DIRECT: Making Touch Tracking on Ordinary Surfaces Practical with Hybrid Depth-Infrared Sensing

Robert Xiao **Scott Hudson** **Chris Harrison**
Carnegie Mellon University, Human-Computer Interaction Institute
5000 Forbes Avenue, Pittsburgh, PA 15213
{brx, scott.hudson, chris.harrison}@cs.cmu.edu

ABSTRACT
Several generations of inexpensive depth cameras have opened the possibility for new kinds of interaction on everyday surfaces. A number of research systems have demonstrated that depth cameras, combined with projectors for output, can turn nearly any reasonably flat surface into a touch-sensitive display. However, even with the latest generation of depth cameras, it has been difficult to obtain sufficient sensing fidelity across a table-sized surface to get much beyond a proof-of-concept demonstration. In this paper we present DIRECT, a novel touch-tracking algorithm that merges depth and infrared imagery captured by a commodity sensor. This yields significantly better touch tracking than from depth data alone, as well as any prior system. Further extending prior work, DIRECT supports arbitrary user orientation and requires no prior calibration or background capture. We describe the implementation of our system and quantify its accuracy through a comparison study of previously published, depth-based touch-tracking algorithms. Results show that our technique boosts touch detection accuracy by 15% and reduces positional error by 55% compared to the next best-performing technique.

Author Keywords
Touch tracking; depth sensing; sensor fusion.

ACM Classification Keywords
H.5.2. Information interfaces and presentation (e.g. HCI): User interfaces: Input devices and strategies.

INTRODUCTION
Touch interfaces have become ubiquitous for small screens due to the popularity of touchscreen-based smartphones and tablets. However, for much larger displays, touchscreens remain expensive and can be intrusive to install in some environments. On the other hand, walls, tables, and other relatively flat surfaces are already present in many spaces. The introduction of digital projectors and low-cost depth camera technologies raises the possibility of transforming these everyday surfaces into large, touch-sensitive computing experiences.

While free-space hand and finger tracking research spans several decades of research, starting with seminal work by Krueger in the Video Place System [10], comparatively little research has examined finger and touch tracking on ordinary, unmodified surfaces. This can be attributed to the difficult challenge of first segmenting a finger from the background to extract its spatial position and then undertaking the even more challenging task of sensing when a finger has physically contacted a surface (vs. merely hovering close to it).

The advent of inexpensive depth cameras offered a promising potential solution for addressing this challenge. Early work by Wilson et al. [24] demonstrated the potential of this approach for detecting touches on arbitrary surfaces.

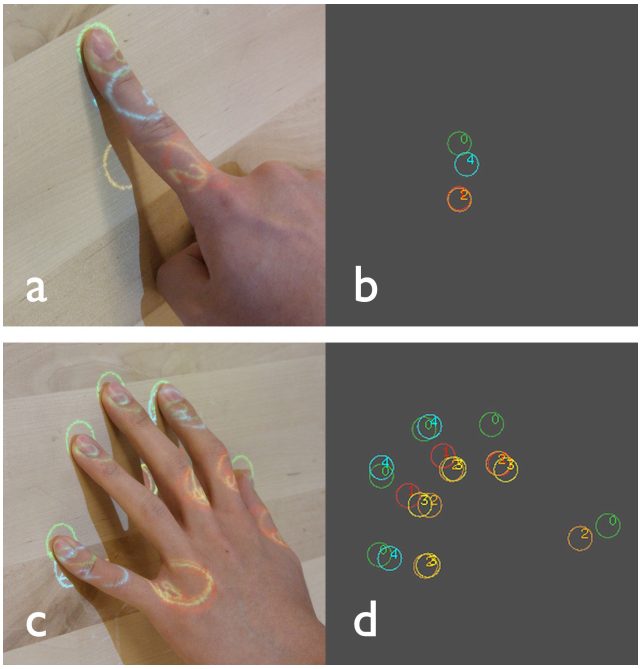


Figure 1. Comparison of depth-camera-based touch tracking methods. Our method (DIRECT) is shown in green and labeled 0. Comparison methods are single-frame model (1/red), maximum distance model [24] (2/orange), statistical model [7, 25] (3/yellow) and slice finding [6] (4/cyan).

ISS '16, November 06-09, 2016, Niagara Falls, ON, Canada
© 2016 ACM. ISBN 978-1-4503-4248-3/16/11···$15.00
DOI: http://dx.doi.org/10.1145/2992154.2992173

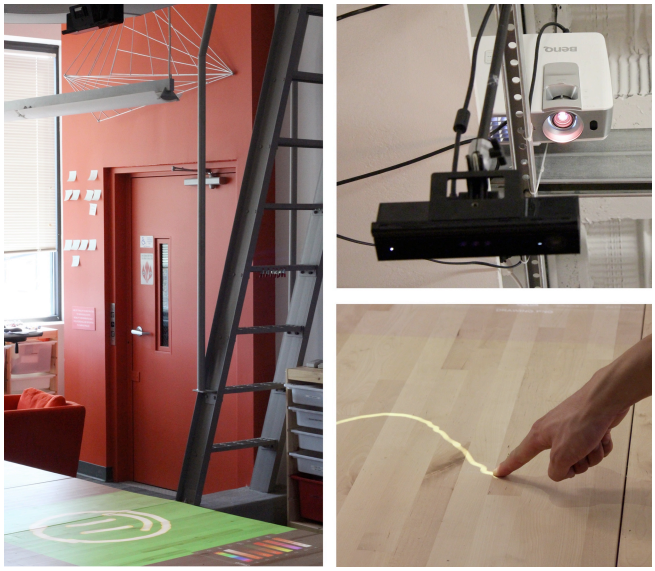


Figure 2. The system setup. Left: the Kinect depth camera is mounted 1.6 m from the table surface. Top right: the projector and Kinect are rigidly mounted and calibrated to each other (but not the surface). Bottom right: The table surface functions as a touchscreen.

While prior approaches have demonstrated that depth-based touch tracking should be viable, full exploration of the design space requires input sensing, which exhibits high stability and positional accuracy, as well as reliable touch segmentation/detection with both low false positives and low false negatives.

Unfortunately, attaining this very high accuracy strains the capabilities of even the latest generation of depth cameras. The depth resolution and noise characteristics of current generation sensors means that fingertips simply merge into the surface at distances needed to cover reasonably sized work surfaces (Figure 3a), making precise touch tracking extremely difficult. This has made it challenging to move beyond the first proof-of-concept research systems to practical use in real deployments.

CONTRIBUTIONS

In this paper, we describe DIRECT (Depth and IR Enhanced Contact Tracking), a new touch-tracking approach that merges depth and infrared image data (from a single sensor) to provide significantly enhanced finger tracking (Figures 1 and 2). Infrared imagery provides precise finger boundaries, while depth imagery provides precise contact detection. Additionally, the use of infrared data allows the system to more robustly reject tracking errors arising from noisy depth data. This approach allows DIRECT to provide touch tracking precision to within a single pixel in the depth image, and overall, finger tracking accuracy approaching that of conventional touch-screens. Additionally, our approach makes no assumptions about user position or orientation (in contrast to all prior systems, which require *a priori* knowledge of finger orientation to correct for undetected fingertips), nor does it require prior calibration or background capture.

We describe in detail the technical implementation of DIRECT and discuss its capabilities and unique characteristics relative to other touch tracking approaches. We further contribute a multi-technique comparison study – the first evaluation of its type – where we compare four previously published methods against one another, as well as against DIRECT. As we will show, DIRECT readily outperforms these prior techniques with respect to viable distance, touch precision, touch stability, multi-finger segmentation, and touch detection false positives and false negatives.

Additionally, to encourage research in this domain, as well as facilitate replication and comparative studies, we have made implementations of DIRECT and our four comparative methods available for download and review at https://github.com/nneonneo/direct-handtracking.

RELATED WORK

There are many different approaches for touch tracking on large surfaces. The simplest approach is to create a special purpose surface, using *e.g.* cameras [5, 13] or capacitive sensors [11]. Alternatively, existing surfaces can be retrofitted with sensors, such as acoustic sensors to detect the sound of a tap [16], or infrared emitters and receivers to detect occlusion from a finger. There are also methods that can operate on ad hoc, uninstrumented surfaces. These systems most often use optical sensors (*e.g.*, cameras [9] or LIDAR [15]).

Detecting whether a finger has contacted a surface is challenging with conventional RGB or infrared cameras, which has inspired several approaches. PlayAnywhere [22] demonstrated a touch tracking approach based on analyzing the shadows cast by a finger near the surface. Sugita et al. [19] detect touches by tracking the visual change in the fingernail when it is pressed against a surface. TouchLight [23] uses a stereo pair of cameras, detecting touches when the finger images pass beyond a virtual plane. Many other systems use finger dwell time or an external sensor (accelerometer [8], microphone, or acoustic sensor [16, 26]) to detect touch events.

Most related to our present technique are depth camera-based touch-tracking systems. Depth cameras sense the physical distance from the sensor to each point in the field of view, making it possible (in concept) to innately sense whether a finger has contacted a surface or not. Broadly, these systems can be placed into two categories:

Background modeling approaches compute and store a model or snapshot of the depth background. Touches are detected where the live depth data differs from the background depth map in specific ways. Wilson [24] uses a background snapshot computed from the maximum depth point observed at each pixel over a small window of time. KinectFusion [7] uses a background model developed by analyzing the 3D structure of the scene from multiple angles (SLAM), effectively producing a statistically derived background map. WorldKit [25] also uses a statistical approach, computing the mean and standard deviation for

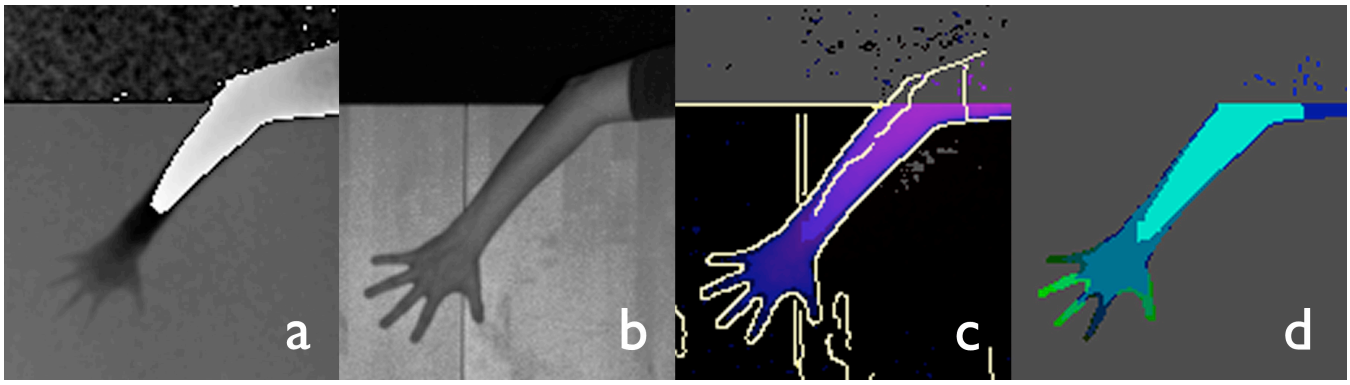


Figure 3. Close-up of touch tracking process for five fingers laid flat on the table. (a) depth image, (b) infrared image, (c) infrared edges overlaid on z-score map, (d) segmentation result. In (d), *arm* pixels are cyan, *hand* pixels are blue-green, and *finger* pixels are combined with *fingertip* pixels and shown in various shades of green.

each pixel, modeling both the background and noise. Finally, MirageTable [1] captures a background mesh and renders foreground hands as particles.

Finger modeling approaches attempt to segment fingers based on their physical characteristics, and generally do not require background data. OmniTouch [6] used a template-finding approach to label finger-like cylindrical slices within depth images. The slices are then merged into fingers, and finally touch contacts. FlexPad [18] detected and removed hands by analyzing the subsurface scattering of the Kinect's structured infrared light pattern, allowing the background to be uniquely segmented.

Surprisingly few systems attempt to fuse depth sensing with other sensing modalities for touch tracking. Of the existing literature on sensor fusion depth-sensing systems, only the Dante Vision project [17] uses a multisensory approach for touch tracking, combining depth sensing with thermal imaging infrared camera. This is used to improve touch contact detection accuracy (a thermal imprint is left on the surface upon physical touch), though at the expense of continuous tracking and high contact latency (~200 ms).

IMPLEMENTATION

We implemented our touch tracking algorithm and comparison techniques in C++ on a 2.66 GHz 3-core Windows PC, with a Kinect for Windows 2 providing the depth and infrared imagery. The Kinect 2 is a time-of-flight depth camera, which uses active infrared illumination to determine the distances to objects in the scene. It provides 512x424 pixel depth and infrared images at 30 frames per second. A BenQ W1070 projector with a resolution of 1920x1080 is also mounted above our test surface (a wooden table) to provide visual feedback.

The Kinect 2 is mounted 1.60 meters above a large table surface, and the projector is 2.35 meters above the surface (Figure 2). At the horizontal edges of the Kinect's field of view, the table surface is 2.0 meters from the Kinect. Both the projector and Kinect are securely mounted to the ceiling, and were calibrated to each other using multiple views of a planar calibration target.

The present configuration allows the projector to project a 1.0×2.0 meter image onto the table surface, with the Kinect capable of sensing objects across the entire projected area. At this distance, each projected pixel is 1.0 mm square, and each Kinect depth pixel is 4.4 mm square at the table surface. Thus, even with this second generation sensor, a typical fingertip resting on the table is less than 3 depth image pixels wide, underscoring the sensing challenge.

Our approach combines background modeling and anthropometric modeling approaches. More specifically, DIRECT models both the background *and* the user's arms, hands, and fingers using separate processes. Our processing pipeline is carefully optimized so that it runs at camera frame rate (30 FPS) using a single core of the PC.

The touch-tracking pipeline is illustrated in Figures 3 and 4. Figure 3 shows a hand laid flat on the table, which is challenging for depth-based touch tracking approaches because the fingertips generally fuse with the background due to sensor imprecision and noise (Figure 3a). In Figure 3d, we show that DIRECT can still segment the fingers (labeled in different colors) right down to the fingertip. Figure 4 shows another challenging case - a single extended finger raised 60º. This is problematic because there are very few depth pixels available for the fingertip itself. As before, DIRECT is able to segment the crucial fingertip.

Background Modeling

Our system uses a statistical model of the background, inspired in part by the implementation in WorldKit [25]. At every pixel, we maintain a rolling window of five seconds of depth data and compute the mean and standard deviation. This model allows us to establish both a highly accurate mean depth background, as well as a noise profile at every pixel in the scene.

With these rolling windows, we support *dynamic* updating of the background model. If the standard deviation exceeds a certain depth-dependent threshold (accounting for higher average noise further from the sensor), the pixel is temporarily "stunned" and its background model mean and standard deviation will be held constant until the moving average

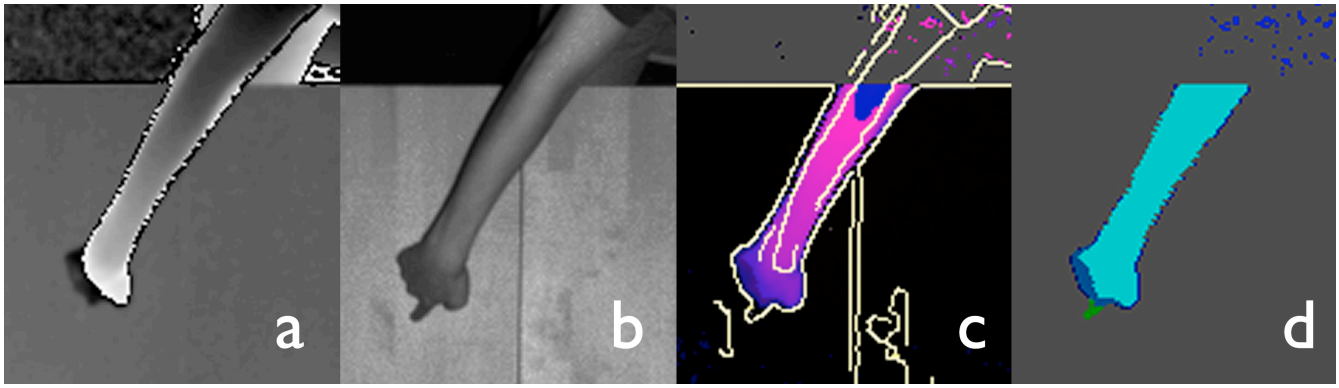


Figure 4. Close-up of touch tracking process for a finger angled at 60º vertically. (a) depth, (b) infrared, (c) edges and depth map z-scores, and (d) filled blobs. Refer to Figure 3 for a full color key.

drops below the threshold. This approach accurately tracks long-term changes in the environment (*e.g.*, objects moved around), while ignoring short-term changes (*e.g.*, actively-interacting hands and fingers). In our present implementation, the background is updated in a separate thread, running at 15 fps to avoid excessively frequent updates. This type of dynamic background updating is crucial for long-running systems to deal with shifts in the environment (*e.g.*, movement of objects residing on the surface), yet most existing systems (*e.g.*, [24, 25]) use only a single static background model captured during initial setup. We note that highly stationary hands and fingers could be "integrated into the background" with this approach, though we observed that, in practice, users rarely stay stationary for several seconds over top of active touch interfaces.

Infrared Edge Detection

We use the infrared image primarily to detect the boundary between the fingertip and the surrounding surface. As such, our first step is to detect edges in the infrared image. The Kinect 2's infrared image comes from the same sensor as the depth data, and thus it is precisely registered to the depth image. We use a Canny edge filter [2] to locate candidate edge pixels in the image (7x7 Sobel filter, with hysteresis thresholds of 4000 and 8000), with results shown in Figure 5. Note the parameters are not specific to the operating depth or objects in the scene.

After running the Canny edge filter, some edges may have gaps. These can occur due to *e.g.,* multiple edges meeting. A common gap-filling technique, image dilation followed by erosion, is inappropriate for our case due to the small size of the fingertip. Instead, we employ an *edge-linking* [12] algorithm across the edge map, which walks along Canny edge boundaries and bridges one-pixel breaks between neighboring edges. After applying this algorithm, fingertips and hands are usually fully enclosed (Figures 3c and 4c, pale yellow lines).

Figure 5. Canny edge detection.
Left: Kinect IR image. Right: Edge map.

Surfaces with a very similar infrared albedo to skin could cause issues for edge finding. However, anecdotally, we have found that the shadows cast by the arm and hand (illuminated by the active IR emitter found in many depth cameras), even near the fingertip, helps to increase contrast (as seen in Figures 3b and 4b).

Iterative Flood-Fill Segmentation

Our touch tracking pipeline consists of a sequence of iterative flood fills, each responsible for a different pixel type: arm filling, hand filling, finger filling, and fingertip filling. When each flood fill completes, it triggers the next fill in the sequence, starting from pixels on its boundary. Critically, each flooded area is *linked* to the parent area from which it was seeded (*e.g.* the finger fill starts from the hand that it is attached to), forming a hierarchy of filled objects that match their respective anthropometric requirements derived from [3, 4, 14, 21]. For a finger to be successfully segmented (and passed as input to an interface), a complete hierarchy must exist – a process that robustly rejects finger-like objects that are not connected to hands, arms and so on (*e.g.*, a whiteboard marker laying on a tabletop).

Arm Stage

The first stage is labeling arm pixels, and merging them into connected *arm blobs* (Figures 3d and 4d, light blue). Arm pixels are defined as pixels that are at least 5 cm closer to the sensor than the background mean, *i.e.* pixels that are at least 5 cm above the surface. This high threshold is chosen both to unambiguously distinguish human activity from background noise (standard deviations at the edge of the depth map can reach ~1.5 cm, so 5 cm is more than 3 standard deviations away), and to detect human forearms even when laid *totally* flat on the table (6.3 cm is the 2.5th percentile diameter of a human forearm).

Hand Stage

Then, for all arm blobs, our algorithm flood fills downwards towards the hand pixels, defined as pixels that are at least 12 mm from the surface. This threshold was chosen to segment individual fingers apart from the hand (12 mm is the 2.5th percentile thickness of a human finger, allowing us to detect even small fingers laying flat on a table). During this step, the fill is constrained to avoid pixels with high depth variance in their local neighborhood. This constraint ensures that this flood fill does not simply fill into noisy pixels surrounding the arm. The result is a *hand blob* attached to the parent arm blob (Figure 3d and 4d, dark blue blob). If multiple hand blobs are found, only the largest is taken (to avoid noise-induced "phantom hands").

Finger Stage

In the third stage, the algorithm fills from the hand blob into finger pixels, which are defined as pixels that are at least one standard deviation above from the surface mean. These pixels are above noise, but are otherwise very close to the surface. Because of this, we constrain the fill to stay within the boundaries derived from the infrared edge map. In the event of a hole in the edge map, the fill will stop at below-noise pixels and thus will not fill too far. This process produces a number of *finger blobs* attached to the hand blob, and the point at which the finger attaches to the hand is called the *finger base*.

Fingertip Stage

Finally, for each finger blob, the algorithm fills further into the below-noise pixels (fingertip pixels). At this point, the depth map is not used (as fingers generally merge with noise), and only the infrared edge constrains the fill. A vast majority of frames fill successfully, however occasionally, a gap in the edge map will cause the flood to escape outside the finger. To mitigate this, our flood fill stops and flags an *overfill error* if the fill extends more than 15 cm from the finger base. This value allows for both the longest human fingers (mean length 8.6 cm, SD 0.5) and is 2 standard deviations above the mean palm center to fingertip length (mean 12.9 cm, SD 0.8), which is the worst-case scenario if the hand fill stage was only partially successful in flooding into the hand. Additionally, this permits the use of grasped pens, markers or styluses used as pointing devices, while still rejecting out-of-control flood fills that spill out onto the background.

If an overfill condition is detected, DIRECT deletes the flooded fingertip pixels and returns a failure indication. This allows the system to gracefully fall back to depth-only touch tracking in the event that the IR image is unusable for any reason (*e.g.* because there are holes in the edge image). Crucially, this enables the algorithm to work in cluttered or complex environments when the edge image may be damaged or unusable, albeit with reduced performance. Otherwise, if no overfill occurs, the fingertip pixels are added to the parent finger blob. The resulting finger blobs are seen in Figure 3d and 4d (varying shades of green).

Touch Point Extraction

During both the finger fill and tip fill, we record the distance of each finger pixel to the finger base. For each detected finger, we simply place the fingertip at the pixel with the highest such distance. We found that this pixel correlates extremely well with the fingertip's actual location, and furthermore that this point is stable enough that touch position smoothing is unnecessary.

If the tip filling failed due to an over-fill, the fingertip's position can be estimated using forward projection, by using the arm and hand positions to determine the orientation of the finger. However, the resulting estimate will be substantially noisier, a fact which can be conveyed to a higher-level filtering or smoothing algorithm.

Touch Contact Detection

To detect if the fingertip is in contact with the surface behind it (*i.e.* to distinguish *hover* from *contact*), we examine the 5x5 neighborhood around the tip pixel. If any pixel is more than 1 cm from the background, we mark the finger as hovering, otherwise the finger is marked as touching the surface. We then apply hysteresis to avoid rapid changes in touch state. Although this touch detection approach is simplistic, it is surprisingly robust; we attribute this to the high precision of our fingertip tracking.

Touch Tracking Properties

The DIRECT approach merges aspects of optical tracking, background modeling and anthropometric finger modeling approaches, and therefore exhibits some unique properties relative to other methods. Compared to depth-only methods, the touch points detected by DIRECT are more stable, as the infrared estimation provides pixel-accurate detection of the fingertip. Consequently, DIRECT requires no temporal touch smoothing to work well, allowing it to have very low latency (15 ms average latency from input depth data to output touch point) without sacrificing accuracy.

While DIRECT uses ideas from finger modeling, it does not directly model the shape of fingers, nor does it make assumptions about the shape of the touch contact area. Therefore, DIRECT is capable of tracking touches when the fingers assume unusual configurations, such as holding all fingers together, performing gestural touches, or holding objects such as pens and styli.

Lastly, DIRECT provides some additional information about the hand and fingers beyond just the touch point. Specifically, it also provides the orientation of the finger (the vector from the finger base to fingertip), the pose of the hand, and the arm associated with each touch. This metadata could be used to implement interaction techniques beyond simple multitouch, *e.g.* using the finger angles for rotational input [20], or to manipulate 3D objects [27], and using the arm data to enable bimanual interactions.

COMPARATIVE TECHNIQUES

To compare the performance of our technique, we implemented four representative depth-camera-based touch tracking methods from the literature (see also Related Work). An

example frame of tracking output from these implementations can be seen in Figure 1 (see also Video Figure).

Of note, many of our comparison techniques were originally developed using the predecessor of the Kinect 2 we use (the "Kinect for Windows", henceforth called Kinect 1 for simplicity). The Kinect 1 sensor uses structured light, which projects an infrared speckle pattern, as opposed to the time-of-flight method used in the Kinect 2. The speckle pattern renders the infrared image virtually unusable for tracking, precluding DIRECT-style sensing. The Kinect 1 features both lower depth image pixel resolution and lower depth resolution (~4 mm per depth unit at a distance of 2 meters) than the Kinect 2. We have thus adjusted and tuned the comparison techniques to work with the Kinect 2 sensor as best possible.

We also did not use any calibration with these techniques. Touch tracking systems often employ a calibration stage where a user touches several points to establish a mapping between the sensor and known physical points. However, if the user calibrates without significantly changing their orientation, this calibration can mask certain orientation-dependent biases (as we show in our results) and make the system dependent on the user and finger orientation. Hence, in our study, we apply only a global calibration between the depth camera and the projector, and do not calibrate using detected finger positions.

All of our comparison techniques have a certain number of "magic numbers" that must be tuned for correct operation of the system. Furthermore, precise adjustments of these numbers can be used to trade off between *e.g.* touch accuracy and false touch detection. For the study, we tuned these numbers to keep false positives acceptably low, as these are especially damaging to the user experience. Specifically, we aimed to reduce false positives to less than one false-positive per five seconds within our target area (2 m^2) when no fingers are present. Although this is still a high rate of false positives, we found that attempting to further reduce this rate led to unacceptable insensitivity in two of our comparison techniques. We tuned each comparison technique independently, communicating with the system's authors when necessary to best reproduce the technique.

Tests on all techniques were done without smoothing of the touch data. Smoothing could be applied to the output of any of these approaches and might increase accuracy. However, this would come at the cost of increased latency and would tend to hide the merits of the technique itself.

Single-Frame Background Model

Our first comparison technique is a common, naïve technique often used for simple touch tracking. It uses a background model consisting of a single captured depth frame. Candidate touch pixels are simply those that lie between a minimum and maximum distance threshold from the background (in our implementation, between 7 and 15 mm from the surface).

For all background model implementations, we apply a low-pass boxcar filter followed by thresholding. A connected-components pass then extracts touch blobs, following the implementation in [24]. Although most naïve implementations do not perform this filtering, we found it necessary to avoid excessive noise in the contact tracking.

Maximum Distance Background Model

The second comparison technique models the background using the *maximum* depth value seen over a window of time. This effectively implements a conservative noise model of the background. Like the single-frame approach, the captured depth is then processed through a low-pass filter and thresholded, followed by a connected-components pass to segment finger touches.

Wilson [24] implements a variation on this technique (pers. comm.), using a histogram to choose *e.g.*, the 90th percentile depth value at each pixel to eliminate outliers in the original Kinect data. With the Kinect 2, we did not observe such outliers, and so our maximum distance model is effectively the same as the histogram method. Wilson tested their system using a Kinect at a maximum distance of 1.5 metres from a table. At that distance, Wilson reported anecdotally that the positional tracking error was about 1.5 cm.

Statistical Background Model

The final background modeling method uses a statistical approach. This implementation uses the same mean and standard deviation calculation as the DIRECT method. New depth frames are converted into Z-scores based on their differences from the background (specifically, the value of each pixel, minus the mean, divided by the standard deviation), and the Z-scores are then filtered, thresholded and connected as in the other methods.

This method closely resembles the background differencing approach used in WorldKit [25]. It also aims to capture the essence of the KinectFusion SLAM touch tracking approach [7], in that it integrates the background profile gradually over time, building a statistical model of the environment that is much more accurate than any single frame. However, our replicated approach lacks the spatial averaging of KinectFusion.

Slice Finding and Merging

For our final comparison technique, we chose to implement the slice-finding approach used in OmniTouch [6]. This approach locates cylindrical slices in the depth image using an elastic template, and then links together adjacent slices to form fingers. Of note, unlike the other approaches, OmniTouch requires that the fingers be clearly separated in the depth image. Furthermore, as it does not use a background model, it may detect erroneous touches on the surface due to objects already in the scene; therefore, for the study, we cleared the table surface of all objects.

The original OmniTouch implementation only supported fingers oriented horizontally. We therefore extended OmniTouch by implementing biaxial template matching – locating candidate finger slices in both the X- and Y-axes –

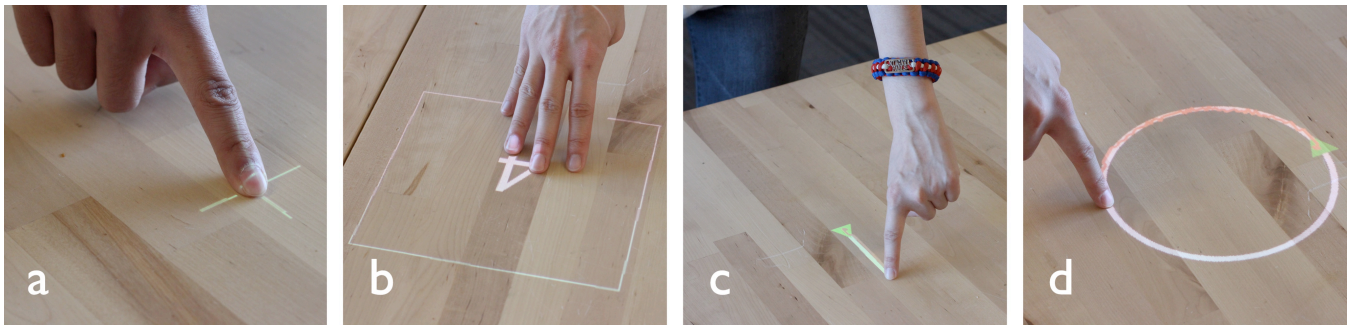


Figure 6. Tasks performed by users in our study. (a) crosshair task, (b) multitouch box task, (c) line shape tracing task, (d) circle shape tracing task.

and then merging the centroids of these slices into fingers. Our approach demonstrably locates fingers oriented in any direction. To best replicate OmniTouch's true performance, we also implemented the system's finger forward-projection method. As noted previously, because the fingers fuse with the depth background upon touch, it is difficult to estimate the true tip position. In response, OmniTouch uses the detected finger's orientation to extend the estimated touch position 15 mm towards the un-sensed tip [pers. comm.]. This improvement is not directly applicable to the previously discussed background-subtraction methods, as they do not model the finger orientation or shape. Of note, the original OmniTouch system was designed to operate at a distance of just 40 cm.

EVALUATION

To assess the accuracy of DIRECT, we ran an evaluation with 12 participants (3 female; average age 25). All users were familiar with touch interfaces. Each study session lasted for roughly 30 minutes, and participants were compensated $10 USD for their time.

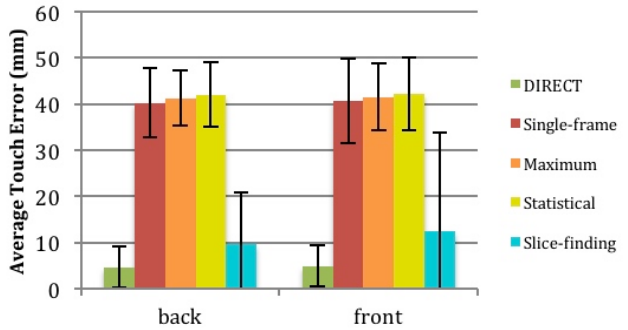
Participants were simply told that the table surface was a touchscreen from the outset, and to touch the surface as they would any ordinary touchscreen. Users were permitted to use either hand (including interchangeably) and to use any finger pose they found comfortable. Users were not required to remove jewelry or roll up sleeves – several users conducted the study while wearing long sleeved shirts, bracelets, watches and rings. Our experiment system ran all five touch-tracking methods (DIRECT plus the four comparison techniques) simultaneously at a consistent 30 fps (the frame rate of the depth sensor). Please also refer to the Video Figure.

Tasks

Participants completed a series of small tasks, organized into three categories. Task order was randomized per participant to mitigate order effects. For each task, users were instructed to stand along one of the two long edges of our test table (thus changing the orientation of their touches).

Crosshair: Participants placed their fingertip on a projected crosshair (Figure 6a), after which the experimenter manually advanced the trial and the touches detected by each tracker were recorded. This task measured the positional accuracy of each finger tracking method and touch segmentation accuracy. Crosshairs were arranged in a 4x8 grid spanning the table surface, but were shown one at a time and in random order. This task was performed twice for each edge of the table.

Multitouch Segmentation: Participants were instructed to place a specific number of fingers within a projected 20 cm square on the table (Figure 6b). The experimenter manually advanced the trial and the number of touches reported within the box for each tracker was recorded. This task was intended to measure the multitouch contact reporting accuracy of each technique (*i.e.*, false positive and false negative touches). Six boxes were specified across the length of the table, and the number of fingers varied from 1-5 for a total of 30 trials, randomly ordered. This task was also performed twice for each edge of the table.



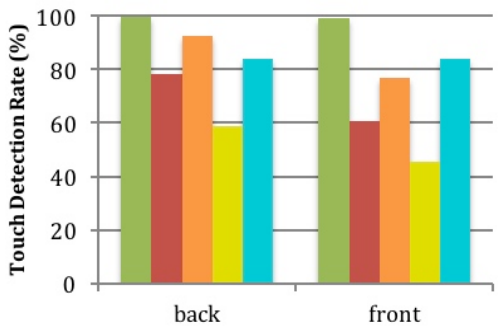


Figure 7. Average touch positional error (left) and touch detection rate (right) for each of the five touch tracking methods. Error bars are standard error.

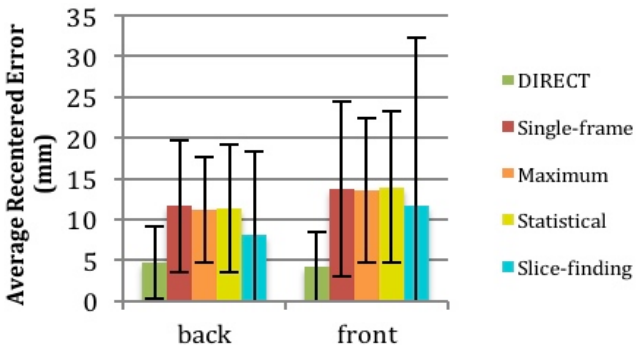


Figure 8. Average positional error after removing the average offset vector and assuming *a priori* knowledge of the user's orientation. Error bars are Standard Error.

Shape Tracing: Participants were instructed to trace a particular projected shape, beginning at the start position indicated with a green triangle (Figures 6c, 6d), and tracing to the end of the path. For each frame between the start and end of the trial, we recorded the touch coordinates for each method. This task was intended to replicate the tracing task used in OmniTouch [6]. There were three instances of this task per table edge, one for each shape: horizontal line, vertical line, and circle.

RESULTS AND DISCUSSION

In our results, we denote the two edges of the table as "back" and "front". The top of the Kinect's image corresponded to the front edge of the table.

Crosshair

The crosshair task allowed us to test both touch positional accuracy and touch detection rate. Due to potential spuriously-detected touches, we measured accuracy as the mean distance from the finger to the *nearest* detected touch point for each tracker. Touches further than 200 mm from the finger were not counted, since those touches would clearly be erroneous. If no touches were detected in range for a particular tracker, the tracker was considered to have failed to detect the touch.

In total, we collected 768 trials for each side of the table. The touch positional accuracy results are summarized in Figure 7. The accuracy results show a slight but consistent increase in accuracy across all trackers when users stood at the front edge of the table.

DIRECT achieved an average Euclidean-distance positional error of 4.8 mm across all trials, with a 99.3% touch detection rate. The next best technique, slice finding, had an average positional error of 11.1 mm and a touch detection rate of 83.9%. The background modeling methods all performed poorly, with average positional errors of over 40 mm and touch detection rates ranging from 52.1% to 84.8%. Put simply, these methods do not have the necessary sophistication to segment small finger contacts at the noise level present when sensing at 1.6 meters.

During development, we noticed that the slice finding method without forward projection performed very poorly (~20 mm average error), so it was clear that finger forward projection was crucial to obtain good accuracy. This is because these methods cannot accurately locate the fingertip in the noise, and so they instead locate a point somewhere along the finger.

Therefore, we also analyzed the accuracy of the four competing approaches by applying a mean offset vector (*i.e.*, a post hoc global offset). This vector depends on knowing the precise finger orientation, and thus the offset correction corresponds to a "calibrating" of the touch algorithm from a *fixed* user position and assuming the finger is extended perpendicular to the table. Consequently, we computed offsets separately for the front and back user positions. Because the prior systems recognize neither the user position nor finger orientation, this result is purely *hypothetical*, but serves as a useful benchmark.

The resulting average *offset-corrected* errors (Figure 8) were 4.46 mm for DIRECT (a negligible 0.3 mm improvement), 9.9 mm for the slice finding method (a modest 1.2 mm improvement), and 12.3-12.7 mm for the back-

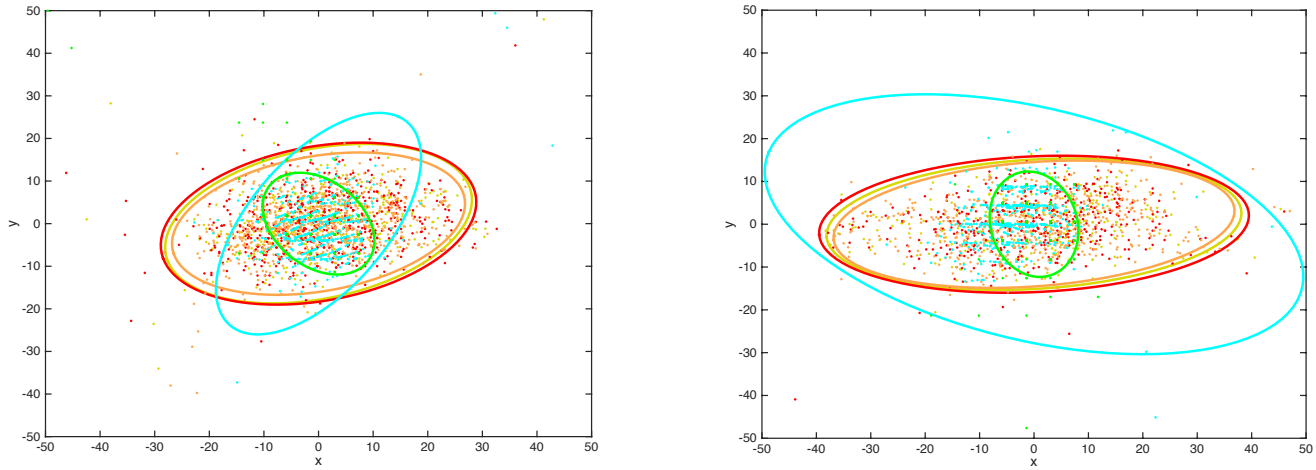


Figure 9. 95% confidence ellipses for crosshair task from the back of the table (left) and front of the table (right). X and Y units are in millimetres; colours are as in Figure 8.

ground modeling approaches (a significant 20-30 mm improvement). To visualize these errors, we also computed the 95% confidence ellipsoids (Figure 9) for each tracker (note that the offset correction corresponds to recentering the ellipsoids). These errors are consistent with prior results from Wilson [24] and OmniTouch [6], suggesting that our offset-corrected comparison implementations are reasonably close to the original implementations.

Multitouch Segmentation

With the multitouch segmentation task, we aimed to measure the false positive and false negative rates for touch detection with multiple fingers present in a small region. Under these conditions, touch trackers might merge adjacent fingers or detect spurious touches between fingers. The differences between the back and front sides were not significant in this task, so the results have been combined. In total, 1440 trials were collected.

Detecting a single extended finger is the easiest task. In single finger trials, DIRECT detected the correct number 95.8% of the time. Single-frame background, maximum frame background and statistical model background achieved 52.8%, 66.3% and 35.1% respectively. Slice-finding was 75.0% accurate.

Detecting several fingers in close proximity is much more challenging when sensing at 1.6 meters. With all trials combined, DIRECT detected the correct number of fingers in 75.5% of trials, more fingers than were present in 2.4% of trials and fewer fingers than were present in 22.1% of trials. The three background modeling approaches – single-frame, maximum frame and statistical model – detected the correct number of fingers 22.2%, 29.2% and 17.3% of the time. Very few trials reported more fingers than were present: 9, 7 and 4 trials respectively (<0.1% of all trials). Instead, these methods tended to miss fingers (77.1%, 70.2%, and 82.4% of trials respectively). Finally, the slice-finding approach detected more fingers in just 9 trials (<0.1% of trials), fewer fingers in 75.4% of trials, and the true number of fingers in 24.0% of trials.

We tuned the comparison technique implementations to minimize spurious touches while nothing was touching the table. However, our multitouch segmentation results suggest that optimizing for this criterion could have rejected too many legitimate touches, reducing touch detection rates. On the other hand, increasing touch sensitivity significantly increases noise and errant touches. For example, decreasing the "low boundary" depth threshold in the maximum-distance background model tracker by a single millimeter results in hundreds of errant touches detected on the surface every second, which is clearly not acceptable.

Shape Tracing

Tracking moving fingers is extra challenging as the exposure time of the depth camera produces motion blur across the moving parts of the frame, reducing accuracy. Further, as already mentioned above, our comparative methods are prone to missing fingers. In the case of finger movement, this will manifest as loss of finger tracking for several frames. During this period, spurious input would often cause the traced path to zigzag. For this reason, it was simply not possible to complete a realistic and useful analysis with our study data.

However, we can use results reported in OmniTouch [6] as one point of comparison. Specifically, OmniTouch reports a mean error of 6.3 mm (SD=3.9 mm) at a sensing distance of 40 cm on a flat notepad. For comparison, DIRECT achieves a mean error of 2.9 mm (mean SD=2.7 mm) at a sensing distance of 160 cm (on a flat table).

CONCLUSION

We have presented DIRECT, a touch tracking system which strategically merges depth and infrared data from an off-the-shelf infrared depth camera to enable highly accurate touch tracking, even at significant distances from the sensor. Overall, DIRECT demonstrates greatly improved touch tracking accuracy (mean error of 4.9 mm) and detection rate (>99%) – roughly twice as good as the next best method in the literature, and nearly ten times better than classic approaches.

There are also immediate ways to further improve our system. For example, in our present implementation, DIRECT outputs integer coordinates on the depth map, which quantizes the X/Y position to 4.4 mm (mean error due to quantization: 3.1 mm). Averaging pixels at the tip could provide a sub-pixel estimate, further boosting accuracy. Additionally, we could apply temporal smoothing to improve touch position stability at the expense of latency.

To conclude, we hope that DIRECT's significantly improved finger tracking accuracy can open new opportunities in ad hoc touch interfaces and better allow this emerging computing modality to be explored. We also believe this advance can help move depth-driven systems from proof-of-concept to more practical and widespread use.

ACKNOWLEDGMENTS

This research was generously supported by the David and Lucile Packard Foundation, Qualcomm, and a Google Faculty Research Award.

REFERENCES

1. Benko, H., Jota, R. and Wilson, A. Miragetable: free-hand interaction on a projected augmented reality tabletop. In *Proc. CHI '12*. 199–208.

2. Canny, J. A Computational Approach To Edge Detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8(6), 1986, 679-698.

3. Eastman Kodak Company. Kodak's Ergonomic Design for People at Work, 2nd Edition. 2003, pp. 48-49.

4. Haley, J. Anthropometry and mass distribution for human analogues. Volume 1, 1988. Aerosp. Med. Res. Lab Wright-Patterson, Ohio.

5. Han, J. Y. Low-cost multi-touch sensing through frustrated total internal reflection. In *Proc. UIST '05*. 115-118.
6. Harrison, C., Benko, H. and Wilson, A.D. OmniTouch: wearable multitouch interaction everywhere. In *Proc. UIST '11*. 441-450.
7. Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A. and Fitzgibbon, A. KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera. In *Proc. UIST '11*. 559-568.
8. Kane, S., Avrahami, D., Wobbrock, J., Harrison, B., Rea, A., Philipose, M. and LaMarca, A. Bonfire: a nomadic system for hybrid laptop-tabletop interaction. In *Proc. UIST '09*. 129-138.
9. Koike, H., Sato, Y. and Kobayashi, Y. Integrating paper and digital information on EnhancedDesk: a method for realtime finger tracking on an augmented desk system. *ACM Trans. on Computer-Human Inter.*, 8(4), 307-322.
10. Krueger, M. W., Gionfriddo, T. and Hinrichsen, K. VIDEOPLACE — an artificial reality. In *Proc. CHI '85*. 35-40.
11. Lee, S. K., Buxton, W. and Smith, K. C. A multi-touch three dimensional touch-sensitive tablet. In *Proc. CHI '85*. 21-25.
12. Maeda, J., Iizawa, T., Ishizaka, T., Ishikawa, C. and Suzuki, Y. Segmentation of Natural Images Using Anisotropic Diffusion and Linking of Boundary Edges. Pattern Recognition, 31(12), 1998.
13. Matsushita, N. and Rekimoto, J. HoloWall: designing a finger, hand, body, and object sensitive wall. In *Proc. UIST '97*. 209-210.
14. NASA. Anthropometry and Biomechanics. NASA-STD-3000: Man-Systems Integration Standards, Volume 1, Section 3. Revision B, July 1995.
15. Paradiso, J., Hsiao, K., Strickon, J., Lifton, J. and Adler, A. Sensor Systems for Interactive Surfaces. IBM Systems Journal, Volume 39, Nos. 3 & 4, October 2000, pp. 892-914.
16. Paradiso, J., Leo, C., Checka, N. and Hsiao, K. Passive acoustic sensing for tracking knocks atop large interactive displays. In *Proc. IEEE Sensors '02*. 521-527.
17. Saba, E.N., Larson, E.C. and Patel, S.N. Dante vision: In-air and touch gesture sensing for natural surface interaction with combined depth and thermal cameras. In *Proc. IEEE ESPA '12*. 167-170.
18. Steimle, J., Jordt, A. and Maes, P. Flexpad: highly flexible bending interactions for projected handheld displays. In *Proc. CHI '13*. 237-246.
19. Sugita, N., Iwai, D. and Sato K. Touch Sensing by Image Analysis of Fingernail. In *Proc. SICE Annual Conference '08*. 1520-1525.
20. Wang, F. and Ren, X. Empirical evaluation for finger input properties in multi-touch interaction. In *Proc. CHI '09*. 1063-1072.
21. White, R. M. *Comparative anthropometry of the hand*. No. NATICK/CEMEL-229. Army Natrick Research and Development Labs, MA. Clothing Equipment and Materials Engineering Lab, 1980.
22. Wilson, A. PlayAnywhere: A Compact Interactive Tabletop Projection-Vision System. In *Proc. UIST '05*. 83-92.
23. Wilson, A. TouchLight: An Imaging Touch Screen and Display for Gesture-Based Interaction. In *Proc. ICMI '04*. 69-76.
24. Wilson, A. Using a depth camera as a touch sensor. In *Proc. ITS '10*. 69-72.
25. Xiao, R., Harrison, C. and Hudson, S. E. WorldKit: Rapid and Easy Creation of Ad-hoc Interactive Applications on Everyday Surfaces. In *Proc. CHI '13*. 879-888.
26. Xiao, R., Lew, G., Marsanico, J., Hariharan, D., Hudson, S.E. and Harrison, C. Toffee: enabling ad hoc, around-device interaction with acoustic time-of-arrival correlation. In *Proc. MobileHCI '14*. 67-76.
27. Xiao, R., Schwarz, J. and Harrison, C. Estimating 3D Finger Angle on Commodity Touchscreens. In *Proc. ITS '15*. 47-50.