Extended Multitouch: Recovering Touch Posture, Handedness, and User Identity using a Depth Camera

Sundar Murugappan[a], **Vinayak**[a], **Niklas Elmqvist**[b], **Karthik Ramani**[a]
[a]School of Mechanical Engineering
[b]School of Electric Engineering
Purdue University, West Lafayette, IN 47907
sundar.murugappan@gmail.com

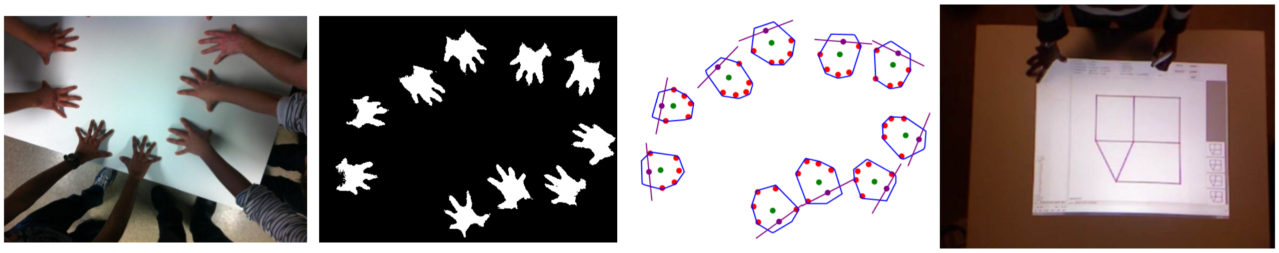


Figure 1. Extracting finger and hand posture from a Kinect depth camera (left) and integrating with a pen+touch sketch interface (right).

ABSTRACT
Multitouch surfaces are becoming prevalent, but most existing technologies are only capable of detecting the user's actual points of contact on the surface and not the identity, posture, and handedness of the user. In this paper, we define the concept of *extended multitouch interaction* as a richer input modality that includes all of this information. We further present a practical solution to achieve this on tabletop displays based on mounting a single commodity depth camera above a horizontal surface. This will enable us to not only detect when the surface is being touched, but also recover the user's exact finger and hand posture, as well as distinguish between different users and their handedness. We validate our approach using two user studies, and deploy the technology in a scratchpad application as well as in a sketching tool that integrates pen and touch for creating beautified sketches.

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation: User Interfaces—*Interaction styles*; I.3.6 Computer Graphics: Methodology and Techniques—*Interaction techniques*

General Terms
Design, Algorithms, Human Factors

Author Keywords
Multitouch, tabletop displays, depth cameras, pen + touch, User studies.

INTRODUCTION
Multitouch surfaces are quickly becoming ubiquitous: from wristwatch-sized music players and pocket-sized smartphones to tablets, digital tabletops, and wall-sized displays, virtually every surface in our everyday surroundings may soon come to life with digital imagery and natural touch input. However, achieving this vision of ubiquitous [28] surface computing requires overcoming several technological hurdles, key among them being how to augment any physical surface with touch sensing. Conventional multitouch technology relies on either capacitive sensors embedded in the surface itself, or rear-mounted cameras capable of detecting the touch points of the user's hand on the surface [9]. Both approaches depend on heavily instrumenting the surface, which is not feasible for widespread deployment.

In this context, Wilson's idea [31] of utilizing a front-mounted depth camera to sense touch input on any physical surface is particularly promising for the following reasons:

- **Minimal instrumentation:** Because they are front-mounted, depth cameras integrate well with both projected imagery as well as conventional screens;
- **Surface characteristics:** Any surface, digital or inanimate, flat or non-flat, horizontal or vertical, can be instrumented to support touch interaction;
- **On and above:** Interaction both on as well as above the surface can be detected and utilized for input;

- **Low cost:** Depth cameras have recently become commodity products with the rise of the Microsoft Kinect^{TM}, originally developed for full-body interaction with the Xbox game console, but usable with any computer.

In this paper, we define a richer touch interaction modality that we call *extended multitouch* (EMT) which includes not only sensing multiple points of touch on and above a surface using a depth camera, but also (1) recovering finger, wrist and hand posture of the user, as well as (2) detecting user identity and (3) user handedness. We have implemented a MS Windows utility that interfaces with a Kinect camera, extracts the above data from the depth image in real-time, and publishes the information as a global hook to other applications, or using the TUIO [15] protocol. To validate our approach, we have also conducted two empirical user studies: one for measuring the tracking accuracy of the touch sensing functionality, and the other for measuring the finger and hand posture accuracy of our utility.

The ability to not only turn any physical surface into a touch-sensitive one, but also to recover full finger and hand posture of any interaction performed on the surface, opens up an entirely new design space of interaction modalities. We have implemented two separate touch applications for the purpose of exploring this design space. The first is a prototype scratchpad where multiple users can draw simple vector graphics using their fingers, each finger on each user being separately identified and drawing a unique color and stroke. The second application is called 'uSketch', and is an advanced sketching application for creating beautified mechanical engineering sketches that can be used for further finite element analysis. 'uSketch' integrates pen-based input that we use for content creation with our touch sensing functionality that we use for navigation and mode changes.

RELATED WORK

Our work is concerned with touch interaction, natural user interfaces, and the integration of touch with other input modalities. Below we review relevant work in these areas.

Multitouch Technologies

Several different technologies exist for sensing multiple touches on a surface, including capacitance, RFID, and vision-based approaches. Vision-based approaches are particularly interested for our purposes since they are suitable for surfaces with large form factor, such as tabletops and walls [16, 21, 30, 19, 9, 22]. These systems rely on the presence of a dedicated surface for interaction.

Recently, depth-sensing cameras are beginning to be used in various interactive applications where any surface can be made interactive. Wilson [31] explored the use of depth sensing cameras to sense touch on any surface. OmniTouch [10] is a wearable depth-sensing and projection system that enables interactive multitouch applications on everyday surfaces. Lightspace [32] is a small room installation that explores interactions between on, above and between surfaces combining multiple depth cameras and projectors.

Sensing Touch Properties

Going beyond merely sensing multiple touch points and deriving *properties* of the touches requires more advanced approaches. We broadly classify existing approaches for this into into vision-based techniques, instrumented glove-based tracking, and specialized hardware.

Vision-based Techniques

Malik et al. [19] used two cameras mounted above the surface to distinguish which finger of which hand touched a surface, but their system requires a black background for reliable recognition. Wang et al. [26] detect the directed finger orientation vector from contact information in real time by considering the dynamics of the finger landing process. Dang et al. [4] mapped fingers to their associated hand by making use of constraints applied to the touch position with finger orientation. Freeman et al. [6] captured the user's hand image on a Microsoft Surface to distinguish hand postures through vision techniques. Finally, Holz et al. [13] proposed a generalized perceived input point model in which fingerprints were extracted and recognized to provide accurate touch information and user identification.

Glove-based Techniques

Gloves are widely popular in virtual and augmented reality environments [24]. Wang et al. [27] used a single camera to track a hand wearing an ordinary cloth glove imprinted with a custom pattern. Marquardt et al. [20] used a fiduciary-tagged gloves on interactive surfaces to gather input about many parts of a hand and to discriminate between one person's or multiple peoples' hands. Although simple and robust, these approaches require users to wear additional instrumented gear which is not always desirable compared to using bare hands.

Specialized Hardware

Specialized hardware is capable of deriving additional touch properties beyond the mere touch points. Benko et al. [1] sensed muscle activity through forearm electromyography to provide additional information about hand and finger movements away from the surface. Lopes et al. [17] integrated touch with sound to expand the expressiveness of touch interfaces by supporting recognition of different body parts, such as fingers and knuckles. TapSense [11] is a similar acoustic sensing system that allows conventional surfaces to identify the type of object being used for input such as fingertip, pad and nail. The above systems require additional instrumentation of either the user or the surface and take a significant amount of time for setup. The presence of background noise is also a deterrent to use acoustic sensing techniques.

Pen and Touch-based Interaction

A few research efforts have explored the combination of pen and touch [34, 7, 2, 35]. Hinckley et al [12] described techniques for direct pen + touch input for a prototype, centered on note-taking and scrapbooking of materials. Lopes et al [18] assessed the suitability of combining pen and multi touch interactions for 3D sketching. Through experimental evidence, they demonstrated that using bimanual interactions simplifies work flows and lowers task times, when compared to the pen-only interface.

OVERVIEW: EXTENDED MULTITOUCH INTERACTION

Traditional multitouch interaction [5] is generally defined as the capability to detect multiple points of touch input on a surface. Here, we enrich this concept into what we call *extended multitouch interaction* (EMT), defined as follows:

- Detecting multiple touch points on and above a surface;
- Recovering finger, wrist and hand postures as well as the handedness (i.e., whether the user is using the left or right hand); and
- Identifying and distinguishing between different users interacting with the surface.

While previous work has explored various combinations of the above, we are aware of no single work that has addressed all three constraints together as a unified framework. To this extent, we propose a two-dimensional hand model that uniquely maps the touch points to the fingers, hands and the users from just the depth data. In the following text, we describe our approach to supporting extended multitouch using low-cost depth cameras.

TOUCH SENSING USING A DEPTH CAMERA

Inspired by Wilson [31], our approach to sensing touch using a depth camera simply builds on analyzing the depth image to detect when objects (i.e., fingers) come in close proximity with the physical surface. Our implementation uses a depth camera mounted above a horizontal surface to transform user gestures on and above the tabletop to multitouch events that can be used by a client application. Fully emulating existing multitouch technology using this setup gives rise to several concrete requirements for our system:

- Determine when the surface is being touched and generate the corresponding touch points;
- Track touch points over time when the user is dragging fingers across the surface;
- Map touch points from the physical surface to coordinate space of the display area; and
- Export multitouch events to clients, either using a global Windows hook, or using the TUIO protocol [15].

Below we discuss the physical setup and each of the above components in more detail.

Physical Setup

Our low-cost setup (Figure 2) consists of (1) a horizontal surface, on and above which interactions take place, (2) a projector, which projects on to a surface providing visualization and feedback, (3) a depth sensing camera (like Microsoft Kinect) for capturing touch input, and (4) a computer for processing the input data from the Kinect and projecting visual output using the projector. Again, the surface to be made interactive need not be empty nor flat.

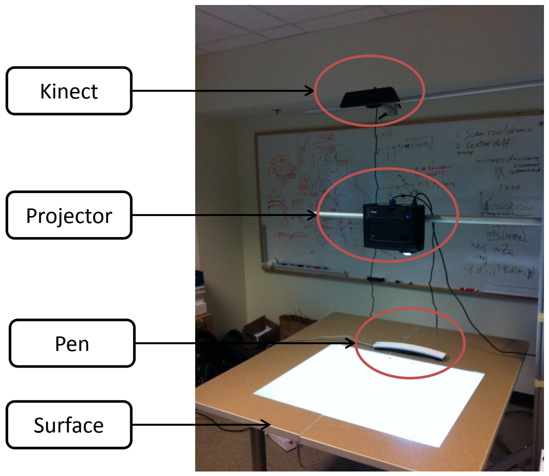


Figure 2. The physical setup showing (from top to bottom) depth camera, projector, pen tracker, and physical surface.

While our focus in this paper is on horizontal (tabletop) setups, there is nothing to prevent the same technique from being used for vertical (wall-mounted) surfaces. The algorithms described below are independent of the configuration and the touch results are not affected as long as a clear line of sight is maintained between the hands and the Kinect.

Basic Touch Sensing

We detect touches by analyzing the image produced by the depth camera. The intensity of every pixel in the depth image corresponds to the distance from the camera. This allows us to define a "touch input region" as a virtual volume located within a lower and upper distance above the surface. This volume is approximately a cuboid (assuming the surface is flat) offset from the surface with the height equal to the difference of the bounds. All pixels with distance values between these bounds correspond to the contact region(s) of the finger(s) with the surface.

Estimating the underlying surface is important for the purpose of obtaining reliable and robust touches using our method. We model the surface as a depth image that can simply be subtracted from the current image to yield the difference. This difference map is then used to detect whether any physical objects in the scene fall within the touch input region. We create a binary image such that the pixels corresponding to these elements are set to white and all the other pixels to black. Figure 3(a) shows the raw depth image with two hands touching the surface and Figure 3(b) is the binary image showing groups of white pixels corresponding to contacts of fingers with the surface.

We use connected component analysis to identify the cluster of pixels from the touch images, yielding touch "blobs." To differentiate between actual touches and noise, we discard the blobs that are below or above a certain area threshold. From experimentation, when the Kinect is at a height of 1.2m from the surface, for a blob to be a candidate for a touch point, the area of the blob must be greater than 20

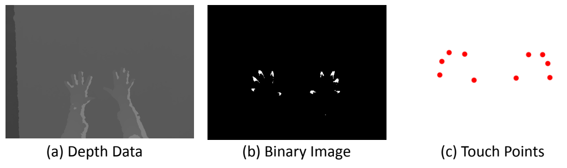


Figure 3. Identifying touch points from the depth data.

square millimeter and less than 220 square millimeter, accommodating a wide variety of finger sizes. We use the center of gravity of the white blobs as the location of touch points. Figure 3(c) shows the result of determining the touch points from the raw depth data and the binary touch image.

Note that this approach also lends itself to defining additional volumes at other distances from the surface, allowing us to detect interaction above the surface as well.

Tracking Touch Points

Tracking touch points over time is critical for developing interactive applications that make use of dragging gestures. Without this information, each new depth frame would treat touch points as being new, and the application would not be able to support smooth dragging operations.

In our implementation, we associate touch points in consecutive depth frames by finding the nearest neighbor between points in the frames. Since the Kinect can record at a maximum rate of thirty frames per second, not all depth frames are necessarily used in tracking touches. Our system intelligently discards outlier depth frames by considering the touch points in a sliding window that includes several frames before the current frame, thereby increasing robustness.

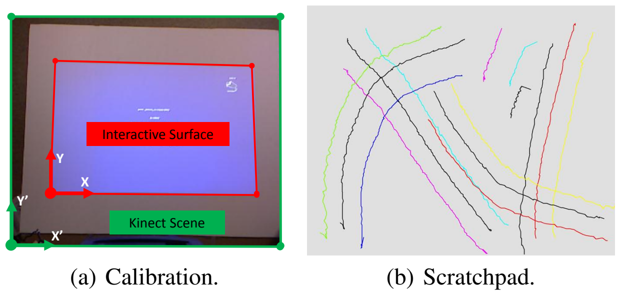


Figure 4. Touch sensing mechanism.

Mapping Touches to Display Space

Traditional multitouch surfaces have a one-to-one mapping between the physical location that is being touched on the surface and the digital location on the underlying display. This provides users with the feeling of directly touching and manipulating the digital content underneath their fingers.

To facilitate a similar interactive experience in our depth camera-based approach, we need to map the depth camera scene to the projected display area (Figure 4(a)). We employ a technique very similar to how regular touch screen sensors are calibrated with LCD displays using a nine-point calibration grid. This gives us the coefficients needed to transform from camera coordinates to display coordinates while taking scaling, rotation, and translation errors into account.

Generating Multitouch Events

The final step in our implementation is to actually generate multitouch events that a client application can handle in order to support (extended) multitouch interaction. We provide two ways to do this: either through a Windows global hook, or using the TUIO protocol [15]. This enables our implementation to be entirely standalone, and external client applications simply need to interface to the system to receive touch events. Figure 4(b) shows a scratchpad application that demonstrates multitouch tracking using these events.

BEYOND STANDARD MULTITOUCH INTERACTION

Our work so far has concerned itself with detecting the two-dimensional (*x* and *y*) locations of touch points on as well as above (i.e., with a *z* component) the physical surface. However, fully supporting our extended multitouch (EMT) model requires being able to recover the finger and hand posture, handedness, and user identity of all touch points. More powerful and rich interaction techniques can be built if such additional information can be leveraged.

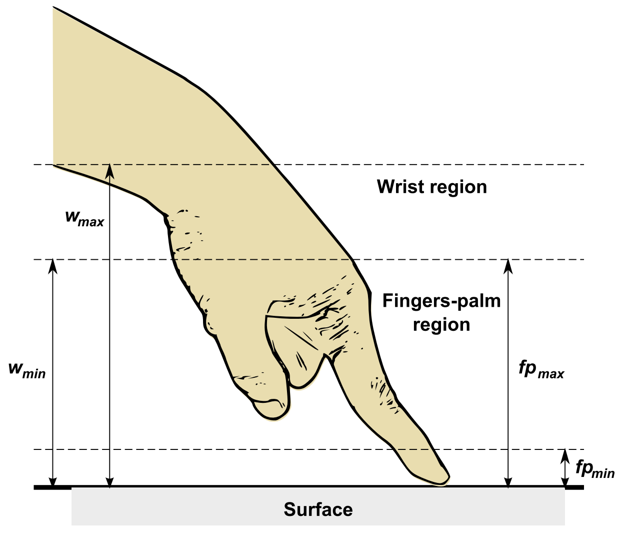


Figure 5. Additional touch regions for extended multitouch.

Extracting Palm and Wrist Data

Similar to the touch input region defined earlier for detecting fingers coming into contact with the physical surface, we can extract the position of the user's palm and wrists from the same depth data. We simply include two additional virtual volumes: the "fingers-palm region" and the "wrist region" (Figure 5). Based on the depth camera image, we construct binary images for each of these regions (Figure 6). These images are then used to recover the touch posture.

A connected component analysis is then performed on the binary image corresponding to 'fingers-palm region' to extract the blobs. For each blob, we compute its a) center of

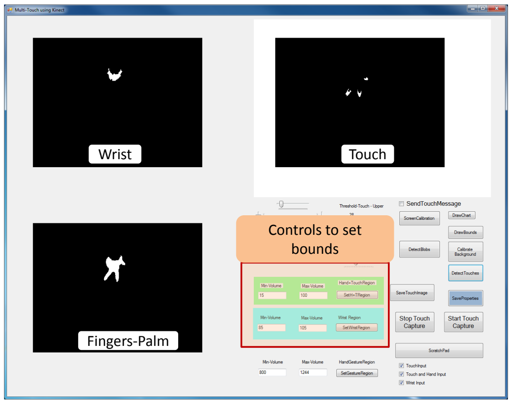


Figure 6. Binary depth images for the three virtual volumes used in our prototype implementation.

gravity, b) principal directions using principal component analysis and c) convex hull. The center of the palm is approximately the center of gravity of the blob, which is sufficient to map fingers and hands to touch points.

The binary image corresponding to the wrist region is processed in the same manner as above. For each extracted blob, we compute its a) center of gravity and b) principal directions. Wrist orientation can be determined from the principal directions. Tabletop surfaces can take advantage of this information to eliminate the inherent problems that arise due to orientation dependency. Furthermore, this position and orientation information provides an estimate of the hand occlusion region, which can be used to build occlusion-aware interfaces that minimize the impact of occlusion [14, 25].

A Two-Dimensional Hand Model

After processing the three binary images, we combine the results and create a two-dimensional model of the hand. Each hand now has a wrist, fingers-palm, the touch points, and their associations. It can also be noted that the distance value of touches, wrist-center, and palm-center can be obtained from the depth data, which, when added to the two-dimensional model, gives us a three-dimensional model.

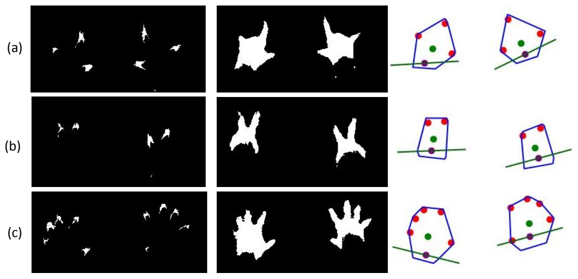


Figure 7. From left to right, each image shows the binary image computed in the touch region, fingers-palm region, and the hand model with touch points, wrist and palm.

Figure 7 shows results of touches from two hands on the table. The red dots on the far right images correspond to the touch points, the green dot is the palm center, and the violet dot is the wrist. The blue closed polygon is the convex hull of the fingers-palm blob and the green line represents the first principal component of the wrist showing orientation of the hand. The principal components of the fingers-palm are not shown to avoid clutter. Figure 1 shows 50 touch points from 10 hands and 5 users.

Modeling Touch Posture

To deduce the mapping between the touch points to the fingers and hand, we take advantage of the different geometric properties that can be computed from the abstract and simplistic hand model derived above. In Figure 8, the purple dot is the center position of the wrist and the purple line represents its first principal component. Similarly, the green dot and green line represent the center position and the first principal component of the fingers-palm region. The yellow arrow (V3) represents the direction of the fingers from the wrist passing through the palm. For every instance of the 2D model, we compute (a) the angles subtended by each touch point with respect to V3, and (b) the distance of each touch point from the wrist.

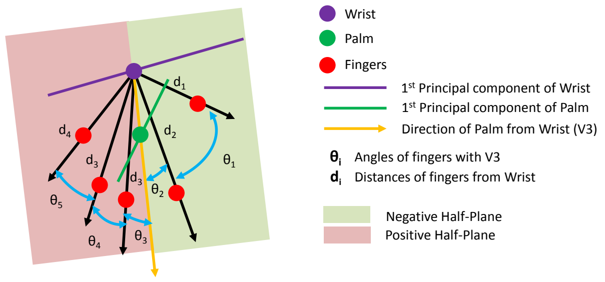

Wrist
Palm
Fingers
1st Principal component of Wrist
1st Principal component of Palm
Direction of Palm from Wrist (V3)
θ_i Angles of fingers with V3
d_i Distances of fingers from Wrist
Negative Half-Plane
Positive Half-Plane
d_1
d_2
d_3
d_4
θ_1
θ_2
θ_3
θ_4
θ_5

Figure 8. The angles and distances of the hand geometry.

The order of the touch points can be deduced from the computed signed angles. We arrange the touch points on a circle based on their subtended angles with V3. For example, when the input is five touch points from the hand, we can see that the thumb is at the either end with both the angle and distance between thumb and index fingers greater than between little and ring fingers. The other fingers can now be labeled logically.

Detecting handedness, i.e., if it is a right or left hand, can be determined from the location of the thumb, i.e., its signed angle with respect to the orientation vector. If the signed angle is negative, then the input pattern is a right hand and vice versa.

Wobbrock et al. [33] outlined 27 different gestural commands performed on a tabletop surface. From studying these hand poses, we collected the following observations:

- The majority of gestures are performed with one, two, three, or five finger touch points;
- One-finger interactions are always performed using the fore-finger (100%);
- For two-finger interactions, the fingers used are thumb-fore finger and fore-middle fingers;

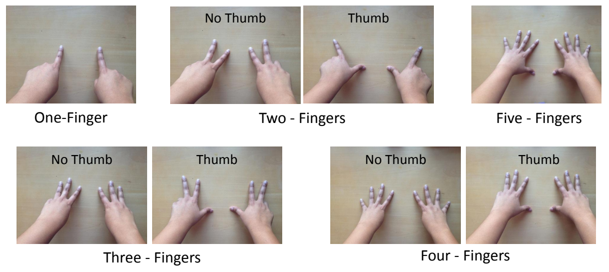


Figure 9. Hand poses during multitouch recognized in our system.

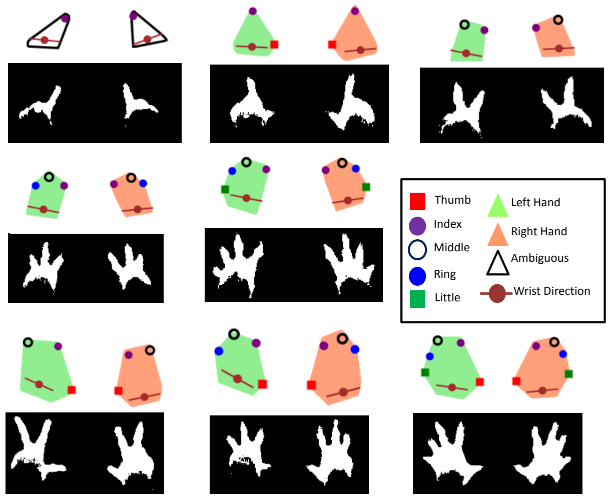


Figure 10. Results (actual snapshots) after mapping touch points to fingers and hands for the poses in figure 9

- For three-finger interactions, the finger combinations used are thumb-fore-middle and fore-middle-ring; and
- For four-finger interactions, the finger combinations are thumb-fore-middle-ring and fore-middle-ring-and-little.

Based on these observations, we propose 16 different touch postures (Figure 9) to be recognized by our system. Figure 10 shows a sample of the results generated by our system for these 16 different postures. Poses that do not belong to the above classification (for example, Figure 7a) are ambiguous and hence our system cannot identify the fingers reliably. Furthermore, the presence of the thumb and fore finger is crucial in our method for the correct mapping between fingers and touch points, and the thumb is crucial for deducing handedness. Hence, the accuracy of the mapping reduces for hand poses which do not have the thumb. In such case, the distance and angular measures alone do not give conclusive evidence of the correct touch posture and handedness.

Detecting User Identity

Finally, being able to match touch inputs to individual users is crucial for more advanced collaborative applications where multiple participants are interacting with the tabletop. Our solution builds on the fact that user arms are also partially visible to the depth camera mounted above the tabletop. We use this information to heuristically model estimated user positions that are off-screen of the depth image. However, this approach depends on users maintaining their relative positions around the tabletop surface, as well as not approaching to closely to each other.

A future improvement to this solution would integrate a second vision-based tracking system that could follow the participants themselves instead of merely their touch points.

USER STUDIES

To validate our approach, we conducted two empirical user studies: one for measuring the tracking accuracy of the touch sensing functionality, and the other for measuring the finger and hand posture accuracy of our utility.

Touch Sensing Accuracy

The accuracy of touch contacts depends on (a) the parameters computed in calibrating the interactive area (projected imagery) to the Kinect scene, (b) the algorithm to detect touch contacts based on depth thresholds, (c) the orientation and size of the finger(s) making the contact, and (d) the user's perception of where they are touching the surface. The above factors cumulatively contribute towards the accuracy of the detected touches and as some of them might not be controlled, these factors model the real-world performance of our system.

Participants

We recruited 14 volunteers (11 males and 3 females) between the ages of 21 and 32 years for the experiment. Among them, 12 were right-handed and 2 were left-handed.

Task

The task was to touch 15 randomly generated locations that were projected on the surface. Each location was visually displayed as a circle with a cross-hair of size 20×20 mm. Each participant was asked to touch every point and hold for two seconds using their index finger. For every location, the system acquired the calibrated touch points repeatedly, between a finger-down and a finger-up event. The system showed only one location at a time, and on every finger-up event, a new point was displayed until the end of the experiment.

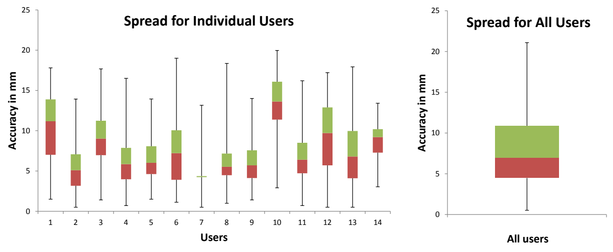


Figure 11. Touch sensing accuracy with average spread for individual users and all users collectively.

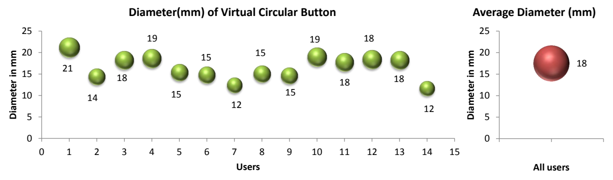


Figure 12. Accuracy of hitting a circular virtual button.

Result

In all, 3715 data points were collected from 14 participants. We analyzed the targeting performance using two independent measures: mean offset and spread [10, 13]. Figure 11 shows the average spread for the data collected. The average spread in the best case is 4mm and for the worst case is 13mm. The average spread, taking into account all data points collectively from all users, is 8mm, which is better than previous findings in the literature [10]. The mean offset for each user is shown in Figure 13(a) with User-10 having the farthest (largest) offset and User-2 being the nearest (small offset) to the target.

Figure 13(b) shows the distribution of user-clicks for all 14 users with compensating offsets. Here, the origin represents the target locations and the red markers represent the acquired touch points. There were no significant difference in results between the handedness of the users. We also calculated the accuracy of hitting a circular virtual button by accommodating 95% of the touches for each user with compensating offsets. Figure 12 shows the minimum button diameter for every user, with lowest (diameter = 12mm) for User-7 and highest (diameter = 21mm) for User-1, the average diameter being 18mm. In comparison, the size of a finger pad is between 10 - 14mm; the fingertip is 8 - 10 mm [3]; the recommended button size for Microsoft Surface is 12mm and for conventional touch input is 15mm [13]. We posit

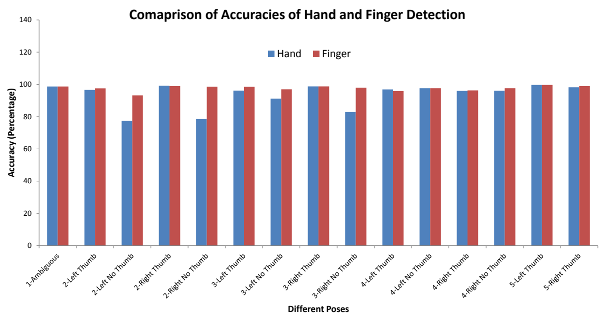


Figure 14. Touch Posture and Handedness Accuracy

that the accuracy of the touches will decrease with increase in height of the Kinect from the surface. Future studies have to be done with regards to this aspect.

Touch Posture and Handedness Accuracy

We conducted a second user study to evaluate the accuracy of identifying the touch posture and handedness from the depth data.

Participants

We recruited 9 volunteers, 7 males and 2 females, between the ages of 22 and 34 years for this experiment.

Task

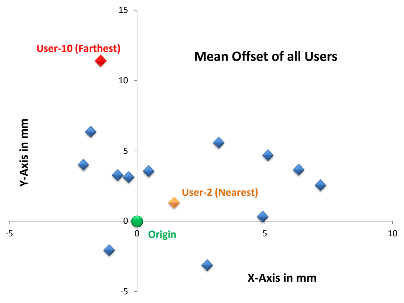
Each participant was asked to perform 8 touch-poses (Figure 9) on the surface with both their left and right hands one after the other. Each pose was to be held for approximately 10 seconds. The participants were not restricted to any position or orientation around the table. No visual feedback was provided to the participant during the study.

Result

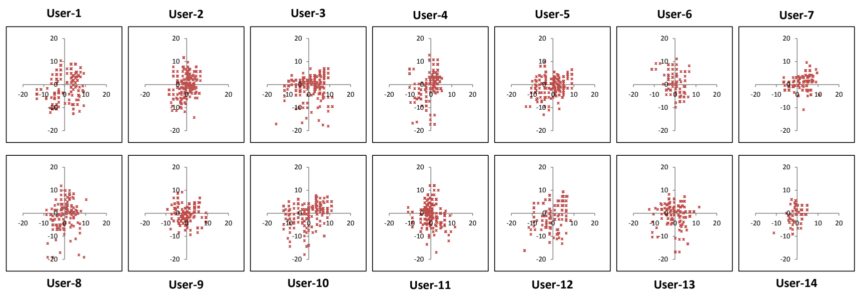
A total of 4971 data points were collected during the experiment. Among them, the system computed a hand posture for only 4918 data points (99%). The possible reason for this is that the users moved their hands either too fast or the hands were not in line of sight to the Kinect.

Figure 14 shows the identification accuracies of touch-posture and handedness for the 16 (8 poses *times* 2 hands each) different touch poses. The overall handedness identification accuracy is 93.2%. We discarded one-finger index pose, as our current implementation cannot distinguish the hands for this particular pose. While the two-finger and three-finger “no thumb” poses had an accuracy of 77% and 86% respectively, their corresponding “thumb” poses had an accuracy of 98% and 97% respectively. Thus, the presence of the thumb is critical in identifying the handedness for two-finger and three-finger poses using our algorithm. Our hand model can be improved by reasoning with the contour of the posture and also convexity defects to better identify the posture. However, this will involve more computation and it is important to consider its trade-offs in real-time applications.

The overall finger identification accuracy was 97.65%. At the end of the user study, each participant was asked to rate



(a) Mean offsets from target (all users).



(b) Distribution of user clicks with compensating mean offsets (mm).

Figure 13. Distribution diagrams.

their least favorable touch-poses. Among all, seven participants voted the four-finger poses ("thumb", "no-thumb", left and right hands) and 5 participants voted the "three-finger-no-thumb" pose as their least favorites.

DESIGN IMPLICATIONS

Surfaces supporting extended multitouch (EMT) enable a richer and more diverse interaction design space than those merely supporting standard multitouch. Here we briefly review some of these designs in more detail.

Interaction Above and Beyond the Surface. Existing research has already explored interaction on and above a surface, but we think there is more work to be done here. Touch screens traditionally do not support hovering, but this is certainly one of the straightforward ideas to explore. For example, hovering over a surface could bring up a tool palette, a context menu, or simply a tooltip, similar to how hovering is handled in current mouse-based interfaces. More complex modalities are certainly possible, including detecting volumetric and 3D gestures, as suggested by Wigdor et al. [29].

Identifying Users. The ability to match touches to users opens the door for personalized touch interaction. For example, instead of relying on social protocols [23], an EMT application can keep track of object ownership and explicitly prevent users lacking permission from interacting with the object. In a command and control setting, this could be utilized to assign different roles to different participants, such as a fire chief having permission to give orders to fire trucks, whereas a police chief instead may only control squad cars.

Recognizing Posture. Our implementation has proved the validity of recovering finger and hand posture using a very simple and low-cost hardware setup. This in turn brings several of the posture-based ideas devised by other researchers into the realm of possibility: examples include context-sensitive touch menus, complex gestures on and above the surface, and providing explicit gestural feedback during training.

EXAMPLE: PENS WITH EXTENDED MULTITOUCH

Though natural, touch input is not suited for tasks such as writing and sketching. These inking tasks are particularly suited for pen-based input which have high precision, whereas our multitouch accuracy is approximately 8mm. Multitouch input in general on all forms and modalities suffer from low precision and our system is no exception. Using a pen for input, on the other hand, it is a familiar tool that exploits the users' experience of being been trained to use a pen right from their early days.

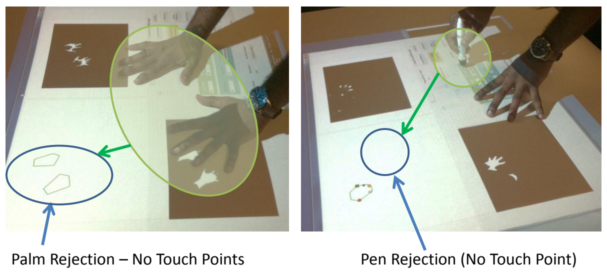

Palm Rejection – No Touch Points
Pen Rejection (No Touch Point)

Figure 15. Palm (left) and pen (right) rejection during touch capture.

However, a digital pen allows for only one input point for interaction, constraining the input bandwidth. Augmenting pen-based input with extended multitouch capabilities increases the users' input bandwidth, which can be leveraged by designers to create rich and actually natural interactions for a variety of applications. We outline a division of labor between pen and touch input based on the theme "the pen writes and touch manipulates" [12]. We present 'uSketch', an advanced sketching application for creating beautified mechanical engineering sketches that can be used for further finite element analysis.

Combining Pen and Touch Input

For enabling bimanual pen and touch interactions, we added the Mimio^{TM} pen to the setup described in Figure 2. Mimio pen technology tracks the 2D position of a digital stylus using ultrasound and infrared, and wirelessly transmits the location to the computer when the stylus is pressed against any flat surface. The pen can be calibrated to any surface size and works both horizontally and vertically. The pen input can then be processed and visually displayed back on the surface using the projector. This process allows the users to write and sketch on any surface, thus making it interactive.

Any object that touches the surface is identified as a touch point by the Kinect sensor. Hence, the system must be able to negate detecting the contact point of the pen as a touch input (Figure 15). To achieve this, we take advantage of how

Kinect senses touch. When the pen is in contact with the surface, it communicates its coordinates to the computer. We take advantage of this coordinate information to negate any touch points detected by the system within a small radius of the pen contact. Similarly, when the whole hand rests against the surface, the system detects a big white blob as the palm rests completely on the surface (Figure 15). The system discards the input as it does not correspond to a touch point. Hence, both the pen and palm are conveniently discarded by the design of how the system processes the depth data from Kinect.

‘uSketch’

Figure 1 shows a snapshot of the ‘uSketch’ application on the interactive surface taken from above the table, and Figure 16 shows the ‘uSketch’ interface with (a) a freehand sketch of a 2D-bracket, (b) the beautified sketch with interpreted symbols, and (c) the deformation results in ANSYS^{TM}.

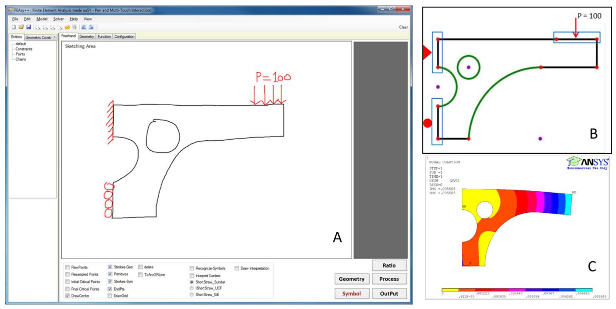


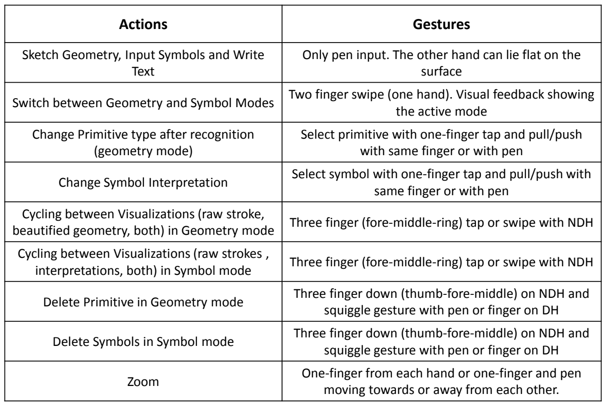
Figure 16. The ‘uSketch’ interface showing (a) a freehand sketch of a 2D-bracket, (b) the beautified sketch with interpreted symbols, and (c) the deformation results in ANSYS^{TM}.

Guiard [8], proposed general principles for asymmetric bimanual actions comprising of three rules: (a) the dominant hand (DH) moves within the frame of reference defined by the non-dominant hand (NDH); (b) the sequence of motion generally sees the NDH setting the reference frame prior to actions with the DH being made within that context; and (c) that the DH works at a higher level of precision than the NDH in both spatial and temporal terms. Hence, the pen is preferably held in the DH and both NDH and DH be used for touch-input with the pen tucked in the DH. Here, we outline the bimanual pen and touch interactions that we built into ‘uSketch’ (see figure 17).

Using bimanual interactions with pen and touch eliminates the need for constant switching between modes and also the need for user interface elements for changing visualizations and other interactions. We posit that using such interactions simplifies work flows and lowers task times, when compared to the pen-only or touch-only interface.

CONCLUSIONS AND FUTURE WORK

We have presented a multitouch sensing approach based on low-cost depth cameras where any physical surface, flat or non-flat, horizontal or vertical, projected or screen, can be instrumented with touch sensing. Our contribution goes beyond existing depth camera touch sensing approaches into



Actions	Gestures
Sketch Geometry, Input Symbols and Write Text	Only pen input. The other hand can lie flat on the surface
Switch between Geometry and Symbol Modes	Two finger swipe (one hand). Visual feedback showing the active mode
Change Primitive type after recognition (geometry mode)	Select primitive with one-finger tap and pull/push with same finger or with pen
Change Symbol Interpretation	Select symbol with one-finger tap and pull/push with same finger or with pen
Cycling between Visualizations (raw stroke, beautified geometry, both) in Geometry mode	Three finger (fore-middle-ring) tap or swipe with NDH
Cycling between Visualizations (raw strokes , interpretations, both) in Symbol mode	Three finger (fore-middle-ring) tap or swipe with NDH
Delete Primitive in Geometry mode	Three finger down (thumb-fore-middle) on NDH and squiggle gesture with pen or finger on DH
Delete Symbols in Symbol mode	Three finger down (thumb-fore-middle) on NDH and squiggle gesture with pen or finger on DH
Zoom	One-finger from each hand or one-finger and pen moving towards or away from each other.

Figure 17. Bimanual pen and touch interactions for ‘uSketch’.

what we call *extended multitouch*, which we define as the ability to not only sense multiple touches, but also detecting interaction above the touch surface, recovering the user’s finger and hand posture, and distinguishing between the users interacting with the surface. We validate the work using two user studies gauging the accuracy of touch and posture recognition, as well as using two sketch applications where we begin to explore the design space of this novel idea.

We have only scratched the surface of this extended multitouch interaction approach, and, as discussed in the design implications above, we can see many opportunities for future work in this area. In particular, we envision continuing our investigations into combining multiple interaction modalities for early engineering design and sketching.

REFERENCES

1. Benko, H., and Wilson, A. DepthTouch: Using depth-sensing camera to enable freehand interactions on and above the interactive surface. In *Proceedings of the IEEE Workshop on Tabletops and Interactive Surfaces* (2009), vol. 8.

2. Brandl, P., Forlines, C., Wigdor, D., Haller, M., and Shen, C. Combining and measuring the benefits of bimanual pen and direct-touch interaction on horizontal interfaces. In *Proceedings of the ACM Conference on Advanced Visual Interfaces* (2008), pp. 154–161.

3. Dandekar, K., Raju, B. I., and Srinivasan, M. A. 3-d finite-element models of human and monkey fingertips to investigate the mechanics of tactile sense. *Journal of Biomechanical Engineering 125*, 5 (2003), 682–691.

4. Dang, C. T., Straub, M., and André, E. Hand distinction for multi-touch tabletop interaction. In *Proceedings of the ACM Conference on Interactive Tabletops and Surfaces* (2009), pp. 101–108.

5. Dietz, P., and Leigh, D. DiamondTouch: a multi-user touch technology. In *Proceedings of the ACM Symposium on User interface Software and Technology* (2001), pp. 219–226.

6. Freeman, D., Benko, H., Morris, M. R., and Wigdor, D. ShadowGuides: visualizations for in-situ learning of multi-touch and whole-hand gestures. In *Proceedings of the ACM Conference on Interactive Tabletops and Surfaces* (2009), pp. 165–172.

7. Frisch, M., Heydekorn, J., and Dachselt, R. Investigating multi-touch and pen gestures for diagram editing on interactive surfaces. In *Proceedings of the ACM Conference on Interactive Tabletops and Surfaces* (2009), pp. 149–156.

8. Guiard, Y. Asymmetric division of labor in human skilled bimanual action: The kinematic chain as a model. *Journal of Motor Behavior 19*, 4 (1987), 486–517.

9. Han, J. Y. Low-cost multi-touch sensing through frustrated total internal reflection. In *Proceedings of the ACM Symposium on User Interface Software and Technology* (2005), pp. 115–118.

10. Harrison, C., Benko, H., and Wilson, A. D. OmniTouch: wearable multitouch interaction everywhere. In *Proceedings of the ACM Symposium on User Interface Software and Technology* (2011), pp. 441–450.

11. Harrison, C., Schwarz, J., and Hudson, S. E. TapSense: enhancing finger interaction on touch surfaces. In *Proceedings of the ACM Symposium on User Interface Software and Technology* (2011), pp. 627–636.

12. Hinckley, K., Yatani, K., Pahud, M., Coddington, N., Rodenhouse, J., Wilson, A., Benko, H., and Buxton, B. Manual deskterity: an exploration of simultaneous pen + touch direct input. In *Extended Abstracts of the ACM Conference on Human Factors in Ccomputing Systems* (2010), pp. 2793–2802.

13. Holz, C., and Baudisch, P. The generalized perceived input point model and how to double touch accuracy by extracting fingerprints. In *Proceedings of ACM Conference on Human Factors in Computing Systems* (2010), pp. 581–590.

14. Javed, W., Kim, K., Ghani, S., and Elmqvist, N. Evaluating physical/virtual occlusion management techniques for horizontal displays. In *Human-Computer Interaction - Proceedings of INTERACT* (2011), vol. 6948 of *Lecture Notes in Computer Science*, Springer, pp. 391–408.

15. Kaltenbrunner, M. reacTIVision and TUIO: A tangible tabletop toolkit. In *Proceedings of the ACM Conference on Interactive Tabletops and Surfaces* (2009), pp. 9–16.

16. Krueger, M. W., Gionfriddo, T., and Hinrichsen, K. VIDEOPLACE: an artificial reality. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (1985), pp. 35–40.

17. Lopes, P., Jota, R., and Jorge, J. A. Augmenting touch interaction through acoustic sensing. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces* (2011), pp. 53–56.

18. Lopes, P., Mendes, D., Araújo, B., and Jorge, J. A. Combining bimanual manipulation and pen-based input for 3d modelling. In *Proceedings of the Eurographics Symposium on Sketch-Based Interfaces and Modeling* (2011), pp. 15–22.

19. Malik, S., and Laszlo, J. Visual touchpad: a two-handed gestural input device. In *Proceedings of the ACM International Conference on Multimodal Interfaces* (2004), pp. 289–296.

20. Marquardt, N., Kiemer, J., Ledo, D., Boring, S., and Greenberg, S. Designing user-, hand-, and handpart-aware tabletop interactions with the TouchID toolkit. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces* (2011), pp. 21–30.

21. Matsushita, N., and Rekimoto, J. HoloWall: designing a finger, hand, body, and object sensitive wall. In *Proceedings of the ACM Symposium on User Interface Software and Technology* (1997), pp. 209–210.

22. Microsoft Corporation. Microsoft surface, 2012. `http://www.microsoft.com/surface/` (accessed March 2012).

23. Morris, M. R., Ryall, K., Shen, C., Forlines, C., and Vernier, F. Beyond 'social protocols': multi-user coordination policies for co-located groupware. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work* (2004), pp. 262–265.

24. Sturman, D. J., and Zeltzer, D. A survey of glove-based input. *IEEE Computer Graphics and Applications 14*, 1 (Jan. 1994), 30–39.

25. Vogel, D., and Balakrishnan, R. Occlusion-aware interfaces. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (2010), pp. 263–272.

26. Wang, F., Cao, X., Ren, X., and Iran, P. Detecting and leveraging finger orientation for interaction with direct-touch surfaces. In *Proceedings of the ACM Symposium on User Interface Software and Technology* (2009), pp. 23–32.

27. Wang, R. Y., and Popović, J. Real-time hand-tracking with a color glove. *ACM Transactions on Graphics 28*, 3 (2009).

28. Weiser, M. The computer for the twenty-first century. *Scientific American 3*, 265 (Sept. 1991), 94–104.

29. Wigdor, D., Benko, H., Pella, J., Lombardo, J., and Williams, S. Rock & rails: extending multi-touch interactions with shape gestures to enable precise spatial manipulations. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (2011), pp. 1581–1590.

30. Wilson, A. D. TouchLight: an imaging touch screen and display for gesture-based interaction. In *Proceedings of the ACM International Conference on Multimodal interfaces* (2004), pp. 69–76.

31. WILSON, A. D. Using a depth camera as a touch sensor. In *Proceedings of the ACM Conference on Interactive Tabletops and Surfaces* (2010), pp. 69–72.

32. WILSON, A. D., AND BENKO, H. Combining multiple depth cameras and projectors for interactions on, above and between surfaces. In *Proceedings of the ACM Symposium on User Interface Software and Technology* (2010), pp. 273–282.

33. WOBBROCK, J. O., MORRIS, M. R., AND WILSON, A. D. User-defined gestures for surface computing. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (2009), pp. 1083–1092.

34. YEE, K.-P. Two-handed interaction on a tablet display. In *Extended Abstracts of the ACM Conference on Human Factors in Computing Systems* (2004), pp. 1493–1496.

35. ZELEZNIK, R., BRAGDON, A., ADEPUTRA, F., AND KO, H.-S. Hands-on math: a page-based multi-touch and pen desktop for technical work and problem solving. In *Proceedings of the ACM Symposium on User Interface Software and Technology* (2010), pp. 17–26.