Detecting and Leveraging Finger Orientation for Interaction with Direct-Touch Surfaces

Feng Wang[1], Xiang Cao[2], Xiangshi Ren[1] and Pourang Irani[3]

[1]School of Information
Kochi University of Technology, Japan
wangfeng@acm.org
ren.xiangshi@kochi-tech.ac.jp

[2]Microsoft Research Cambridge
Cambridge CB3 0FB
United Kingdom
xiangc@microsoft.com

[3]Computer Science Dept.
University of Manitoba
Canada
irani@cs.umanitoba.ca

ABSTRACT

Current interactions on direct-touch interactive surfaces are often modeled based on properties of the input channel that are common in traditional graphical user interfaces (GUI) such as x-y coordinate information. Leveraging additional information available on the surfaces could potentially result in richer and novel interactions. In this paper we specifically explore the role of finger orientation. This property is typically ignored in touch-based interactions partly because of the ambiguity in determining it solely from the contact shape. We present a simple algorithm that unambiguously detects the directed finger orientation vector in real-time from contact information only, by considering the dynamics of the finger landing process. Results of an experimental evaluation show that our algorithm is stable and accurate. We then demonstrate how finger orientation can be leveraged to enable novel interactions and to infer higher-level information such as hand occlusion or user position. We present a set of orientation-aware interaction techniques and widgets for direct-touch surfaces.

ACM Classification: I.3.6 [Methodology and Techniques]: Interaction techniques.

General terms: Design, Human Factors.

Keywords: Finger orientation, direct-touch surface, orientation aware interface.

INTRODUCTION

Interactive techniques on touch and multi-touch surfaces are generating significant appeal in the general public due to their inherently natural affordances. One main reason for this naturalness is derived from the ability to let users employ their bare fingers and directly manipulate the system without intermediary devices. Researchers have demonstrated that direct-touch interactive displays offer a more compelling method to interact with a system than working indirectly with a mouse or with other types of pointing devices [11, 15, 22, 30, 37] .

UIST'09, October 4–7, 2009, Victoria, British Columbia, Canada.

However, current multi-touch designs are mainly based on multi-point information, i.e. the touch-sensitive devices primarily use the center coordinates of the human finger's contact region as cursor positions. Therefore, most interactions mainly rely on touch positions or variations in touch movements. Relatively few research demonstrations have used auxiliary information other than touch position, such as the shape [7, 36] or size of the contact region [3] for enhancing the naturalness of the interaction.

One potentially accessible piece of information from a finger's contact point is its orientation. Orientation is a natural cue as it provides the direction a user is pointing in and is used in many daily interactions, such as pointing to communicate with others, to acquire or position an object, or even to lead and direct attention. If we consider an orientation vector consisting of a direction and an angle from a point of reference, very few systems have used this to enhance the naturalness of the interactions. By extracting the longitudinal axis of a finger's contact shape, it is straightforward to detect the undirected angular configuration (0°~180°) of a straight line that approximates the user's finger. However, this result is ambiguous. The exact orientation vector of the finger could be one of two opposite directions that align with the undirected line, making it difficult to utilize the exact direction of the finger for interaction (Figure 1a).

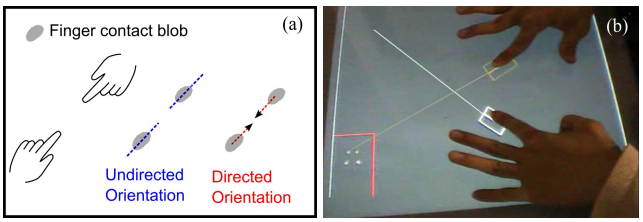

Figure 1. (a) Undirected orientation vs. directed orientation vector of the finger. (b) Orientation detection in action.

To address this limitation, we present a novel and robust algorithm that accurately and unambiguously detects the orientation vector by considering the dynamics in finger contact (Figure 1b). This algorithm is general enough for any direct-touch surface that generates contact shape information, without resorting to additional sensors. We demonstrate that finger orientation information is key in the design of orientation-aware interactions and widgets, for example to facilitate target selection, or to optimally orient

elements in the workspace to adapt to the user's position. Additional information about the user can also be inferred from finger orientation, such as hand occlusion region or the position of the user. These cues can in turn be leveraged to further enrich the interaction on touch-surfaces.

RELATED WORK

Direct-Touch Surface Technologies

In recent years we have witnessed an exponential growth of technologies to support direct-touch interactive surfaces [2, 9, 11, 15, 18, 19, 29, 34-37]. DiamondTouch [9], one of the earlier multi-touch systems allows multiple users to simultaneously interact on a tabletop. Through capacitive coupling it associates touch regions with each user, useful for supporting user-specific operations. The rough shape of each finger contact is determined through an antenna matrix.

Computer-vision-based technologies are also widely employed to enable direct-touch surfaces. Han et al. [15] introduced a multi-touch system based on Frustrated Total Internal Reflection (FTIR), which recovers the surface regions being depressed by fingers. An alternative approach is based on Diffuse Illumination (DI), which detects not only the contact regions but also fingers hovering above the surface within a certain distance. This is used on systems such as the Microsoft Surface [21]. Compared to capacitive-based sensing, vision-based systems provide higher fidelity in detecting the contact shape.

Briefly, current interactive surfaces provide contact shape of the fingers, regardless of the underlying technology. This is the basis of our algorithm, which aims at determining the finger orientation from contact shape input only.

User Interfaces Design

With advances in hardware functionality and improved features with multi-touch systems, researchers are expending parallel efforts in designing new techniques and leveraging upon novel hardware designs. We relate our work to two general categories of interaction techniques: techniques that replicate mouse features on direct-touch surfaces; and techniques that leverage upon additional hand and finger properties.

Adapting Mouse Interactions. There exist several motivations for adapting mouse-based interactions on direct-touch surfaces. Touch-based interactions are known to result in imprecise selections. Researchers have proposed numerous solutions to improve the precision of bare finger interactions [1, 3, 25, 28, 32] and these solutions can be categorized as follows: direct touch improvement [28], cursor offset [3, 28, 32], target zoom-in or control-display ratio adjustment [4, 27] and on-screen widget [1] to precisely select a target. Furthermore, researchers have explored the benefits of using multi-point input to interact with traditional GUI elements [23-25].

To ensure compatibility with traditional legacy applications, researchers have studied cursor control and mouse simulation techniques. The DiamondTouch-mouse [10] supports a right-click by tapping with a second finger. DTMouse [11] further enhances the functionality of the DiamoundTouch-mouse by addressing issues such as mouse-over, smooth toggling of left mouse button, ergonomics and precise input. In DTMouse, states of the mouse were determined based on timeout intervals of holding a finger down.

Matejka et al. [20] presented SDMouse to emulate the functionality of a conventional mouse, including a tracking state, three buttons and chording. The first finger down is used as a tracking finger, and the combination of two or three fingers are used to trigger left-click, right-click or scrolling events based on the side and the distance of finger touch points. However, these systems are based solely on extracting the coordinates of the finger contact points on the screen. Such systems can be unstable or need redefinition if the user triggers these states in a different orientation.

Leveraging Additional Finger Properties. In addition to using the center coordinates of the contact region, researchers have proposed techniques that use finger or hand properties for new interactions. Benko et al. [3] proposed the use of contact size to simulate pressure input on the tabletop. They introduced rocking and pressing gestures to define various states, including a "click" event. Wilson et al. [36] used the contact contours to emulate physical reactions between touch input and digital objects. Cao et al. [7] presented ShapeTouch that leverages the contact shape to enable richer interactions similar to those in the real world. Davidson et al. [8] demonstrated a pressure-based depth sorting technique using a pressure-sensitive surface, which extends standard two-dimensional manipulation techniques, particularly those controlled by multi-touch input.

Finger orientation was firstly adopted by Malik et al. [19] in the Visual Touchpad system. The system uses a pair of overhead cameras to track the entire hand of the user, and infers finger orientations accordingly. Microsoft Surface [21] determines full finger orientations by leveraging additional hover information enabled by the DI technology. Both these approaches rely on a specific sensing technology, and are therefore not generally applicable to other systems. These systems also did not investigate interaction designs that specifically utilize finger orientation.

Wang and Ren [33] empirically investigated finger contact properties such as size, shape, width, length and orientation using a FTIR-based multi-touch surface. They have speculated a few interaction designs leveraging finger orientation, but did not provide an algorithm to detect finger orientation in real time. Inspired by some of their findings, we propose in this paper a general finger orientation detection algorithm that requires contact shape

information only. We then further explore novel interactions that leverage upon finger orientations.

FINGER ORIENTATION DETECTION ALGORITHM

In this section we describe our algorithm that detects the directed orientation vector of the user's finger, based on real-time information collected from the shape of the finger contact. Here the finger orientation refers to the 2D orientation of the finger's projection on the surface. Prior literature [3, 12] points at two types of finger touch on interactive surfaces: *vertical touch* and *oblique touch* (Figure 2). A vertical contact occurs when the finger is directly pointing downward, toward the surface (Figure 2a). Obviously this does not provide usable orientation information. Conversely, an oblique touch occurs when the finger lands on the surface at an oblique angle (Figure 2b). Considering common practices handling physical objects, as well as the necessity to accommodate long fingernails by some people (especially women), we expect that the oblique touch is more likely to happen when people touch interactive surfaces. A unique finger orientation can be determined from an oblique touch, which is the basis of our algorithm.

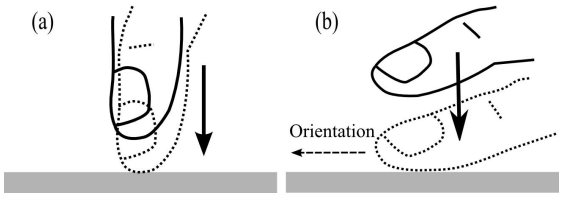

Figure 2. Two ways of finger touch. (a) vertical touch. (b) oblique touch.

For each frame of input that contains all contact pixels on the surface, we first conduct a connected component analysis to extract all finger contact regions. Then for each finger contact, its orientation is determined by our algorithm. The algorithm has four major steps: fitting the contact shape; detecting the oblique touch; disambiguating the finger direction; and continually tracking the orientation.

Fitting Contact Shape

When a finger obliquely touches the surface, its contact region appears as an elliptic shape (Figure 3). This shape can then be fitted into a perfect ellipse described by Equation 1 using least-square fitting, as presented in [33]:

$$\left(\frac{(x-x_0)\cos\theta+(y-y_0)\sin\theta}{length/2}\right)^2+\left(\frac{(y-y_0)\cos\theta-(x-x_0)\sin\theta}{width/2}\right)^2=1 \quad (1)$$

The *length* (magnitude of the long axis), *width* (magnitude of the short axis), and slant angle θ ($0 \le \theta \le \pi$) describe the shape of the ellipse; and (x_0, y_0) is the center coordinate of the finger contact. The *area* of the contact region can be calculated by simply counting the pixels within it. The slant angle θ describes the undirected orientation of the finger.

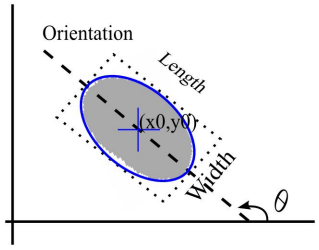

Figure 3. Finger contact region fitted to an ellipse. Width, length and slant angle can be obtained from results of the fit.

Identifying Oblique Touch

In order to generate reliable finger orientation, we need to determine whether the finger is currently in an oblique touch state. Two properties of the finger contact region are critical for identifying an oblique touch: *area* and *aspect ratio*. Equation 2 shows the identification criteria:

$$\begin{cases} area > ta \\ aspect\ \ ratio = \dfrac{length}{width} > ts \end{cases} \quad (2)$$

where *ta* and *ts* are empirically determined thresholds. Both criteria need to be satisfied for an oblique touch to be identified, otherwise we consider the finger to be in either vertical or accidental touch and ignore its orientation. These two criteria were determined based on pilot trials and previous literature.

Area Criterion. As found in prior investigations of finger input properties [33], the contact area is significantly different in vertical and oblique touches. The mean contact area in vertical touch is between 28.48 and 33.52 mm^2, whereas the mean contact area in oblique touch is significantly larger (between 165.06 and 292.99 mm^2). After further validation by pilot trials, we set the threshold *ta* to be 120 mm^2.

Aspect Ratio Criterion. Area alone is not reliable enough to identify an oblique touch because a large contact area can also result from pressing harder in a vertical touch. The undirected finger orientation information is stable only when the finger contact is elongated. The larger the aspect ratio (i.e., the more oblique the finger is), the more accurate is our estimation of the orientation. In our algorithm, based on pilot trials of comfortable manipulations, we set the aspect ratio threshold *ts* to be 120%.

Disambiguating Finger Direction

From the contact shape fitting step we have acquired the undirected finger orientation θ. However, the true direction of the finger could be either θ or $180° + \theta$. The key innovation of our algorithm is to resolve this ambiguity by considering the dynamics in the finger landing process.

The human finger has soft and deformable tissues. The distortion of the finger muscle is inevitable upon contact. Since it is difficult to extract full orientation from a single finger contact, we instead closely examine the deformation of the finger's contact region in the process of it landing on

the surface. Figure 4 shows the contact region across time when a finger is landing.

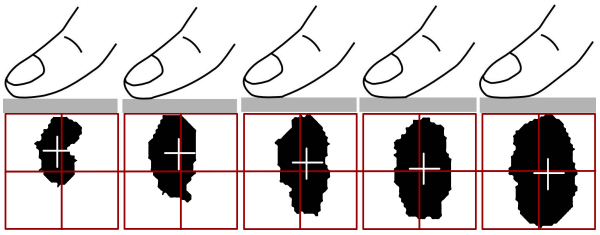

Figure 4. Finger contact deformation over time. The crosshair shows the center of the contact region.

It is apparent that the center point of the finger contact moves inward, towards the user's palm. This movement can be explained by closely examining the landing process in an oblique touch: the finger tip gets in contact with the surface first; as the pad of the finger increases its contact area the center of the contact region shifts inward.

Considering the finger's deformation, by tracking the variation of the contact center during the landing process, we can roughly infer which side the user's palm lies in, and in turn which direction the finger is pointing to. Figure 5 shows the variation of the contact center between two consecutive frames t-1 (blue) and t (red). Frame t-1 is the last frame of non-oblique touch state and frame t is the first frame of oblique touch state. We can then calculate angle α as a rough estimation of the directed finger orientation by taking the azimuth angle of vector $(-\Delta x, -\Delta y) = (x(t\text{-}1) - x(t), y(t\text{-}1) - y(t))$, which points away from the palm.

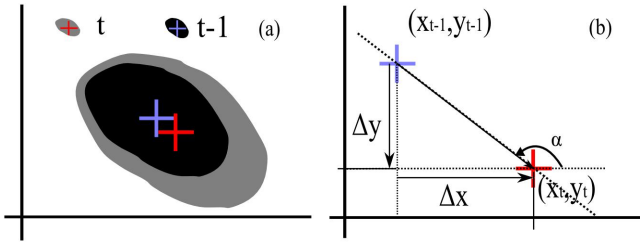

Figure 5. Finger direction disambiguation. (a) Finger contact at frame t-1 and t. (b) Rough estimation of directed finger orientation.

Finally, α is used as the cue to disambiguate the undirected finger orientation θ, so that the final directed finger orientation Φ is consistent with α.

$$\Phi = \begin{cases} \theta & (|\alpha - \theta| \le 90°) \\ \theta + 180° & (|\alpha - \theta| > 90°) \end{cases} \quad (3)$$

Continual Orientation Tracking

Once the complete finger orientation has been unambiguously determined for one frame using the previous step, this step need not be repeated for the following frames. The orientation disambiguation of the following frames then depends on the fact that no abrupt change of orientation will occur between each two consecutive frames. Due to the finger's range-of-motion and the limitations imposed by the physical anatomy of the finger, the variation in the finger's orientation is likely to be very gradual.

In every subsequent frame t+1, the directed finger orientation in the previous frame $\Phi(t)$ is used as the cue to disambiguate the current undirected finger orientation θ(t+1), i.e.:

$$\Phi(t+1) = \begin{cases} \theta(t+1) & (|\Phi(t) - \theta(t+1)| \le 90°) \\ \theta(t+1) + 180° & (|\Phi(t) - \theta(t+1)| > 90°) \end{cases} \quad (4)$$

PERFORMANCE EVALUATION

Goal

We conducted an experimental evaluation to assess the performance of our finger orientation detection algorithm, including the stability and precision in determining orientation of static and dynamic fingers. We note that given the irregular shape of a finger and different viewing perspectives, in practice the finger orientation is largely subject to human interpretation. An objective "true value" does not exist in a strict sense. Instead, to best inform interaction designs, in this evaluation we compare the detected orientation to the finger orientation subjectively perceived by the user; this is what users would rely on for real interactions if no visual feedback is provided.

Apparatus

The apparatus we used in the study is a direct-touch tabletop surface based on Frustrated Total Internal Reflection (FTIR) technology [15]. The tabletop is approximately 27"×18" in size, and 0.8m in height. A camera installed beneath the surface captures the input image, working at a resolution of 640×480 pixels and at a capture rate of 30 fps. The experimental software is built upon the Touch-Lib open source API [26], augmented by our orientation detection algorithm. The system runs on a 2.4GHz Duo Core PC with Windows XP SP2 OS.

Participants

Eight volunteers, four male and four female, 26-37 years old, participated in the experiment. All were right-handed and had no prior experience with direct-touch surfaces.

Task

We evaluated the algorithm with four tasks, each examining a different aspect of the algorithm. The participant sat in front of the tabletop and used the right index finger to complete each task. We did not provide any visual feedback concerning the orientation of the finger as detected by the algorithm. As a result, participants had to completely rely on their subjective perception of the finger orientation.

Task 1: Static Orientation Stability. This task examines the stability of the algorithm when the finger is kept still. The participant touches the surface at an arbitrary position and finger orientation, and dwells in the position for more than 5 seconds. The user lifts the finger when prompted by the experimenter. All the values of finger orientation were recorded and the data up to 5 seconds are used to evaluate the orientation stability during this period.

Task 2: Static Orientation Precision. This task examines the orientation detection precision for a static finger. A red cross is displayed on the surface. An arrow on the red cross indicates the orientation to point at by the participant. Participants touch the center of the cross while matching the finger orientation as accurately as possible to the direction of the cross (Figure 6a). We use four directions for the task: 165°, 150°, 135°, and 120° (counterclockwise from east), as these can be comfortably achieved by the right index finger. If necessary, the participant can further adjust the finger orientation after landing on the surface. Once satisfied, the participant presses a key on the keyboard to indicate completion. We record the detection error (detected finger orientation minus actual arrow direction, at the moment of task completion) for each trial.

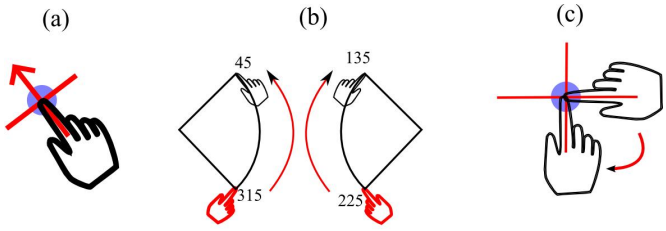

Figure 6. Evaluation tasks. (a) Static orientation precision task. (b) Dynamic orientation precision task. (c) Involuntary Position Variation task.

Task 3 - Dynamic Orientation Precision. This task examines the orientation detection precision when the finger is moving and rotating on the surface. Participants trace the index finger along a circular arc displayed on the surface from a start to an end point (Figure 6b). At any given point during the movement, the finger orientation needs to be aligned with the arc (i.e. its tangential direction at the point) as precisely as possible. Two arcs are used for the task: counterclockwise from 315° to 45°; and clockwise from 225° to 135°. This arc tracing task effectively enables us to continuously acquire the (perceived) orientation "ground truth" during a dynamic operation. For each point on the arc, we record the detection error (detected finger orientation minus tangential direction at the point).

Task 4 – Involuntary Position Variation in Rotation. This task examines the involuntary variation of finger position coordinates (x, y) associated with a finger rotation. An involuntary variation in position occurs when the user incidentally moves the center coordinate of the finger during a rotation. In this task, the participant placed the finger on a red cross displayed on the surface, and rotated it clockwise from 0° to 270° while keeping the finger position static (Figure 6c). We record the range in variation of the finger's center position during the rotation.

For all tasks, the participant was asked to repeat the trial if the orientation disambiguation result was incorrect (i.e., providing the opposite orientation). Within each task, all trials were randomly ordered to prevent practice effects.

Design

Participants performed each task with six repetitions. The full evaluation consisted of: 8 participants × [Task-1 + (4 orientations in Task-2) + (2 arcs in Task-3) + Task-4] × 6 repetitions = 384 trials in total.

Result

Disambiguation Success Rate. For all tasks, the disambiguation algorithm generated 13 errors in total. This resulted in a success rate of 96.7% (384 out of 397 trials), indicating good performance of the algorithm.

Static Orientation Stability (Task 1). The average variation range during each finger dwelling period is 0.59° (std. dev.= 0.15°). This demonstrates that our algorithm is very stable. The low level of random noise is caused by both the user's unconscious finger jitter, and the imaging noise introduced by the camera. According to this result, in practice we can ignore finger orientation changes that are less than 1°.

Static Orientation Precision (Task 2). Results of Task 2 show that the detected finger orientation matches closely with the finger orientation perceived by the user. The average detection error (absolute value) is 2.69° (std. dev. = 1.76°). ANOVA showed no significant difference between the detected finger orientation and the perceived orientation, indicating the detection error was not biased towards one specific direction. When considering the signs of the error, we obtained an upper bound of +5.84° and a lower bound of -4.70° at the 95% confidence interval.

Note that this is the detection error when there is no visual feedback and therefore incorporates both the imprecision of the algorithm and variations in user perception. This indicates that for interactions that involve a single touch action without visual feedback, our algorithm can provide a precision within approximately ±5°. Across the complete 360° orientation range, this gives 36 usable orientation levels (each with the tolerance interval of 10°) that can be reliably detected for interaction. However, in the presence of visual feedback, the user can adjust their input accordingly and perform closed-loop actions with much higher accuracy (~1°) as suggested by our results on static orientation stability.

Dynamic Orientation Precision (Task 3). The results of task-3 show that the continual orientation tracking algorithm is reasonably accurate across the whole movement range. The average orientation error (absolute value) is 14.35° (std. dev. = 9.53°). Again ANOVA showed no significant difference between the detected finger orientation and the perceived orientation, indicating the lack of systematic detection bias. The upper and lower bound of signed error at the 95% confidence interval was +29.69° and -26.81° respectively. This increased error is partly explained by the difficulty for the user to smoothly and precisely control finger orientation during finger movement. Based on our observation, instead of continuously rotating the finger throughout the trial, most participants made discrete compensation changes of finger orientation when they

noticed it deviated from the arc. This observation is likely to hold in other actions of simultaneous finger movement and rotation as well. According to this, in interaction designs we should ideally avoid requiring the user to precisely control the finger orientation while moving the finger, especially if no visual feedback is provided. No significant difference in orientation detection resulted between clockwise and counterclockwise movements.

Involuntary Position Variation in Rotation (Task 4). The average position variation during finger rotation was 2.02mm for x-coordinate (std. dev. = 0.96mm); and 2.00mm for y-coordinate (std. dev. = 1.08mm). Aside from detection noise, this variation can be explained by two factors: the user unconsciously moves the finger during rotation; and the user's perceived rotation center does not precisely match the finger center detected by the system. This variation in finger position needs to be taken into account when designing interactions based on finger rotation. Displacements under 2mm of the finger's position during rotation should be ignored in such interactions.

On the other hand, the detected orientation across each rotation trial shows that it changes continuously and smoothly at a relatively constant rate, different from the discrete jumps observed in Task 3. This confirms that the user is able to finely control finger orientation for interactions when the finger position is kept static. Combining this with results from Task 3, for interactions that involve both finger translation and rotation, the best strategy may be to let the user first move and rotate the finger simultaneously for coarse maneuver, but to also allow the user to "park" the finger for finer orientation adjustment in the end.

INTERACTIONS USING FINGER ORIENTATION

Finger orientation information as detected by our algorithm may lead to a set of new interaction designs.

Enhancing Target Acquisition

Finger orientation can be employed to design new target acquisition techniques on interactive surfaces.

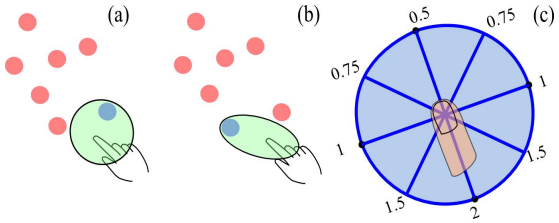

Figure 7. (a) Regular bubble cursor. (b) Directed bubble cursor. (c) Distance weights according to finger orientation.

Directed Bubble Cursor. Bubble cursor [13] is an efficient target selection technique that dynamically resizes the activation region of an area cursor [17] so that it always select the one target at the shortest distance to the center of the cursor (Figure 7a). On a direct-touch surface, we can further enhance the bubble cursor by considering the finger orientation, so that the selection is biased towards targets in front of the finger direction, and against targets to the back of it. The shape of the activation region may also become slightly skewed to reflect this (Figure 7b). This is realized by applying different multiplying weights to the target distances based on the target's relative azimuth angle compared to the finger center and orientation. Targets with an azimuth angle of 0° (i.e. in line with the finger orientation) have the smallest weight, and those with an azimuth angle of 180° (i.e. opposite to the finger orientation) have the largest weight (Figure 7c). By choosing the target with the shortest weighted distance, the directed bubble cursor displays a behavior that is consistent with real-world conventions when using a finger to refer to objects, where both finger position and direction play a role.

Aim and grab. The finger orientation can also be utilized to select objects that are far away on the interactive surface by distant pointing. To do so, the user touches the finger on the surface to cast a selection ray aligned to the finger orientation. The first object intersected by the ray gets selected. The user can rotate the finger to aim at different targets. To switch between multiple objects intersected by the selection ray, the user can move the finger forward or backward along the ray. This is similar to the Depth Ray selection technique proposed by [14] for 3D volumetric displays. A finger flick inward brings the selected object to the user (Figure 8).

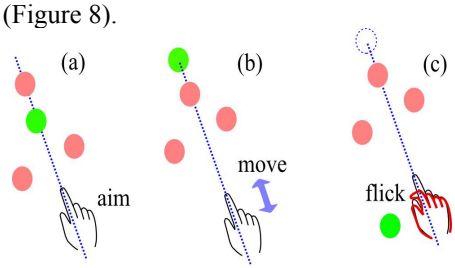

Figure 8. Aim and grab. (a) Aim finger to select an object. (b) Move finger along the orientation vector to switch between multiple objects. (c) Flick finger inward to bring the selected object.

As discussed in [33], we can also use the intersection of two selection rays determined by two fingers for precise selection of distant targets (Figure 1b).

Orientation-Sensitive Widgets

Finger orientation can be treated as an additional direct input dimension for interface widgets. Figure 9 shows two example designs of such orientation-sensitive widgets.

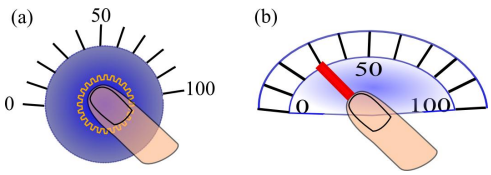

Figure 9. Orientation-sensitive controls. (a) Orientation-sensitive button. (b) Orientation dial.

An orientation-sensitive button (Figure 9a) allows the user to use the finger orientation to specify the parameter of the button functionality while hitting the button. By doing so, function invocation and parameter specification are

combined into a single step (a valuable substitute to widgets such as combo boxes on GUIs). As discussed previously, for such a widget with no continuous visual feedback we can support an orientation resolution of 10°.

An orientation dial (Figure 9b) allows the user to continuously adjust a parameter with high precision. Compared to other parameter adjustment widgets such as a slider, this supports a large range of parameter values while requiring minimal finger movement and screen estate.

INFERENCES FROM FINGER ORIENTATION

In addition to directly utilizing finger orientation for input, we can make further inferences about the user by considering the finger orientations and positions.

Estimating Occlusion Region

For the design of direct-touch interactions, hand and finger occlusion is often a major concern that cannot be entirely avoided. However, based on the position and orientation of the finger touch, we could effectively estimate the hand occlusion region on the fly, and adapt interface layouts to minimize the impact of occlusion.

Considering the anatomy of the human finger and palm, we estimate the occlusion region to be a circular sector opposite to the finger orientation Φ, with the vertex at the center of the finger tip (x, y), and the central angle at approximately $\delta = 60°$ (angular value selected from[5]; Figure 10).

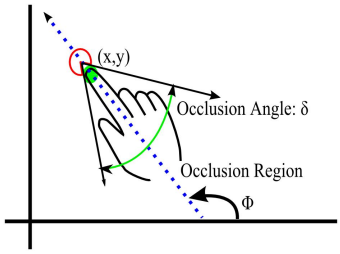

Figure 10. Occlusion region estimation.

With knowledge of the occlusion region, we can dynamically place content and interface elements outside it. In addition, we can design special interface widgets that adapt to accommodate the finger orientation and avoid occlusion. For example, a pie-menu or torus-menu with a gap can dynamically reorient itself so that the gap is always aligned with the body of the finger (Figure 11) [33]. Brandl et al. [6] explored similar occlusion-aware menu designs, but required the user to use a pen and rest the palm on the surface simultaneously to determine the menu orientation. Comparatively, our finger orientation detection algorithm allows these designs to be broadly applied to various scenarios and technologies.

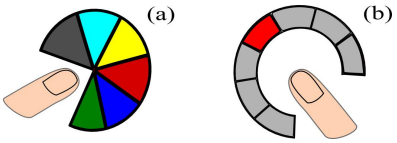

Figure 11. Menus adapting to finger orientation to avoid occlusion. (a) Pie menu. (b) Torus menu.

Inferring User Position

When a user puts a finger on a horizontal surface, typically the finger points away from the user's body (Figure 12). For tabletop interaction, this provides a simple cue to infer the rough position of the user. Although inherently imprecise, this can be useful enough for simple scenarios where the user can only take a few possible positions. In a typical interactive tabletop usage scenario, the user sits along either one of the two long sides of the tabletop. Knowing the orientation of the finger touch, we can infer that the operating user is sitting at the side opposite to the finger orientation. This information is particularly useful for orienting the interface and content (especially text) to suit the user's perspective.

Another common usage scenario is when two users sit on opposite sides of the tabletop. By applying the same heuristics, we provide a lightweight way of differentiating finger touch inputs from different users without resorting to technologies like the DiamondTouch [9]. Many user-specific operations can then be easily supported, such as the use of interface widgets that function differently depending on who triggers them, or setting different operation privileges for different users (Figure 12).

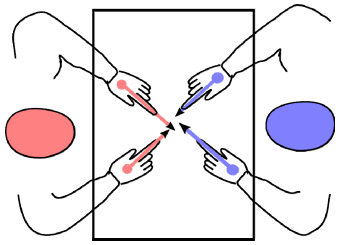

Figure 12. Inferring user position from finger orientation.

Relationship between Multiple Fingers

The anatomy of the human hand imposes certain constraints on the possible orientations and positions of fingers from the same hand. We can exploit this information to infer the relationship between multiple fingers on the surface.

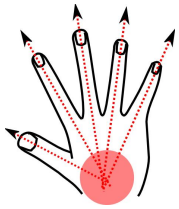

Figure 13. Typical hand configuration.

In natural and comfortable positions, the orientation of a finger indicates a departure away from the center of the palm. As a result, the lines of direction based on the orientations of two or more fingers from the same hand will intersect and provide a rough location of the user's palm. This location is usually to the opposite side of the directions pointed by all finger, and within a reasonable distance from the position of the fingertips (Figure 13).

Based on this information, for a pair of finger touch points, we calculate the intersection point I of the two straight lines aligned with their positions and orientations. For each fingertip position P with orientation Φ, we calculate the orientation angle Φ_{IP} of ray IP (i.e. pointing from I to P). If $|\Phi_{IP} - \Phi| < 90°$ and distance $|IP| < td$ (td is an empirically determined threshold, chosen to be 140 mm in our implementation) for both fingers, we determine that they belong to the same hand (Figure 14a). Otherwise the two fingers belong to different hands (Figure 14b). In the latter case we may also infer whether the two hands belong to different users in some simple cases (Figure 14c). As discussed previously, if assuming that the two users are sitting in fixed positions across a horizontal surface, and the orientations of the two fingers clearly point oppositely to where the users are supposed to be sitting, we can then associate these two hands with each user. When three or more fingers are touching, we may determine their pair-wise relationships, and make higher-level inferences if necessary. Obviously, this approach does not account for atypical cases such as when two hands overlap. However it would be reliable enough for interaction purposes in natural scenarios.

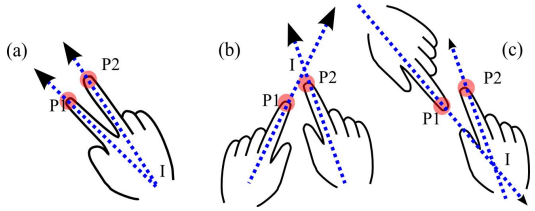

Figure 14. Inferring relationship between fingers. (a) Same hand. (b) Two hands from same user. (c) Two hands from different users.

Knowing the relationship between finger touches can be particularly useful for various interactions. Moscovich and Hughes [24] experimentally showed that multi-finger manipulations by one hand and by two hands are suitable for different tasks. Inspired by this, we could assign different functionalities for each case. For example, fingers from the same hand result in multi-finger inking on a digital object; fingers from both hands by the same user result in the classic rotation/scaling manipulation; and fingers from multiple users could "tear apart" the object to create multiple copies.

Enabling Orientation-Invariant Input

The direct-touch input on interactive surfaces naturally affords using the finger to perform trajectory-based gestures, similar to pen gestures that are broadly used on tablet PCs or handheld devices. Pen gestures performed on those devices usually have an unambiguous upright orientation relative to the input panel. However, for gestures performed on interactive surfaces, especially horizontal tabletops, the orientation of the gesture inputted can be arbitrary depending on the user's position. This creates a dilemma for unambiguous gesture recognition on interactive tabletops. Either the system has to assume the user is inputting from a fixed orientation, which constrains the usage of the tabletop. Alternately the system has to recognize input in a rotation-invariant way and avoid any orientation-specific gesture, which largely limits the gesture design space. This problem becomes even more prominent if we want to introduce handwriting recognition input on tabletops, since many of the English and numerical characters are inherently orientation-specific (Figure 15).

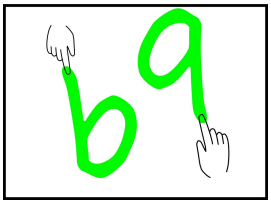

Figure 15. Ambiguous gestures caused by different hand orientations.

By taking finger orientation into account, this problem could be alleviated. Before recognition, the orientation of the input gesture can be normalized by a compensated rotation determined by the average finger orientation while performing the gesture. As a result, the user could perform finger gestures or handwrite unambiguously from any side of the tabletop. Blasko et al. [5] explored similar concepts on a handheld tablet by estimating its own orientation through face tracking or stylus pose.

Another example of orientation-invariant input is to support multi-finger mouse emulation on interactive tabletops. Matejka et al. [20] presented SDMouse, an efficient technique to simulate full mouse functionality by mapping different buttons to different fingers on a multi-touch screen. The technique differentiates "mouse buttons" partly by their directional position with reference to the index finger. For example, the finger on the left side of the index finger is considered the thumb and mapped to the left button. This poses a problem for migrating SDMouse onto a tabletop surface, where the definition of "a side" is ambiguous and varies with the hand orientation. Again, by considering the orientation of the index finger, we can unambiguously associate fingers to buttons located in a reachable location regardless of the user's position (Figure 16).

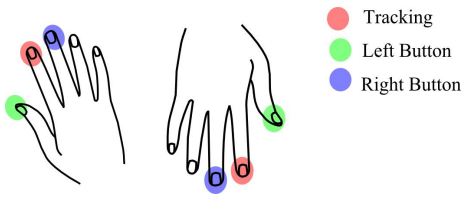

Figure 16. Orientation-invariant mouse emulation.

DISCUSSION

Algorithm Limitations

We have shown that our finger detection algorithm is effective and accurate. However a few limitations do exist:

The algorithm assumes an oblique touch from the user, which is the case for most common interaction scenarios.

However, as Forlines et al. [12] discussed, such an oblique touch may be less obvious when the user touches a vertical surface, or in areas of a tabletop that are very close to themselves. For these scenarios, the interaction designs would need to afford supplemental ways for the user to indicate the concept of orientation.

The orientation disambiguation step relies on the finger center displacement during the finger landing process and assumes that this displacement is caused solely by the deformation of the finger. This requires that the landing action should consist of a downward vertical movement of the finger only, which is typically true in regular cases. In the less frequent case that the finger landing is accompanied by a concurrent horizontal finger movement (i.e. "sliding down"), the finger disambiguation algorithm could be biased. However, the concurrent horizontal movement during the short period of finger landing is unlikely to be significant enough to reverse the disambiguation result.

Although our experimental evaluation only examined the algorithm performance with the index finger, in practice we observed that the algorithm also works well with most other fingers, with the only exception being the thumb. Restricted by the anatomy of the human hand, a user would typically touch the surface with the side face of the thumb, instead of its pad. This does not display the usual center displacement pattern as with other fingers, and usually results in an incorrect detection by our algorithm. On the other hand, when users do touch the surface with the pad of the thumb, they usually do so with the thumb pointing towards themselves, as opposed to it being away from themselves as with other fingers. Our algorithm can correctly detect the thumb orientation in this case. However this action is less informative for inferring the user's position.

It should be noted that using finger orientation alone is not the final answer for novel interaction designs on interactive surfaces. By combining finger orientation with other input properties of the hand such as size, shape, or pressure, the limitations of our algorithms would be overcome in interaction scenarios.

Technology Compatibility

We have tested our algorithm on a representative computer-vision-based technology. Our algorithm requires solely the contact shape of the finger to work. This input requirement makes our algorithm largely compatible with a variety of other sensing technologies, such as capacity-based sensing or embedded optical sensor arrays [16]. However given the nature of the different technologies, the parameters of the algorithm may need to be adjusted accordingly. For example, FTIR-based devices often require the user to press slightly harder to generate enough touch area for processing. On the other hand, technologies that provide certain additional information such as hover state, as in ThinSight [16], can utilize these to further improve the reliability of our algorithm.

3D Finger Orientation

In this paper we focused on the 2D orientation of the finger. However, 3D finger orientation may also be interesting for interacting with digital surfaces. This information could potentially be acquired by using new sensing technologies such as depth cameras (www.3dvsystems.com). For example, with knowledge of the full 3D orientation of the contact finger, we could enable tilt-based interactions such as those explored with the TiltMenu [31] for pen-based interactions. Alternatively, intuitive 3D manipulations of digital objects may be explored on interactive surfaces by using the finger as a proxy.

FUTURE WORK

Several open questions remain to be explored in the future:

First, we would like to further improve our algorithm to overcome the limitations discussed previously, for example by considering the detailed geometry of the fingers' touch. This would be led by a deeper investigation of the properties of finger contact, including less typical scenarios such as when the user touches with the side of a finger. Particular attention would be given to the thumb, which has several unique properties compared to other fingers. This is especially important as some multi-touch manipulations often involve the movement of both thumb and index finger.

Additionally, based on the inference made from the structural interrelationship of the fingers, we are interested in experimenting with extracting higher-level information, to cluster touch points into congruent hand configurations. Continuous tracking of the user's full hands may also be possible by considering the dynamics of the fingers even when they are not always touching the surface.

At the current time, most of our designs have been implemented as proof-of-concepts, while "aim and grab" and "orientation-sensitive widgets" are in design stage. In the future we will iterate on these prototypes to improve the design details. We have not yet implemented an orientation-invariant gesture/handwriting recognition engine as we proposed. We plan to develop and experimentally evaluate such an engine, and also explore its applications in other scenarios such as with handheld devices.

CONCLUSION

Our contribution in this paper is two-fold. We first presented a simple and generally applicable algorithm to unambiguously detect the directed orientation of user's fingers on interactive surfaces from contact information only. Researchers can apply this algorithm on various direct-touch surfaces to serve their own applications. We then explored user interface designs that leverage this finger orientation information, as well as further inferences that can be made from finger orientations. These designs and inferences can be useful for interaction with a variety of direct-touch devices that generate finger orientation information, either using our general algorithm or other more specialized sensing technologies. Our work shows

that finger orientation is a feasible and valuable input dimension that can be utilized for novel interactions on interactive surfaces.

ACKNOWLEDGEMENTS

We thank the members of Ren Lab in Kochi University of Technology for discussion and support, study participants, and anonymous reviewers for valuable insights.

REFERENCES

1. Albinsson, P.r.-A. and Zhai, S., High precision touch screen interaction. *CHI 2003*, 105-112.
2. Benko, H., Wilson, A.D. and Balakrishnan, R., Sphere: multi-touch interactions on a spherical display. *UIST 2008*, 77-86.
3. Benko, H., Wilson, A.D. and Baudisch, P., Precise selection techniques for multi-touch screens. *CHI 2006*, 1263-1272.
4. Blanch, R., Guiard, Y. and Beaudouin-Lafon, M., Semantic pointing: improving target acquisition with control-display ratio adaptation. *CHI 2004*, 519-526.
5. Blasko, G., Beaver, W., Kamvar, M., and Feiner, S., Workplane-orientation-sensing techniques for tablet PCs. *UIST 2004 Conf. Supplement*, 1-2.
6. Brandl, P., Seifried, T., Leitner, J., Haller, M., Doray, B. and To, P., Occlusion-Aware Menu Design for Digital Tabletops. *CHI 2009*, 3223-3228.
7. Cao, X., Wilson, A.D., Balakrishnan, R., Hinckley, K. and Hudson, S., ShapeTouch: Leveraging contact shape on interactive surfaces. *TABLETOP 2008*, 129 - 136
8. Davidson, P.L. and Han, J.Y., Extending 2D object arrangement with pressure-sensitive layering cues. *UIST 2008*, 87-90.
9. Dietz, P. and Leigh, D., DiamondTouch: a multi-user touch technology. *UIST 2001*, 219-226.
10. Esenther, A., Forlines, C., Ryall, K. and Shipman, S. Support for Multi-User, *Multi-Touch Applications*, 2002.
11. Esenther, A. and Ryall, K., Fluid DTMouse: better mouse support for touch-based interactions. *AVI 2006*, 112-115.
12. Forlines, C., Wigdor, D., Shen, C. and Balakrishnan, R., Direct-touch vs. mouse input for tabletop displays. *CHI 2007*, 647-656.
13. Grossman, T. and Balakrishnan, R., The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. *CHI 2005*, 281-290.
14. Grossman, T. and Balakrishnan, R., The design and evaluation of selection techniques for 3D volumetric displays. *UIST 2006*, 3-12.
15. Han, J.Y., Low-cost multi-touch sensing through frustrated total internal reflection. *UIST 2005*, 115-118.
16. Izadi, S., Hodges, S., Butler, A., Rrustemi, A. and Buxton, B., ThinSight: integrated optical multi-touch sensing through thin form-factor displays. *workshop on Emerging displays technologies*, 6.
17. Kabbash, P. and Buxton, W.A.S., The ``prince'' technique: Fitts' law and selection using area cursors. *CHI 1995*, 273-279.
18. Lee, S., Buxton, W. and Smith, K.C., A multi-touch three dimensional touch-sensitive tablet. *CHI 1985*, ACM, 21-25.
19. Malik, S. and Laszlo, J., Visual touchpad: a two-handed gestural input device. *Multimodal interfaces 2004*, 289-296.
20. Matejka, J., Grossman, T., Lo, J. and Fitzmaurice, G., The Design and Evaluation of Multi-Finger Mouse Emulation Techniques. *CHI 2009*, 1073-1082
21. Microsoft. Microsoft Surface, 2007, http://www.microsoft.com/surface.
22. Morris, M.J. Supporting Effective Interaction With Tabletop Groupware, *Meredith June Morris*, Stanford University, 2006.
23. Moscovich, T. Principles and Applications of Multi-touch Interaction, Brown University, 2007.
24. Moscovich, T. and Hughes, J.F., Indirect mappings of multi-touch input using one and two hands. *CHI 2008*, 1275-1284.
25. Moscovich, T. and Hughes, J.F., Multi-finger cursor techniques. *GI 2006*, 1-7.
26. Nuigroup. Touchlib, Nui Group, 2009. http://www.nuigroup.com
27. Olwal, A. and Feiner, S., Rubbing the Fisheye: precise touch-screen interaction with gestures and fisheye views. *UIST 2003 Supplyment*, 83-84.
28. Potter, R.L., Weldon, L.J. and Shneiderman, B., Improving the accuracy of touch screens: an experimental evaluation of three strategies. *CHI 1988*, 27-32.
29. Rekimoto, J., SmartSkin: an infrastructure for freehand manipulation on interactive surfaces. *CHI 2002*, 113-120.
30. Scott, S.D., Carpendale, M.S.T. and Inkpen, K.M., Territoriality in Collaborative Tabletop Workspaces. *CSCW 2004*, ACM, 294-303.
31. Tian, F., Xu, L., Wang, H., Zhang, X., Liu, Y., Setlur, V. and Dai, G., Tilt menu: using the 3D orientation information of pen devices to extend the selection capability of pen-based user interfaces. *CHI 200*8, 1371-1380
32. Vogel, D. and Baudisch, P., Shift: a technique for operating pen-based interfaces using touch. *CHI 2007*, 657-666.
33. Wang, F. and Ren, X., Empirical Evaluation for Finger Input Properties In Multi-touch Interaction. *CHI 2009*, 1063-1072
34. Wilson, A.D. PlayAnywhere: a compact interactive tabletop projection-vision system. *UIST 2005*, 83-92.
35. Wilson, A.D., TouchLight: an imaging touch screen and display for gesture-based interaction. *Multimodal interfaces 2004*, 69-76.
36. Wilson, A.D., Izadi, S., Hilliges, O., Garcia-Mendoza, A. and Kirk, D., Bringing physics to the surface. *UIST 2008*, 67-76.
37. Wu, M. and Balakrishnan, R., Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. *UIST 2003*, 193-202.