Integrating Touch and Near Touch Interactions for Information Visualizations

Aras Balali Moghaddam
University of Victoria
BC, Canada
arasbm@uvic.ca

Jeremy Svendsen
University of Victoria
jeremys@ece.uvic.ca

Melanie Tory
University of Victoria
mtory@cs.uvic.ca

Alexandra Branzan Albu
University of Victoria
aalbu@uvic.ca

Abstract
This paper proposes a novel interaction paradigm for multi-touch interfaces, that integrates both touch and near-touch interactions. The paper describes the hardware prototype that we have built, as well as the computer vision approach that we propose for real-time hand tracking and differentiation between near-touch and touch events. We also present a case study showing how near-touch and touch interactions can be successfully integrated in an information visualization application.

Keywords
near touch interaction, multitouch, information visualization, computer vision

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation: User interfaces Input Devices and Strategies
**Introduction**

In recent years, multi-touch has gained the attention of many HCI researchers. In a multi-touch interface any contact that is made between the user’s hand, usually fingertips, and the display surface is registered as a direct input. Typically, all touch points are treated equally, regardless of the state of the user’s hand above the surface. Our present work aims to extend multi-touch interaction to distinguish between interactions on the surface versus those near the surface (i.e. touch versus near-touch).

Detecting near touch actions enables several new ways of interacting with software on touch screens. For example, it allows a form of interaction similar to a “mouse-over” event, which can be used to invoke a tooltip or hover query. Hover queries are of particular interest in visualization tools as a lightweight way of exploring item details without actually selecting the items. In addition, combining touch and near touch interaction enables a wider range of possible recognizable gestures than either alone. A study by Epps et. al. [3] demonstrated that some gestures conceived by users, such as grab and release, are impossible to capture with conventional touch screens because part of the action occurs above the surface. Similarly, Marquardt et. al.[7] argued that an interactive surface and the space immediately above it should be seen as a continuum, and that gestures on the surfaces should flow seamlessly into gestures above the surface. They demonstrated many new types of interactions that are possible with such a continuous interaction space, such as tapping an object to select it and then lifting the hand to move the object to a far away location.

We present a new approach to detecting near surface interaction and distinguishing it from touch. Our method uses computer vision to track the fingers and detect whether or not they are touching the surface. We have also developed a novel hardware setup to improve the performance of our tracking algorithms. We present these methods and demonstrate their application to an information visualization tool in the following sections. In our ongoing work we plan to extend our computer vision system to detect interesting hand postures and use these as input to our application. The long term objective of this project is to improve multi-touch interaction with information visualization tools.

*Figure 1: A snapshot of our prototype in use.*

**Related Work**

In this section we review some of the existing work that has studied hand interaction near the surface of a multi-touch display. We focus particularly on those papers that have emphasized distinguishing touch from near surface interaction. Current approaches in capturing user input near the surface can be divided into rear tracking and front tracking, depending on the position of the camera. Tracking users’ hands from above the surface has the advantage of having an undistorted view of the hands (i.e. they are not seen through a diffuser). However there are several disadvantages; these include a greater chance of occlusion, especially with multiple users, requiring per user calibration to take into account thickness of users’ fingers, and the camera may intrude on the users’ workspace. Front surface tracking can be combined with a
separate channel to track multi-touch input, however, this makes the overall system more complex.

Takeoka et al.\[9\] introduced a very accurate technique for detecting hand posture near the surface, using multiple planes of infra-red light stacked above the surface and a camera mounted behind the screen. In comparison to this approach, our method is less accurate in locating the distance of the hand from the surface; however, it is less complex and easier to reproduce. Our prototype also requires fewer computational resources since substantially fewer frames per second need to be processed to achieve a responsive system.

In work similar to ours, Hilliges et al.\[6\] explored interaction using hand gestures in the space above a rear projection surface. Our technique is different since we employ a high resolution LCD screen. Using an LCD screen has many advantages over a rear projection screen\[8\] but it imposes new challenges for capturing images of the user’s hand from behind the screen, and therefore their technique can not be used in combination with an LCD screen. We have introduced several adjustments to the hardware setup to reduce effects of the LCD on the view of the camera. Furthermore, we aim at using the near touch area to enhance interactions with information visualizations.

Another approach to tracking users’ hands near the surface involves shadow tracking on multi-touch tables\[2\]. In comparison to shadow tracking our technique is more scalable and portable since it does not require front surface illumination.

### Hardware Prototype

The multi-touch hardware that are currently available in the market, such as resistive and capacitive touch screens, do not capture any information about the state of the hand above the surface and are not suitable for our study. Han\[4\] introduce a low cost technique for constructing scalable multi-touch interfaces. A number of other computer vision based DIY techniques for building multi-touch interfaces have emerged in the last few years. The ultimate goal in the construction of these interfaces is to optimize tracking of finger touches on the display surface. The infra-red illumination and processing of the video from the camera is typically done in a way that would filter away most of a user’s hand from the picture, making these techniques unsuitable for our purpose.

In our prototype, the camera receives a 2D image of the full...
hand above the surface, as opposed to blobs corresponding to fingertips. This image is then processed by our computer vision software to detect and locate gestures of interest.

**Touch versus Near Touch**

When the user performs a gesture near the surface (e.g., hovering their index finger above a visualization) there is no passive haptic feedback to provide a reference point or an accurate sense of the distance of their finger from the screen. Based on preliminary observations, we believe that distance of the hand to the surface does not need to be measured as accurately as the corresponding position of the fingers on the 2D display surface. It is however very important to accurately distinguish between near touch and touch state. Because of the immediate haptic feedback, failure to accurately distinguish between these two states could result in user interactions that feel unresponsive or ambiguous.

**Computer Vision Method**

Using as real-time input data the video stream from the camera, we can successfully locate the fingertip and determine if it is touching the screen or hovering over it. The images that we work with are 640x480 gray-level images encoded on 8 bits. Our method involves several steps, which are detailed below. The events detected by the computer vision process (touch and near touch interaction) are passed to the visualization prototype via TUIO messages.

**Preprocessing**

The image acquisition process involves a significant amount of background noise. There are two sources of noise in the image: random background noise which has an average intensity level lower than the intensity of the hand and the direct observation of the LEDs which results in bright regions around the edge of the image.

The random background noise can be removed by smoothing and thresholding the image. The smoothing is a simple averaging with a 3x3 box filter. The thresholding sets all values below a predefined threshold to zero and maintains the brightness of all pixels above the threshold. Our implementation uses a threshold $T_{\text{noise}} = 20$.

The bright peripheral regions are removed by restraining the region of interest to the central part of the display. The central region of interest is of size 440x280. This is done by cropping the edges of the display.

![Figure 3: This demonstrates the results of the block matching algorithm. a) The thresholded image of the hand. b) The image with arrows indicating the hand’s motion.](image)

**Locating the Hand**

The hand location is detected in real-time by using motion information. In a first step, all pixels in motion within the image are detected. Motion detection is implemented using an optical flow approach [1] based on block matching with
blocks of size 16x16. The second and final step detects the center of the hand as the center of the largest blob in motion.

**Detecting the Fingertip**

Once a central point on the hand has been detected, the next step is to detect the fingertip. This is done through region growing (pixel aggregation around the central pixel). After the region is grown, the topmost pixel of the region is detected.

![Figure 4: Sharpness of the fingertip boundaries where: a) the finger is touching the screen, b) the finger is hovering above the screen.](image)

**Detecting Touch**

In order to distinguish between hover and touch, the sharpness of the fingertip boundary is analyzed. The further the hand gets from the screen, the more blurred it becomes. In contrast, when the finger is touching the screen, the region corresponding to it exhibits a very sharp edge (see fig 4). Since we already know the location of the fingertip, the algorithm looks for gradients of high magnitude just below the fingertip. Gradients are computed using first order differences. Examples of a finger touching the screen and a finger hovering can be seen in figure 4.

![Figure 5: The visualization application. a) Background. The user can press and hold one finger anywhere on this background to start a file selection menu and load one or more data files. b) Border. This border groups visualizations related to one data file. By using two fingers on this border, the user can perform zooming and panning. c) Touch. Touching a bar in the histogram will select the data items represented by the bar. In this case the user has selected USD in the currency histogram. This selection highlights all the records that use USD in other linked charts (i.e. brushing). d) Hover. In another histogram, the user notices that all records which use USD as currency belong to the same customer segment. The user hovers over the bar to quickly retrieve more information about this customer segment without changing their previous selection. A pop-up is displayed containing "UBS Network" as the name of the customer segment, as well as the number of records that belong to this segment.](image)
Case Study
In order to experiment with near touch interaction, we have developed a preliminary information visualization prototype. We selected this application area because it could benefit from both touch and near touch interactions and because it fit well with our research interests in designing visualizations for touch screens. The software is written in the Python programming language, using PyMT as the primary development library. Figure 5 shows the application in use. The application can process several types of tabular data and create default visualizations to describe that data. For instance, a histogram is generated for each dimension. A near touch interaction invokes a hover query, revealing detailed information for a given bar. By contrast, a touch interaction invokes selection, and subsequent brushing of related bars in other charts.

Conclusion and Future Work
We have demonstrated a novel method for hand tracking both on and above a touch surface, and for distinguishing touch and near touch interactions. Our prototype visualization tool provides a simple example of how these touch interactions can be utilized within an application. The system is currently limited to single touch and hover interaction as a proof of concept. However, we see no obstacles to extending it with multi-touch interaction in near future. In our ongoing work, we are improving the capabilities of this tool, attempting to detect various hand shapes, and exploring how such gestures may improve touch interaction with visualization interfaces.

References