popular convex hull and convex defect detection approach (Homma & Takenaka 1985, Liu et al. 2011). The final step is the removal of defects irrelevant to fingertip detection.

In the first step a convex hull is placed around the hand segment. Defects are defined in relation to this hull. Any region within the convex hull which is not part of the hand segment and which is adjacent to an edge of the convex hull is considered a defect. If this algorithm is applied to a hand with spread fingers, six defects are commonly detected. Four of these defects lie between the fingers and the other two lie between the smallest finger and the wrist and between the thumb and the wrist. All but two of the endpoints of the defect regions coincide with fingertip positions.

In order to remove endpoints which do not correspond to fingertips, we use the observation that for convex defects between fingers their depth is larger than their width (Liu et al. 2011). Figure 3 gives an example. Note that this algorithm only works as long as the fingers remain approximately straight. Fingers which are bent excessively can cause their defining defect to be rejected and their fingertip position lost as a result.

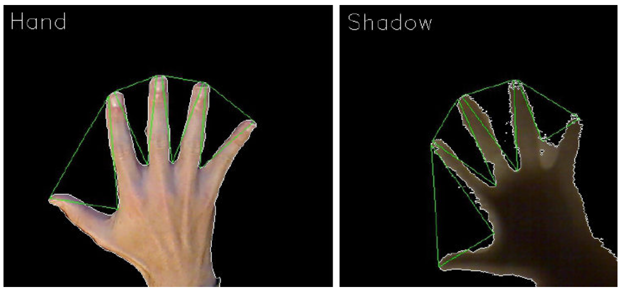


Figure 3: Finger tip detection by computing convex defects of the convex hulls of the hand (left) and of the shadow region (right).

4.3 Calibration for Depth Calculations

Accurately determining fingertip positions in three dimensions requires a one-off calibration step defining the relative positions and orientations of the camera, desktop surface, and light source.

4.3.1 Camera Calibration

In order to simplify the subsequent computation of 3D positions we would like to have an idealised camera. i.e., a "perfect pinhole camera". A pinhole camera does not have a lens and hence does not introduce radial distortion. The image plane of a pinhole camera is exactly perpendicular to the optical axis and so tangential distortion which can be introduced by an imprecise alignment of optical axis and image plane is absent. A pinhole camera produces images which are a perfect projection of the objects in the world onto the image plane. Its well defined geometry aids in the determination of the position of objects in the world and is therefore desirable in computer vision applications.

In our application we assume that users utilise cheap consumer-level web cams. We hence have to correct inherent distortions, which is achieved by imaging a known object and comparing properties derived from the image (e.g., distances and angles between lines) with the known ones. We use the popular chessboard calibration grid shown in figure 4, since open source software for the calibration is available, and since it allows us to define the interaction plane as a by-product of the camera calibration.

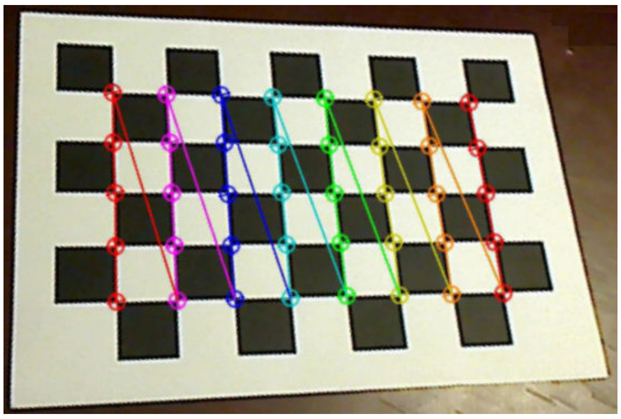


Figure 4: An image of a chessboard pattern of known dimensions is sufficient to characterise the distortion characteristics of the camera, and the camera's location in reference to the surface upon which it is placed. Furthermore corner points of the chessboard pattern can be used to define the surface plane equation.

4.3.2 Desktop Plane Definition

The definition of the interaction (desktop) plane is achieved using the calibration grid in figure 4. The thickness of the paper is considered negligible and a planar homography between the points on the calibration grid and the imaging plane is defined as a series of translations and rotations. The relative positions of the focal point of the camera, and three points on the calibration grid are sufficient to obtain a unique solution. However, in order to deal with inaccuracies due to noise we use all points of the calibration grid. An optimal solution for the plane equation is found by using a least square method, which minimises the distances of the points to the proposed plane.

4.3.3 Light Source Estimation

Given the intrinsic and extrinsic parameters of the camera and the equation of the desktop plane in relation to the camera coordinate system it is possible to unambiguously determine the positions of shadow pixels in the interaction plane. If we know two 3D points and the locations of their shadows on the interaction plane, we can compute the light source location as the intersection of the lines formed by each point and its shadow. If the lines are parallel then the light source is at infinity (e.g., when using sun light in an outdoor application).

In order to avoid the need for special 3D objects for calibration, we use a series of predefined hand positions of the user. The first position is that of the hand with fingers close together, placed flat on the desk (see figure 5). The second posture is a hand made into a fist and with only the index finger making a pointing gesture. The fist should be placed flat against the desk (see figure 6). The height h of the finger tip of the index finger relative to the interaction plane is assumed to be 80% of the width of the bounding box of the hand position in figure 5. This approximation was motivated by the fact that the index finger is the fourth finger of the hand, and it gave good results in our experiments.

By sliding the hand around in the second configuration, as illustrated in figure 6, we obtain a sequence of 3D points and corresponding shadow positions, which we can then use to estimate the light source position by using the closest intersection point of all rays or using a light source at infinity if the rays are approximately parallel.

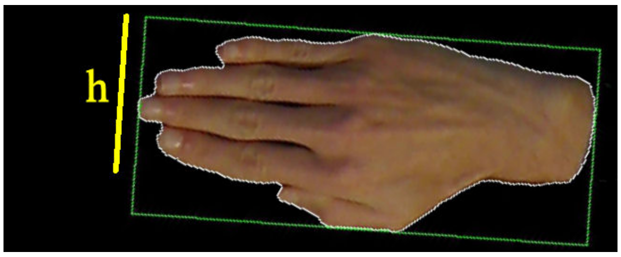


Figure 5: A bounding box is fitted to the segmented hand in order to determine the width of the hand, and the height h of the index finger above the interaction plane for the light calibration step.

4.4 Computation of 3D Finger Tip Positions

3D positions of objects can be computed from their 2D position on the image plane and their shadow position using calculations similar to epipolar geometry in stereo vision. In our case the light source and interaction surface provide similar information as the second

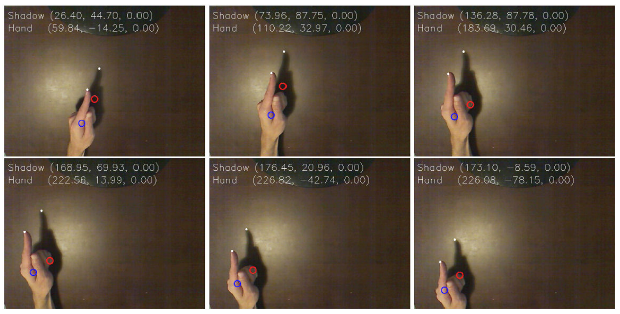

Shadow (26.40, 44.70, 0.00)
Hand (59.84, −14.25, 0.00)
Shadow (73.96, 87.75, 0.00)
Hand (110.22, 32.97, 0.00)
Shadow (136.28, 87.78, 0.00)
Hand (183.69, 30.46, 0.00)
Shadow (168.95, 69.93, 0.00)
Hand (222.56, 13.99, 0.00)
Shadow (176.45, 20.96, 0.00)
Hand (226.82, −42.74, 0.00)
Shadow (173.10, −8.59, 0.00)
Hand (226.08, −78.15, 0.00)

Figure 6: The light source is estimated as the intersection of the lines formed by the tips of the index finger and the corresponding shadow points. The shadow points lie in the interaction plane and the finger tip height above the plane is determined by placing the hand flatly onto the interaction plane as illustrated in figure 5.

camera used in stereo vision. The set-up would not be suitable for general applications due to problems with shadows being frequently occluded. However, for touch surface style interactions our configuration provides the required information since users sit opposite to the camera, the light source is on the side, and the hand is usually approximately flat. The 3D finger tip position O is calculated as illustrated in figure 7: A line is cast from the view point position V through the shadow's position S' on the view plane. The intersection of this line with the interaction plane (obtained in the calibration step) is the shadow's position S. The unknown 3D position of the point O is given by the intersection of the line from S to the light source L and the line from the view point V through O', the projection of O onto the view plane.

In our application the points O and S are the finger tip positions and their shadows, which are calculated as explained in subsection 4.2. Note that the above calculation can be applied to any object for which a corresponding shadow can be identified. For example, figure 2 shows the use of a pen as interaction device.

Once a point can be tracked in three dimensions, the detection of touches can be defined by specifying a threshold for the distance between the point and the interaction plane. This is necessary, since a finger has a non-zero thickness. The threshold for this was determined experimentally.

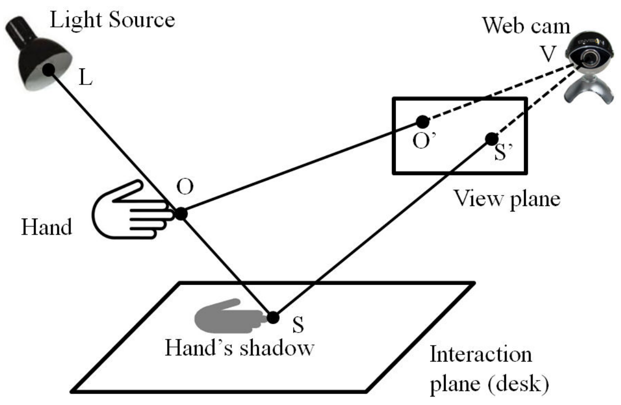


Figure 7: The setup of our virtual touchscreen. A light source illuminates the hand, which casts a shadow onto the interaction plane. The position of a 3D point O can be computed using the light source position L, the web cam (view point) position V, and the projections O' and S' of O and S, respectively, onto the view plane.

5 Implementation

Our application was developed in C/C++ using the OpenCV library (OpenCV 2011) for camera calibration, colour space conversion, pixel colour averaging, and image segmentation. We also used the "Geometric Tools" library (Eberly 2011), particularly for the definition of the desk plane.

5.1 Segmentation

For image segmentation we used the OpenCV functions `cvAcc` and `cvAbsDiff` to accumulate and calculate absolute differences for images. The results of these operations are image data structures (`IplImage`) with the same dimensions as the input images. The computation of averages is accomplished by using `cvConvertScale` to divide by the number of captured images.

All subsequent threshold values refer to 8 bit numbers (0 to 255) for the corresponding colour channels. In order to place thresholds above and below the average pixel values, the average difference values are scaled and subtracted/added to the average. Typical scaling values for setting the thresholds above and below the average are 7.0 and 6.0 respectively, resulting in a range of approximately a 2.5% on either side. Any value falling outside these thresholds is considered foreground and anything which falls between these thresholds is considered background. For the shadow detection RGB colours were converted into HSV colours using `cvCvtColor`. Typical thresholds for the value and saturation channels were 5.0 and 2.0, respectively.

Pixel classification was achieved using the `cvInRange` function, which outputs monochromatic images, which can be used as masks for subsequent operations. For example, inverting the shadow mask using `cvNot` and combining it with the background mask using `cvAnd` creates a mask for the hand (all pixels which are not background and not shadow).

Noise in the image data resulted in some pixels being misclassified. We removed it by applying a Gaussian smoothing with a 9x9 window to the hand mask and shadow mask. Holes in the hand region were filled using a morphological operator.

5.2 Computation of 3D Finger Tip Positions

In subsection 4.4 we explained that the 3D position of finger tips is defined as the intersection of two rays. Due to noise, calibration errors, and numerical inaccuracies the two rays are unlikely to intersect in practice. Instead we use an algorithm from Paul Bourke (Bourke 2011) to find the point where the distance between the two rays is minimal.

Given the 3D positions of points of interest and the definition of the desk plane touch interactions are defined by instances where the distance between 3D point and shadow point falls below a given threshold. We determined experimentally that using a threshold of 10mm gave best results, which represents approximately the thickness of a finger. We use a coordinate transformation matrix for easy conversion between the camera and desk coordinate system.

6 Results

We evaluated our prototype by investigating the influence of different parameters, by creating three proof-of-concept applications, and by an informal user test. We compared the interface to traditional mouse and touch interfaces and to the "Shadow Gestures" paper, the closest related research.

6.1 Experimental Setup

As light source we used an ordinary ceiling light with an $11W$ fluorescent bulb. This is not a perfect point light source and resulted in a soft shadow. The curtains of the room were closed resulting in a low ambient light, without specular reflections from sunlight. We used a Logitech QuickCam Pro 9000 web cam with a resolution of 640x480 pixels. The configuration of the camera, desk, and light source for development and testing is shown in figure 8. The interaction surface was a portion of a desk, which was slightly glossy and prone to some measure of reflection. It also produced a sizable highlight as shown in figure 2. Tests with more matte surfaces improved results, but adding constraints in form of allowable surface properties was deemed too restricting for practical applications.

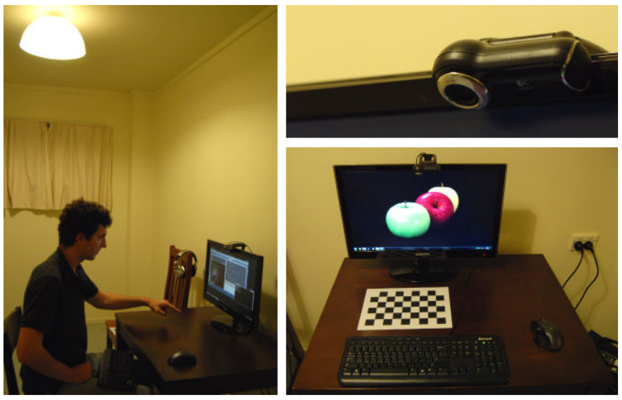


Figure 8: The virtual touch screen set-up employed during development. Left: The positions of the point light source, desk and camera. Right-bottom: The interaction surface (desk) with camera and calibration grid. Right-top: The orientation of the camera with respect to the monitor and desk.

All computations were performed on a PC with an Intel Core i5 750 2.67 GHz CPU, 4 GB 1333 MHz RAM, and an Nvidia GeForce GTX 460 graphics card with 768 MB GDDR5 and 336 CUDA cores. Note that no hardware acceleration was used for the computer vision algorithms and that the CUDA toolkit for OpenCV was not installed. Hence the performance on other systems should scale approximately linearly with CPU performance. With the above configuration we achieved real-time performance of above 20 frames per second.

6.2 Influence of Parameters

6.2.1 Thresholds

Initial tests showed that the utilised segmentation technique worked well over non-uniform static backgrounds for stable lighting environments. The threshold values described in the previous sections required careful tuning. We developed an interface for simplifying this task, but it is still a manual process, which needs to be repeated if the setup is changed. Ideally this process should be automated, e.g. by performing a sequence of predefined hand postures and gestures.

6.2.2 Environmental Parameters

The background needs to be static and hence can not be used in environments where people or objects are moving around, which might partially obscure the background or change the lighting environment (e.g., cast a shadow on the desk). We haven't developed applications with two-hand input, but as long as the hands and their shadows are separated, we can't see any reason why this should not work. Our model assumes that the foreground colours vary from the background colour, hence it would not work for setups where the desk and user's skin colour are similar or where the background's material does not show visible shadows.

The lighting environment needs to be a point light source. We found that the application still works for a ceiling lamp generating slightly soft shadows, but it would not work for large fluorescent lights. The algorithms were designed such that the application would also work in an outdoor environment using the sun as point light source. In this case the light source would be positioned at infinity resulting in an orthogonal shadow projection. This will have to be tested in future work.

6.2.3 Calibration Parameters

The calibration of the desktop plane uses well-tested algorithms and no problems were noticed. The calibration of the light source, described in subsection 4.3.3, uses several simplifications. The fact that the height h in figure 6 is only a rough estimation of the actual height introduces errors when computing the depth values from the positions of the fingers and theirs shadows. However, this was not considered a problem, since the errors do only effect actual depth estimates. For the applications we tested, such as moving or extruding an object in 3D, relative height changes are sufficient. In order to see why, note that the traditional mouse interface uses distance scaling dependent on accelerations (Microsoft Corporation 2002). This is considered intuitive despite resulting in a non-linear behaviour and the same physical mouse position during interactions corresponding to different screen positions. Our tests indicate that the main factors for usability are that the user can perceive 3D cursor motions, their relationship to hand motions, and that their effect on the 3D application is predictable.

6.3 Proof-of-Concept Applications

6.3.1 Single-Touch Interface

Our first proof-of-concept application is a sketch-type interface using the most outward point of the hand as a single interaction point. This can be a finger tip, the tip of the entire hand, or a stylus as explained in subsection 4.2.1 and illustrated in figure 1 and 2.

Figure 9 demonstrates that hand and shadow detection can be used to successfully detect contact with the interaction surface and subsequent finger motions. The hand tip and its shadow on the desk surface are marked with white points. When contact is detected the point colour becomes green. The 3D coordinates displayed in the figure show the 3D position of the interaction point. It can be seen that during contact the depth value can vary slightly due to noise, a changing finger angle, and compression of the finger tip.

Touch detection is sufficiently accurate to draw complex shapes as shown in figure 10. Some jitter is visible partially due to finger motions and partially due to noise in the detected finger position.

6.3.2 Multi-Touch Interface

The second proof-of-concept application explores the possibility of multi-touch interactions using multiple finger tips. Figure 11 shows that as long as the hand shadow is fully visible 3D positions can be tracked for

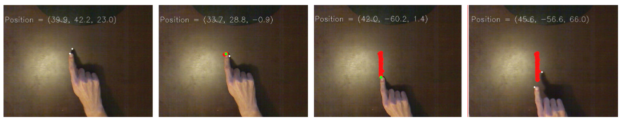

Position = (39.9, 42.2, 23.0)
Position = (33.7, 28.8, −0.9)
Position = (43.0, −60.2, 1.4)
Position = (45.6, −56.6, 66.0)

Figure 9: Examples of a sketch-application. The sketch starts as soon as the finger touches the interaction surface and tracks the finger motion until the contact stops.

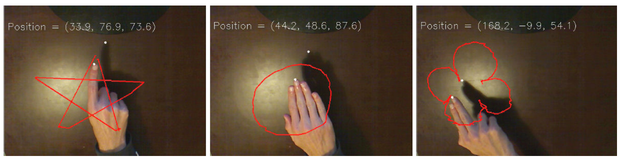

Position = (33.9, 76.9, 73.6)
Position = (44.2, 48.6, 87.6)
Position = (168.2, −9.9, 54.1)

Figure 10: Different shapes drawn using different hand gestures with a single interaction point. Shapes can be drawn by connecting multiple contact points with line segments (left), or by a continuous finger motion (middle and right).

all fingers. The association of finger tips and shadow points is achieved by minimising the distances between each pair of points. If the hand is rotated or the hand overlaps the shadow region the finger tip detection can fail for the shadow region. In that case the 3D position of the corresponding finger tip cannot be computed. Improvements might be possible by using different finger tip detection techniques or by using a hand model and computing a 3D configuration matching both the hand and shadow image.

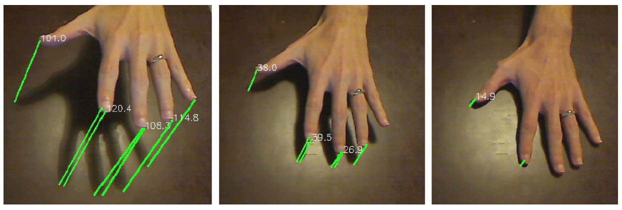


Figure 11: Tracking of the 3D position of multiple fingers (left). The displayed numbers give the height of each finger tip in millimeters. The green lines connect finger tips with the corresponding shadow points. If the hand moves closer to the interaction plane the detection of convex defects fails and the 3D position can not be computed (middle and right).

6.4 3D Modelling Application

The third proof-of-concept application demonstrates a 3D modeling application. 3D extruded objects are created by touching the interaction surface, moving the finger to draw a 2D shape, and extruding it by lifting the hand. For the configuration in figure 8 the maximum possible extrusion height was about 18*cm*. The current application extrudes the raw 2D sketches without modification. Spline curves could be used to smooth the sketch input and OpenGL tessellation algorithm could be used to create a closed 3D shape.

6.5 Informal User Test

We tried our application with five users unfamiliar with the project and aged between 7 and 39 years. The participants were asked to use the sketching application without any explanations or introduction to the program. All users exhibited what appeared to be a high level of enjoyment, and there was some competition among the younger users (7 and 11 years

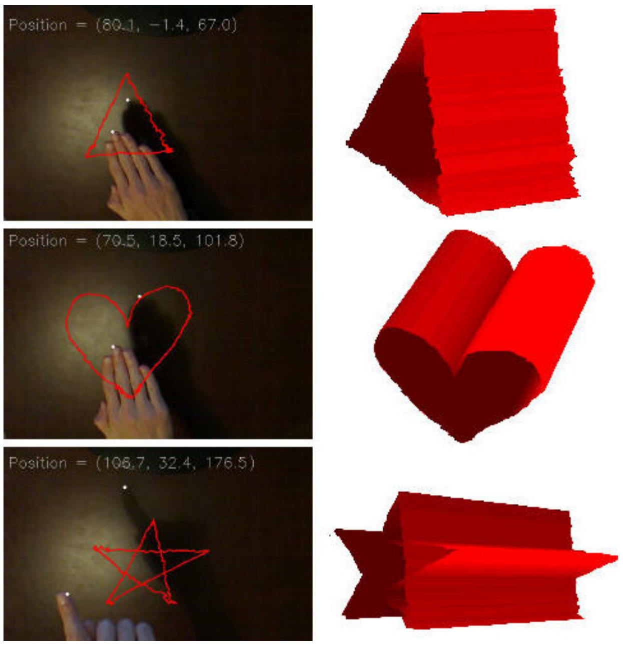

Position = (80.1, −1.4, 67.0)
Position = (70.5, 18.5, 101.8)
Position = (106.7, 32.4, 176.5)

Figure 12: Examples of a 3D modelling application: 3D shapes are created by drawing a 2D shape using a single-finger gesture on the surface, and then extruding it by lifting the hand.

old) for time using the program. Finger tracking performed as expected and described in the previous subsections. Difficulties arose as users assumed multitouch capabilities and multi-user support would work. Furthermore, as multiple users crowded around the desk, blocking of the light source became an issue as standing peoples heads came between the light source and the desk. Also the sometimes inconsistent touch registration on parts of the surface near the centre of the highlight and in the darker regions to the right were found to be frustrating.

Users did not seem to have any intuitive grasp of what areas were likely to respond to touches better than others. User reactions indicated that the most important improvement for future work are multi-touch support and a more robust touch registration.

6.6 Comparison with Traditional Interfaces

The most popular traditional positional input devices are mice and touch screens. Similar to mice our input device does not allow direct interaction with the display, but uses instead a cursor moved by the hand motions. This maximises the visible screen space, but introduces a level of indirection.

In contrast to both mice and touch screens we can input 3D positions. However, we feel that the mouse interface is by far the most ergonomical one. Also the mouse interface seems to be most precise and allows scaling of motions using accelerations. While this could also be implemented for our interface, the implementation is not as straight forward since the start and end of motions would have to be indicated either by hand gestures or surface contacts.

While our interface allows multi-touch interactions in principle, we found this difficult in practice due to self-occlusions. Similar to mice our interface is portable and is independent of the display device. Like most interfaces a smooth motion, e.g. drawing a smooth curve, is difficult without postprocessing. We did not make any formal tests of the achievable precision, although our informal tests indicate that in terms of point selection our interface is less precise than a mouse.

6.7 Comparison with "Shadow Gestures"

We found only one other paper using hand gestures and their shadows for 3D interactions (Segen & Kumar 1999). In contrast to our research the paper puts many more constraints on the environment including a uniform background, pre-calibrated cameras, and a calibration step for the light source, which is not described in the paper. While the authors mention that they compute 3D points, most interactions use 2D information and surface touches are not detected.

In contrast to our goal of extending touch screen interactions, Segen and Kumar use the shapes formed by two fingers to define different gestures. The capabilities of the resulting gesture interface are demonstrated with different example applications, such as manipulating 3D objects or steering a plane through a 3D terrain. This seems to be predominantly achieved by the orientation and shape of the two fingers, rather than computing accurate 3D position. The authors mention that their system fails if the hand moves close to the table.

7 Conclusion

The mouse and keyboard combination of input devices has dominated the desktop computing interface for decades. The explosion of portable computing devices such as iPhone and iPad has made intuitive touch and motion controls popular and they are now slowly finding their way to the desktop platform.

An analysis of the strengths of the mouse and keyboard configuration and the weaknesses of current approaches to touch interfaces motivated a new approach combining the advantages of both of these interfaces. In this paper we attempted to develop such an interface using inexpensive hardware (webcam and light bulb).

We have presented a prototype of a 3D virtual touch screen which uses a web cam to track the position of the user's fingers and their shadows and from this computes 3D points. We showed that the system works well for single touch inputs, but problems exist for multi-touch interactions due to finger shadows being partially occluded.

In contrast to previous work by Segen and Kumar (Segen & Kumar 1999) we provide a more stable tracking technique allowing a wider range of environmental conditions and simpler calibration steps. Most importantly we move away from gestures and towards touchscreen emulation, and hence simulate and extend an input mechanism that is already being widely adopted outside the desktop platform.

Another advantage of our application is that it can be used in combination with a traditional mouse interface. While mouse interactions occur our interface can be disabled, and when the mouse is moved aside (not visible on the interaction surface) the 3D touch capabilities can be enabled.

8 Future Work

Two key categories of work remain ahead. First, segmentation of shadows and hands from the background must be improved. Second, occlusion of relevant shadow information must be dealt with. For shadow detection and background subtraction, a system which automatically adjusts to different backgrounds is necessary. Thresholds based on the lightness or darkness of the background on a per pixel basis should be produced. Additionally improved hand detection is required, e.g., by using skin colour detection (Kakumanu et al. 2007, Vassili et al. 2003, Liu et al. 2011). The tracking and prediction of finger and shadow movements should also be investigated as a method for dealing with transitory occlusion periods.

In previous work we developed "LifeSketch", an application for the sketch-based modelling of interactive 3D environments (Yang & Wünsche 2010, Guan & Wünsche 2011, Olsen et al. 2011, Schauwecker et al. 2011). The interface presented in this paper will significantly simplify many of the presented modelling interactions. For example, we showed that complex buildings can be modelled from 2D outlines, but often additional 3D parameters such as extrusion depth are required (Olsen et al. 2011). Similarly animating 3D objects is cumbersome by using only 2D touch/sketch interactions (Schauwecker et al. 2011).

8.1 Future Applications

The applications so far developed and described in this paper only show a small fraction of what the input mechanism produced here can accomplish. Expansion of application capabilities should be a future priority. In June 2011, Microsoft gave the first official public viewing of Windows 8. The default interface for the world's most popular operating system is being designed from the ground up to be touch-based. Microsoft currently states that a mouse and keyboard will still work for interactions, but the design should create increased demand for low cost desktop based touch interfaces.

Our design works for any configuration using a single point light source, which casts shadows onto a roughly planar surface. We are hence interested to try the system in outdoor environments using the sun as light source. Problems will occur when the sun stands low and the shadows are stretched. Also the sun's position will move over time, which requires occasional recalibration. Two such calibrations should be enough to interpolate and predict further position changes.

References

Argyros, A. A. & Lourakis, M. I. A. (2006), Binocular hand tracking and reconstruction based on 2d shape matching, *in* 'Proc. International Conference on Pattern Recognition (ICPR '06)', pp. 207–210.

Bourke, P. (2011), 'Geometry, surfaces, curves, polyhedra'. `http://paulbourke.net/geometry`.

Bradski, G. & Kaehler, A. (2008), 'Learning OpenCV: Computer vision with the OpenCV library'.

Chen, Q., Georganas, N. & Petriu, E. (2007), Real-time vision-based hand gesture recognition using haar-like features, *in* 'Instrumentation and Measurement Technology Conference Proceedings, 2007. IMTC 2007. IEEE', pp. 1–6.

Chong, H. Y., Gortler, S. J. & Zickler, T. (2008), 'A perception-based color space for illumination-invariant image processing', *ACM Trans. Graph.* **27**(3), 1–7.

Eberly, D. (2011), 'Geometric tools'. `http://www.geometrictools.com`.

Erol, A., Bebis, G., Nicolescu, M., Boyle, R. D. & Twombly, X. (2005), A review on vision-based full dof hand motion estimation, *in* 'Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops - Volume 03', IEEE Computer Society, Washington, DC, USA, pp. 75–82.

Erol, A., Bebis, G., Nicolescu, M., Boyle, R. D. & Twombly, X. (2007), 'Vision-based hand pose estimation: A review', *Journal of Computer Vision and Image Understanding* **108**, 52–73.

Gonzalez, B. & Latulipe, C. (2011), BiCEP: bimanual color exploration plugin, *in* 'Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems', CHI EA '11, ACM, New York, NY, USA, pp. 1483–1488.

Guan, L. & Wünsche, B. C. (2011), Sketch-Based Crowd Modelling, *in* 'Proceedings of the 12th Australasian User Interface Conference (AUIC 2011)', pp. 67–76. `http://www.cs.auckland.ac.nz/~burkhard/Publications/AUIC2011_GuanWuensche.pdf`.

Hardenberg, C. & Bérard, F. (2001), Bare-hand human-computer interaction, *in* 'Proceedings of the 2001 workshop on Perceptive user interfaces', PUI '01, ACM, New York, NY, USA, pp. 1–8.

Homma, K. & Takenaka, E.-I. (1985), 'An image processing method for feature extraction of space-occupying lesions', *Journal of Nuclear Medicine* **26**(12), 1472–1477.

Hsieh, C.-C., Tsai, M.-R. & Su, M.-C. (2008), A fingertip extraction method and its application to handwritten alphanumeric characters recognition, *in* 'Proceedings of the 2008 IEEE International Conference on Signal Image Technology and Internet Based Systems', SITIS '08, IEEE Computer Society, Washington, DC, USA, pp. 293–300.

Kakumanu, P., Makrogiannis, S. & Bourbakis, N. (2007), 'A survey of skin-color modeling and detection methods', *Pattern Recogn.* **40**(3), 1106–1122.

Laptev, I. & Lindeberg, T. (2001), Tracking of multi-state hand models using particle filtering and a hierarchy of multi-scale image features, *in* 'Proceedings of the Third International Conference on Scale-Space and Morphology in Computer Vision', Scale-Space '01, Springer-Verlag, London, UK, pp. 63–74.

Latulipe, C., Mann, S., Kaplan, C. S. & Clarke, C. L. A. (2006), symspline: symmetric two-handed spline manipulation, *in* 'Proceedings of the SIGCHI conference on Human Factors in computing systems', CHI '06, ACM, New York, NY, USA, pp. 349–358.

Lee, M. & Woo, W. (2003), ARKB: 3d vision-based augmented reality keyboard, *in* 'ICAT'.

Liu, R. (2010), A framework for webcam-based hand rehabilitation exercises, BSc Honours Dissertation, Graphics Group, Department of Computer Science, University of Auckland, New Zealand. `http://www.cs.auckland.ac.nz/~burkhard/Reports/2010_S1_RuiLiu.pdf`.

Liu, R., Wünsche, B. C., Lutteroth, C. & Delmas, P. (2011), A framework for webcam-based hand rehabilitation exercises, *in* 'Proceedings of VISAPP 2011', pp. 626 – 631. `http://www.cs.auckland.ac.nz/~burkhard/Publications/VISAPP2011_LiuEtAl.pdf`.

Mahmoudi, F. & Parviz, M. (2006), 'Visual hand tracking algorithms', *Geometric Modeling and Imaging–New Trends* **0**, 228–232.

Malik, S. & Laszlo, J. (2004), Visual touchpad: a two-handed gestural input device, *in* 'Proceedings of the 6th international conference on Multimodal interfaces', ICMI '04, ACM, New York, NY, USA, pp. 289–296.

Microsoft Corporation (2002), 'Pointer ballistics for Windows XP'. `http://msdn.microsoft.com/en-us/windows/hardware/gg463319.aspx`.

Olsen, D. J., Pitman, N. D., Basakand, S. & Wünsche, B. C. (2011), Sketch-based building modelling, *in* 'Proceedings of GRAPP 2011', pp. 119–124. `http://www.cs.auckland.ac.nz/~burkhard/Publications/GRAPP2011_OlsenEtAl.pdf`.

OpenCV (2011), 'homepage'. `http://opencv.willowgarage.com/wiki`.

Park, J. & Yoon, Y.-L. (2006), 'LED-glove based interactions in multi-modal displays for teleconferencing', *International Conference on Artificial Reality and Telexistence* pp. 395–399.

Schauwecker, K., van den Hurk, S., Yuen, W. & Wünsche, B. C. (2011), Sketched interaction metaphors for character animation, *in* 'Proceedings of GRAPP 2011', pp. 247–252. `http://www.cs.auckland.ac.nz/~burkhard/Publications/GRAPP2011_SchauweckerEtAl.pdf`.

Segen, J. & Kumar, S. (1999), 'Shadow gestures: 3d hand pose estimation using a single camera', *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on* **1**, 1479.

SIMTICS Ltd. (2011), 'SIMTICS Ltd. homepage'. `http://www.simtics.com/`.

Song, P., Winkler, S., Gilani, S. O. & Zhou, Z. (2007), Vision-based projected tabletop interface for finger interactions, *in* 'Proceedings of the 2007 IEEE international conference on Human-computer interaction', HCI'07, Springer-Verlag, Berlin, Heidelberg, pp. 49–58.

Stenger, B., Mendona, P. R. S. & Cipolla, R. (2001), 'Model-based 3d tracking of an articulated hand', *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on* **2**, 310.

Stenger, B., Thayananthan, A., Torr, P. H. S. & Cipolla, R. (2006), 'Model-based hand tracking using a hierarchical bayesian filter', *IEEE Trans. Pattern Anal. Mach. Intell.* **28**(9), 1372–1384.

Terajima, K., Komuro, T. & Ishikawa, M. (2009), Fast finger tracking system for in-air typing interface, *in* 'Proceedings of the 27th international conference extended abstracts on Human factors in computing systems', CHI EA '09, ACM, New York, NY, USA, pp. 3739–3744.

Vassili, V. V., Sazonov, V. & Andreeva, A. (2003), A survey on pixel-based skin color detection techniques, *in* 'Proc. Graphicon', pp. 85–92.

Wang, R. Y. & Popović, J. (2009), Real-time hand-tracking with a color glove, *in* 'SIGGRAPH '09: ACM SIGGRAPH 2009 papers', ACM, New York, NY, USA, pp. 1–8.

Wang, X., Zhang, X. & Dai, G. (2007), Tracking of deformable human hand in real time as continuous input for gesture-based interaction, *in* 'Proceedings of the 12th international conference on Intelligent user interfaces', IUI '07, ACM, New York, NY, USA, pp. 235–242.

Yang, R. & Wünsche, B. C. (2010), LifeSketch - A Framework for Sketch-Based Modelling and Animation of 3D Objects, *in* 'Proceedings of the Australasian User Interface Conference (AUIC 2010)', pp. 1–10. `http://www.cs.auckland.ac.nz/~burkhard/Publications/AUIC2010_YangWuensche.pdf`.