Usage and Recognition of Finger Orientation for Multi-Touch Tabletop Interaction

Chi Tai Dang, and Elisabeth André

Augsburg University, Human Centered Multimedia, Department of Computer Science,
Universitaetsstr. 6a, 86159 Augsburg, Germany
{LastName}@informatik.uni-augsburg.de

Abstract. Building on the observation that finger orientation is an inherent part of human's interaction in the real world, exploiting finger orientation for multi-touch tabletop interaction would facilitate more natural interaction techniques. We motivate this by means of examples where the finger orientation improves or enriches interaction. Afterwards, we present a simple and fast approach to detect the finger orientation reliably for multi-touch tabletop interaction. The steps involved are computationally cheap and therefore suit the needs of tracking software operating under time-critical conditions. We show that the presented approach enables the detection of finger orientation also for fingers that touch the tabletop surface only slightly. Further, recognition rates on real data gained from the camera within a multi-touch tabletop are presented in order to give a measure for the precision and reliability of the presented approach.

Keywords: Finger Orientation, Multi-Touch, Tabletop, Tracking, Interaction

1 Introduction

Over the last years, various multi-touch sensing technologies evolved and have matured out of their infancy, for example vision-based techniques [1-5] or techniques based on electrical capacitance [6-8]. Such technologies attracted a lot of attention not only from the research community, but also from the general public as well as the industry. Consequently, commercial products based on these technologies became publicly available, for example in the form of horizontal tabletops [6, 9, 10, 11], large displays [12], desktop computers [13, 14], notebooks [13, 14], or smartphones and tablet-sized devices [8]. Particularly smartphones and small multi-touch devices are now commonplace and more and more users become familiar with multi-touch interaction. Commercial horizontal tabletops are not as widespread as small mobile devices mainly because of their higher costs. However, they laid the foundation for multi-touch interaction design and still pose a rich playground for interaction research due to their experimental character and the possibilities of the bigger input space for interaction.

The novelty of those technologies is primarily attributed to interaction techniques and applications involving direct-touch with bare fingers and also in combination with physical artifacts representing digital information. Such interaction techniques are entitled to be natural since users may use their fingers to interact with digital objects in the same way as they would do with physical objects. Indeed, directly touching objects and manipulating them with multiple fingers pose a natural form of interaction that benefits from users' manual dexterity. Direct interaction with fingers allows to use a multi-touch tabletop computer straight away without learning how to use the input device. Humans use their fingers every day for any number of tasks and therefore they are used to interact with their fingers in the real world. To some extent, humans consciously make use of directions in this interaction to complement or convey their intention, for instance when pointing at objects or indicating a direction.

From a software application's point of view, the sensing technologies provide touch properties which have to be interpreted correctly by the application so as to enable the expected natural interaction. Common touch properties used therefor include finger position [6, 15], contact shape [16, 17], or contact area's size [18]. In [19], the authors empirically evaluated and discussed several finger input properties and the usage of those for multi-touch tabletop interaction. Their evaluation yielded guidelines for widget-design and several proposals for widget-designs where finger orientation, that is the direction which the finger points to, plays an important role. To our knowledge, the finger orientation has not been used for interaction in multi-touch applications as also claimed in [19]. One possible reason may be the unavailability of finger orientation in freely available tracker software, such as [15] or [20], which provide only the aforementioned common touch properties. Reliable methods to detect finger orientation unambiguously and with high precision would foster integration of finger orientation detection into tracker software.

Apparently, employing finger orientation to extend or complement natural interaction would offer valuable potential for a more natural form of interaction, since finger orientation is an inherently integrated aspect in human's interaction in the real world.

This paper is structured as follows: In the next two sections, we motivate the use of finger orientation for natural interaction by giving design examples for user interfaces and interaction techniques. Section 4 introduces into the technical background followed by a description of a naive approach and our new approach for detection of finger orientation. In Section 6, we present recognition rates for the presented approaches and discuss benefits and issues. Finally, we summarize and conclude our work in Section 7.

2 Usage of Finger Orientation

Incorporating finger orientation into natural interaction poses chances for improvements for several issues pertaining to touch user interfaces and interactions: manipulation, occlusion, selection, and adaptation. Both interaction techniques and user interfaces can be designed to account for finger orientation as exemplified follows.

Manipulation. Wang and colleagues [21] present a variety of interaction techniques, such as orientation-sensitive widgets, that would be enabled by robust techniques to recognize finger orientations. Such widgets take the finger orientation into account and enable function selection combined with parameter adjustment requiring only little space for interaction. Another example for this interaction technique could be a rotary switch that offers only two states to pin or release objects on the workspace, see Figure 1.

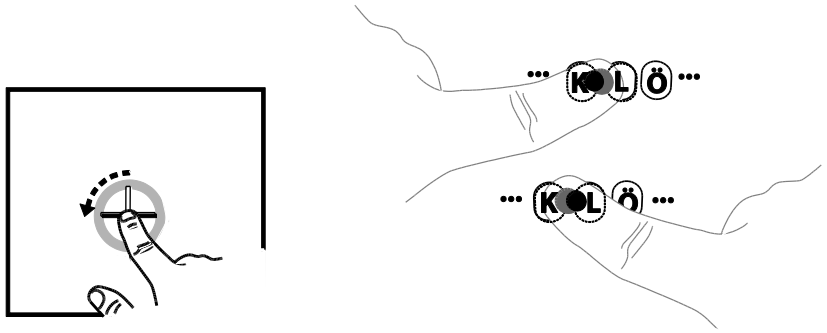

Fig. 1. Rotary switch *(left)*; Adaptation of reported input point *(right)*.

Occlusion. Information about finger orientation would allow us to determine areas that are potentially occluded by the hand and adapt the display of graphical objects accordingly. For instance, widgets could be re-oriented in such a way that they are visible for the most part. Further, techniques such as *occlusion-aware pop-ups* or *occlusion-aware dragging* presented for pen or stylus interaction in [22] could be afforded by means of the finger orientation and realized for interaction with bare fingers. Thereby, hierarchical menus or tooltips might appear in non-occluded area or occluded text segments could be shown in callouts to support selection tasks.

Selection. If the orientation of fingers was recognized, users would be able to use their fingers of either one hand or both hands for pointing at objects displayed on the surface allowing them to select distant objects in a more natural manner as depicted in Figure 2 at the left-hand side. Users could also use thumb and index finger of one hand to span an open angle for object selection. Another option opens up if users make use of both hands to span two open angles as sketched in Figure 2 at the right-hand side. The two selected areas may intersect with each other and create a third selection area which might be used for object selection.

Adaptation. Vogel and colleagues [23] show that the reported input point from a finger touch differs from the intended target location by an offset. This offset is due to the fact that users perceive an input point different than the real target location. Based on this observation, they suggest an adaptation to correct the reported input point by this offset. Here, the finger orientation could provide the direction in which the

adaptation should be applied. As an example, the interaction with a virtual keyboard on small touchscreens would benefit from such an adaptation as sketched in Figure 1. Without adaptation, the reported input point (gray filled circle) would result in ambiguous key selection whereas with an adaptation, the reported input point would better fit the perceived input point (black filled circle).

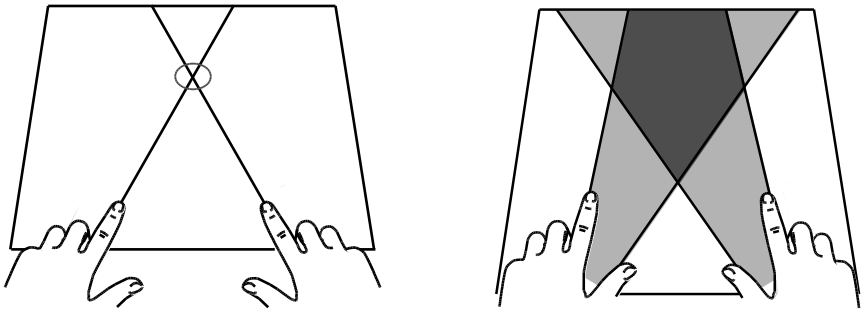

Fig. 2. Distant object selection *(left)*; Fingers spanning open angles for selection *(right)*.

From a pragmatic point of view, user interfaces and interaction techniques that integrate finger orientation would help mitigate one of the main issues in natural interaction, namely the arm fatigue issue [24, 25]. By reducing the overall hand and arm movements for manipulation or selection tasks, users' hand and arm fatigue would be diminished as well. Not only single interaction techniques can be improved but also higher-level recognition tasks, such as the distinction between one- and two-handed interactions [26], would be enabled by reliable detection of finger orientation. In [26], we present a mechanism to distinguish hands based only on the location and orientation of finger contact areas.

3 Related Work

Much work has been done on detecting finger orientation in 3D space employing multiple cameras or color images, for example [27, 28, 32], whereas only less work covers detection of finger orientation for 2D touch sensing technologies or in combination with infrared images.

Malik and colleagues [29] presented the Visual Touchpad which utilized two color cameras mounted above the touchpad to detect user's hands and fingers. They identified fingers' positions through computer vision methods to find the fingertips on a hand contour. The hand contour is also used to determine the finger orientation for each fingertip. Their approach is based on color images and a direct view on the

hands which is quite different from prevalent multi-touch sensing technologies employing infrared images and a bottom view on the sensing surface.

Wang et al. [21] proposed an approach to unambiguously determine the finger orientation that is based on the contact areas produced by finger touches. They fit an ellipse into the contact shape and use the longer ellipse axis for determination of the finger orientation. Moreover, they observe the center point's variation of the contact areas when a finger lands on the surface to disambiguate the finger orientation. Their approach is suitable for sensing technologies that provide only finger contact areas, whereas sensing technologies such as Diffused Illumination offer more potential for detection of finger orientation with a higher precision. In their work, they also show that finger orientation is a useful input property that can be employed to enhance user interactions.

A rather simple and inexpensive way to integrate finger orientation in multi-touch tabletop interaction was conducted by Marquardt et al. [30]. They employed the Microsoft Surface table [9] and a glove which was tagged with several fiduciary markers. The tabletop system was able to detect the markers together with their orientation, thus allowed them to derive finger orientation and to identify individual parts of the hand and their orientations. Wearing gloves is contrary to natural interaction, but Marquardt's approach allows for rapid prototyping of interaction techniques and they indicated that integrating more properties from fingers and hands provide rich opportunities for interaction design.

We pointed out that finger orientation poses meaningful chances to extend or complement natural interaction. In order to be used for user interaction, finger orientation must be determined reliably and with high precision. For this purpose, we present and discuss a simple approach that reliably detects the finger orientation for tabletop setups which employ sensing techniques similar to Diffused Illumination. This approach even detects the finger orientation for difficult cases, where the finger touches the surface only slightly.

4 Diffused Illumination

A lot of multi-touch sensor technologies employing infrared light emerged during recent years [1, 2, 3, 5, 33, 34]. In this paper, we refer to the Diffused Illumination approach which operates on raw images that are comparable with those created by technologies such as DSI or LLP [5]. Diffused Illumination setups have infrared sources mounted in the interior of the table which emit infrared light towards the tabletop surface. Objects such as fingers or physical artifacts on or above the tabletop surface reflect the infrared light back into the table. This reflected infrared light exposes a camera sensor inside the table which delivers images that show the illuminated objects. Therefore, when users' fingers approach the tabletop surface, the captured images show the whole hand with its fingers where the brightness of the hands' and fingers' pixels indicate the closeness to the tabletop surface, for example Figure 3 on the left-hand side. Hence, they offer rich possibilities for object detection through computer-vision methods that take the pixel values into account.

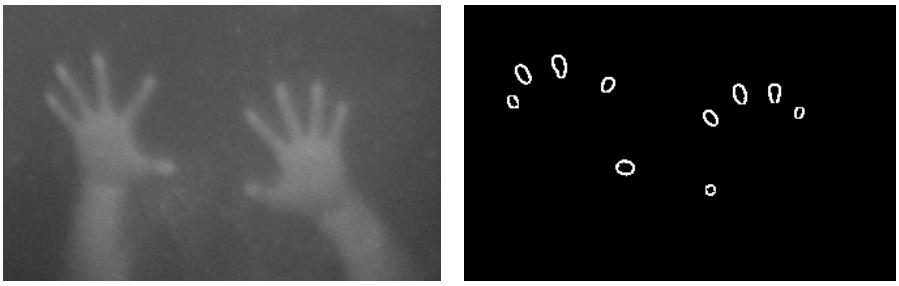

Fig. 3. Infrared image *(left)* and corresponding contour image *(right)*.

The contact area's pixels of a finger that touches the surface are brighter than the pixels of non-contact areas. Hence, a typical process chain to find finger contact areas and to locate the fingers' coordinates exploits this fact and utilizes a brightness threshold to distinguish contact area from non-contact area. The steps comprise of converting the raw camera image into a blob image based on the brightness threshold and afterwards converting the blob image into a contour image. Within the resulting contour image, each contour represents a contact area of one finger that touches the tabletop surface as shown in Figure 3 on the right-hand side. Finally, the coordinates of each finger contact can be determined as the corresponding contour's center location.

5 Finger Orientation Detection

In this section, we will first outline a straightforward way to determine finger orientation which serves as a baseline for the performance comparison in Section 6.3. The remainder of this section starting from Section 5.2 illustrates our new approach in detail in order to ease integration in tracker software.

5.1 A Naive Approach

A naive approach to determine finger orientation bases on the aforementioned contour image. The steps therefor can be accomplished easily with high-level functions that are part of the computer vision package OpenCV [31]. We will further use the term *ellipse method* to denote the approach described in the following.

When considering the contour image in Figure 3, we can spot each finger contour as an ellipse representing the finger contact area. Apparently, such a matter of fact enables to fit an ellipse into each detected contour and take the angles between x-axis and the corresponding longer ellipse axis to determine finger orientations as exemplarily sketched in Figure 4 for only one contour.

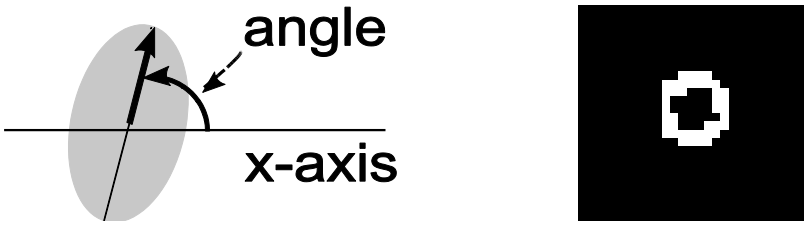

Fig. 4. Angle to *x-axis (left)*. A circular contour *(right)*.

However, this approach suffers from inherent weaknesses since it relies only on the contour of a finger contact. For instance, it produces ambiguous results for circular contours because the axes in a circle can point towards any direction. Furthermore, a detected finger orientation could be wrongly skewed by 180° since the longer axis of an ellipse possesses two possible directions.

180° Adjustment. With the aid of the raw camera image, we are able to resolve the finger orientation in case it is wrongly skewed by 180°. We detect this kind of ambiguity by walking the longer ellipse axis into both directions while looking for the direction that shows up a non-finger pixel at first. The thereby detected direction is where the fingertip ends, thus we adjust the afore-detected finger orientation if it is wrongly skewed. Henceforth, we will use the term *ellipse method + 180-adjust* to denote this addition to the *ellipse method*.

Discussion. *Ellipse method* and *ellipse method + 180-adjust* represent a straight forward way, but there are common cases in which finger interaction produces problematic contours for both methods. For instance, let us consider a child with small fingers touching the surface or finger touches that stem from users touching the surface only slightly. In these cases, the detected contour is very small and features a circular shape in the worst case which leads to imprecise or high deviant finger orientations as exemplarily depicted in Figure 4. This is due to the fact that for small contours, the fewer pixels that contribute to the contour, the more effect one pixel has on the detected finger orientation. This is even more worse if we consider that camera noise always randomly affect pixels of the contour which results in jumping values for the detected finger orientation.

To overcome these issues and furthermore increase precision and stability of the detected finger orientation, we make use of the difference in brightness of proximate pixels kept in the raw image. We draw on that information to determine the finger contour and derive the finger orientation from only a part of the finger contour.

5.2 A Simple and Precise Detection Algorithm

When considering images produced by Diffused Illumination setups, we can identify each finger with its outer contour as depicted in Figure 5. What each finger contour

has in common are two quasi axially symmetrical lines that converge circularly at the fingertip.

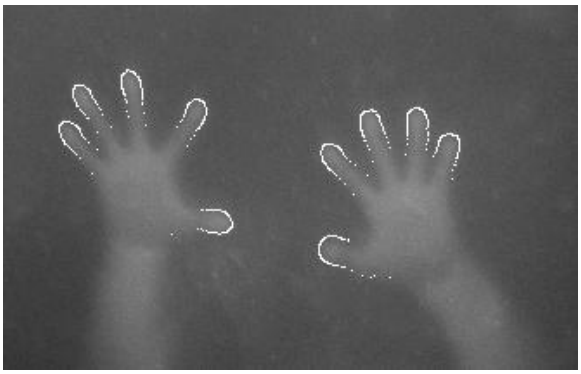

Fig. 5. Finger contours marked with *white lines*.

Our approach exploits these symmetrical running lines to calculate the finger orientation. For each finger, we are able to detect these two lines and they always point into the direction of the finger orientation, thus the thereby identified finger orientation is reliable. Our algorithm comprises of three essential steps, each consecutively applied in the given order to each detected finger position. The detection of finger position was described in Section 4.

1. Detect the finger contour.
2. Determine the symmetrical lines.
3. Determine the angles in which the tracks point to and average them.

1. Detection of Finger Contour. As a first step, the outer finger contour for a given finger position has to be identified. This task is decisive for the remaining steps because the points of the contour intrinsically contribute to the precision of the finger orientation to be determined. Initially, we span a circle with the radius R pixels around the finger contact position and distribute points on that circle at an interval of 5° as depicted in Figure 6. We have chosen $R = 40$ pixels for our images in order to cover at least two times the finger width of small people's fingers and one time and a half the finger width of people with chubby fingers. This value must be adjusted for other tabletop setups depending on the resolution of the raw images and the projection size of the tabletop. For higher resolution cameras, the amount of points on the circle might be increased for a higher precision of detection.

The next step is to perform a search for a contour pixel starting from the finger contact's center position to each point that was distributed on the circle. To be precise, we process 72 paths at an interval of 5°. Within each search run, we compare the pixel

value of each point on the path with the pixel value of the center position. Once the difference of their values exceeds a certain threshold, the search terminates and the pixel coordinate on the path is noted in a list.

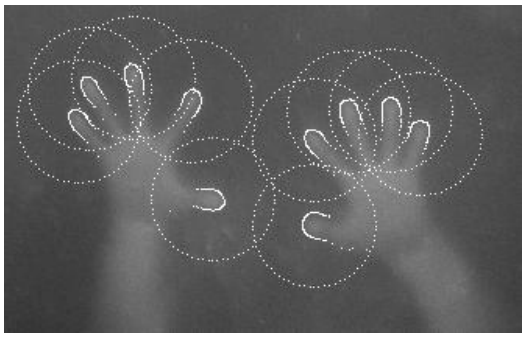

Fig. 6. *Points on circles* spanned around the finger contacts.

For our implementation, we used the value 18 for the threshold. This threshold may vary for other DI settings depending on the brightness and contrast of the captured images. Here, adaptive threshold calibration based on a histogram of the image could compensate for environments with varying light conditions. The result of this step is a list which denotes whether a contour point was found or not for each of the 72 points on the circle. In particular, the list has the following properties:

Definition 1. *Properties of contour-point list*

1. *The ordered entries enumerated from 1 to 72 correspond to the points on the circle from 0° to 355° at an interval of 5°.*
2. *Each entry contains the coordinate (x, y) of a contour pixel.*
3. *If no contour pixel was found, then the entry contains the value (-1, -1).*

This procedure guarantees that in theory the finger contour found contains only the contour of the finger we consider and not a contour point of an adjacent finger. Since we start the search from the finger contact's center position, the first pixel that terminates the search must belong to the same finger. Furthermore, most of the searches terminate quickly because the distances between the finger contour and the finger contact's center position are short, see Figure 6.

2. Determine Symmetrical Lines. The following step operates only on the list defined in the previous section and determines the two quasi-symmetrical lines that contribute to the finger orientation.

In what follows, we treat the list as a ringbuffer where the subsequent entry of entry 72 points at entry 1 and vice versa. Furthermore, the defined names in italic type denote indexes into the list corresponding to the usage in Figure 7 unless otherwise noted. The name *list* denotes the afore defined list of points, where a single point can be retrieved by means of squared brackets as used in the programming language C.

At first, we perform a search for the biggest range that only consists of values of *(-1, -1)*. This range is defined as the gate g_{start}, ..., g_{end} and represents the part where the finger is connected to the hand as depicted in Figure 7. The complementary range $c_{start} = g_{end} + 1$, ..., $c_{end} = g_{start} - 1$ is defined as the finger contour.

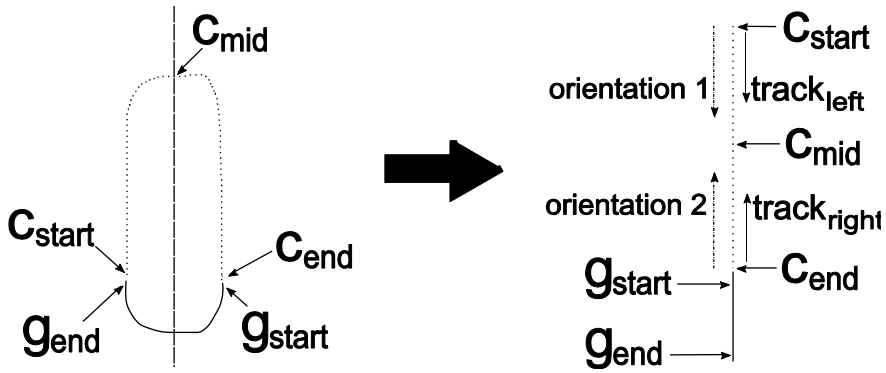

Fig. 7. Finger contour with start and end points *(left)*. List of contour points and its allocation *(right)*.

When considering the finger contour as two parts with c_{mid}: $c_{start} < c_{mid} < c_{end}$, there are two contour tracks that have to be cut back in order to remove the part representing the fingertip. The fingertip part can only be used for detection of finger orientation if the contour tracks are absolutely symmetrical to each other. This cannot be guaranteed due to different finger or hand pose which may destroy symmetry of the fingertip contour. Therefore, the fingertip part of the contour has to be removed as far as possible. We empirically determined that considering only 60% of the points in each contour track suffices to remove the fingertip points. That is the amount of points to include in each contour track is defined as $amount_c = 0.6 * (c_{end} - c_{start}) / 2$. This can be used independent of image resolution because higher resolution would result in more contour pixels, but the relationship between finger and fingertip remains the same. Considering only 60% of the finger contour is a tradeoff between including the contour that contributes to the finger orientation and omitting the fingertip contour. For the following steps, we further define the left contour track as $track_{left} = c_{start}, ..., c_{mid} - amount_c$ and the right contour track as $track_{right} = c_{end}, ..., c_{mid} + amount_c$, see Figure 7.

We have to catch a rare case in this step: if the list does not contain a gate range, then we have an elliptical finger contour as exemplified in Figure 8.

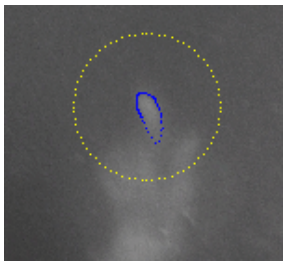

Fig. 8. Elliptical finger contour caused by low contrast.

This case happens rarely and is caused by low contrast of the raw image which leads to falsely detected contour points stored in the list. In such a case, we skip the next step and proceed with the *ellipse method + 180-adjust* to detect finger orientation, which was described in Section 5.1.

3. Determine Average Angle. Both afore ascertained contour tracks of a finger contact contribute to the finger orientation based on the properties of the list defined in Definition 1. The properties imply that c_{start} is the starting point of the left contour track and c_{end} is the starting point of the right contour track. Because of that, we also know that the tracks $track_{left}$ and $track_{right}$ point into the direction where the finger is pointing to as well. Therefore, the last step serves to calculate the angles to the x-axis for $track_{left}$ and $track_{right}$ and average them afterwards to determine the final finger orientation.

Here, two methods with differing complexity may be applied. A computationally complex method bases on linear regression, whereby the points of a contour track are considered as a point cloud. A linear regression model is then fitted on this point cloud using least squares and the resulting slope of the regression line contributes to the finger orientation. This method is applicable for time-critical conditions if sufficient processing power is available.

A much simpler and faster approach that requires linear time draws on the slopes for each point of a contour track. For this approach, the coordinate denoted by the index c_{start} or c_{end} is defined as the anchor point respectively depending on the contour track to work on. That is either $anchor_x = list[c_{start}]_x$, $anchor_y = list[c_{start}]_y$ if we consider $track_{left}$ and otherwise $anchor_x = list[c_{end}]_x$, $anchor_y = list[c_{end}]_y$. The following instructions have to be applied on $track_{left}$ and $track_{right}$ independently. At first, the differences in x-value and y-value from the anchor point to all other contour track points are summed up and the resulting summed x-values and y-values are considered as the slope for the line running through the point cloud for the contour track. This is formally specified with equations 1, 2 and 3 where t_{start} and t_{end} are the

indexes into the contour point list of the corresponding contour track. Finally, both slopes have to be averaged to determine the finger orientation.

$$sum_{\Delta x} = \sum_{i=t_{start}}^{t_{end}} list[i]_x - anchor_x \quad (1)$$

$$sum_{\Delta y} = \sum_{i=t_{start}}^{t_{end}} list[i]_y - anchor_y \quad (2)$$

$$slope = \frac{sum_{\Delta y}}{sum_{\Delta x}} \quad (3)$$

The simpler approach admittedly is prone to errors due to wrong initialization of the anchor point. However, the results we present in Section 6.3 show that this simple method still produces recognition rates that are quite close to those of the linear regression method.

6 Recognition Rates

In order to obtain recognition rates that have relevance for real data and that would occur in real tabletop interaction, we have captured a raw videostream from an infrared camera that was mounted in the table directed towards the surface. The video shows hands and fingers from a user who touches the surface the same way as he would do to interact with an application. Thereby, he uses combinations of one hand and two hands and with different finger combinations multiple times.

Fig. 9. From left to right: *(1)* move object with index finger, *(2)* move object with multiple fingers, *(3)* grasping with all fingers.

For example, the finger combinations included combinations used to move objects with the index finger (Figure 9, Nr. 1) or multiple fingers (Figure 9, Nr. 2), grasping with all fingers (Figure 9, Nr. 3) or zooming with fingers of one hand (Figure 10, Nr. 1) or both hands (Figure 10, Nr. 2). As a result of this, the video contains multiple repeats of parallel landing and lifting fingers of multiple adjacent fingers. While the camera captured the video, the tabletop showed a black screen and gave no visual feedback to the finger contacts. Furthermore, the fingers in some of the captured images were blurry due to quick continuous hand and finger movements. Hence, the video shows images that occur in multi-touch tabletop setups for realistic continuous interaction situations. In addition to the video, we created reference finger orientations for each frame of the video that were used to evaluate the accuracy of the detected finger orientations.

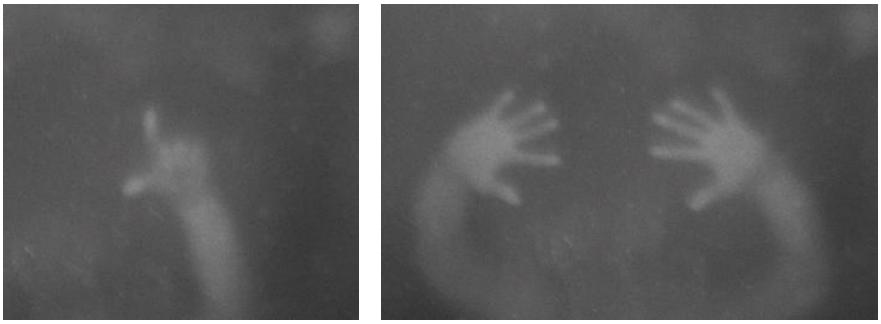

Fig. 10. From left to right: *(1)* zoom with fingers of one hand, *(2)* zoom with fingers of both hands.

We chose a different method to test the performance of our approach than Wang did in [21] because of two reasons. First, Wang evaluated his finger orientation approach with an FTIR table which poses higher contrast and less noise thus creates lower challenges than DI tables. Second, he evaluated his approach by investigating to what extent the system's response was in line with the user's objectives when conducting a set of pre-defined tasks with the index finger.

In the present work, we propose a different evaluation approach which compares the finger orientation found by our algorithm with the finger orientation perceived and determined by three independent human judges. The advantage of our method is that it enables a task-independent evaluation and gives a measure for the precision that is more focused on continuous interaction.

6.1 Reference

The video for the reference finger orientations was available in an uncompressed format to preserve the raw data and comprises of 749 frames captured at 30 fps with a resolution of 640x480 pixels for each frame. The resolution covers a physical surface

space of 60x80 cm. For creating the reference, each frame was converted to a blob image and the center position of each recognized blob was stored with its frame number in an XML-file. In all, 2007 finger contacts were detected. To ease the annotation of finger orientations and to generate the reference automatically, we have developed a graphical tool with the following functionality. Each frame along with the position of the detected finger contacts was presented to the annotator and the annotator had to manually adjust the finger orientations, which were detected through the *ellipse method + 180-adjust* approach. Overall, three persons annotated 6021 finger orientations and hence for each finger contact three finger orientations were created. The annotators were instructed to repeat an adjustment as often as required if they were not absolutely sure about the correctness of the visually perceived finger orientation and their adjusted finger orientation. The data gained through the annotations were used to build the reference finger orientations by calculating an average finger orientation for each finger contact.

6.2 Accuracy of the Reference Orientations

The reason why we averaged annotations procured from three different persons is that different persons may determine slightly different finger orientations due to camera noise, image contrast and image resolution. Our tool enabled the annotator to adjust the finger orientation angle with a precision of two decimal points as depicted in Figure 11.

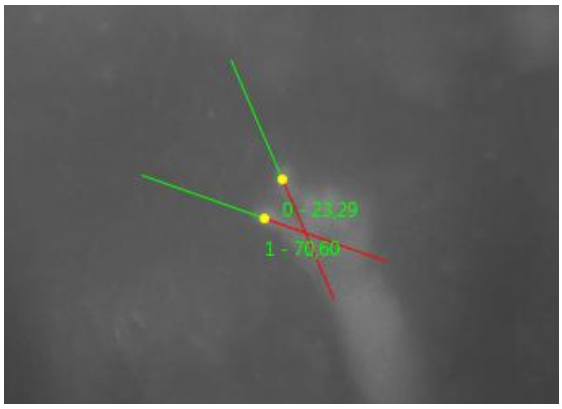

Fig. 11. Annotated index-finger and thumb.

Figure 11 also shows that the raw images are blurry and with low contrast. This makes it difficult to pinpoint an absolute finger orientation. Therefore, the annotation tool supplied a colored visual line alike to a ruler which assisted the annotator in gauging a proper finger orientation by choosing the mid of the acceptable angle ranges. This process enabled us to procure data from multiple annotators and take the

averages for the reference. Overall, the reference finger orientations had a standard deviation of 3.34°.

6.3 Results

The precision of our approach is illustrated by comparing its recognition rates with the recognition rates of the *ellipse method* and the *ellipse method + 180-adjust* that are explained in Section 5.1. As for our approach, we implemented both variants described in Section 5.2 to find a line running through the contour track. The term *Contourtrack* denotes our approach within this section. Table 1 shows the recognition rates for all four approaches and their standard deviation from the finger orientation in the reference. The table lists the recognition rates for four cases: 5°, 10°, 15°, and 25°. The recognition rate in the 5° column shows the percentage of recognized finger orientations at which the difference to the reference is less than 5°. The same interpretation applies to the 10°, 15°, and 25° column.

Table 1. Recognition rates and standard deviation for four methods: 1 = *ellipse method*, 2 = *ellipse method + 180-adjust*, 3 = *Contourtrack* + simple slope, 4 = *Contourtrack* + regression line.

Method	< 5°	< 10°	< 15°	< 25°	*SD*
1	41.5%	60.49%	68.81%	74.39%	86.46°
2	49.58%	74.34%	84.65%	92.53%	23.19°
3	75.29%	93.02%	96.86%	99.3%	6.17°
4	75.24%	94.87%	97.81%	98.9%	6.43°

The naive *ellipse method* shows low recognition rates (60.49% at 10°-error) and a high standard deviation of 86.46° which mainly stems from the wrongly 180° twisted finger orientations. This method hugely benefits from correcting the wrongly 180° twisted finger orientations (*ellipse method + 180-adjust*) as shown in the second row of Table 1. The extension improved the recognition rates (74.34% at 10°-error) and reduced the standard deviation significantly to 23.19°. However, the recognition rates of the naive methods *ellipse method* and *ellipse method + 180-adjust* show that their precision is not reliable enough to be used in realistic applications.

In the third row and the fourth row, the recognition rates of our new approach are shown, which are much better than the naive ellipse methods. Both methods' values show good results which exceeds 93% for a maximum error of 10°. If we accept a maximum error of 25°, then both methods provide recognition rates over 98%. Both standard deviations are less than or equal to 6.43° which is only 3.09° worse than the standard deviation in the human reference data with 3.34°.

Discussion. The precision of our approach increases with higher image resolution and image contrast. The more pixels that can be used for a finger contour, the better the precision. The better the image contrast, the better a finger contour can be detected. Preprocessing steps to reduce image noise or to improve image contrast would

improve the performance, but we have not included such steps so as to keep the approach simple and fast. This way, the approach can be used as a lightweight add-on to already established tracker software. The deviation of our detected finger orientations from the reference orientations was mainly attributed to insufficient contrast of the raw image. Figure 12 illustrates two cases where the finger contour could not be determined correctly. In these cases, the difference between a finger pixel value and a background pixel value was too small, thus finger and background was not distinguishable with the threshold in use.

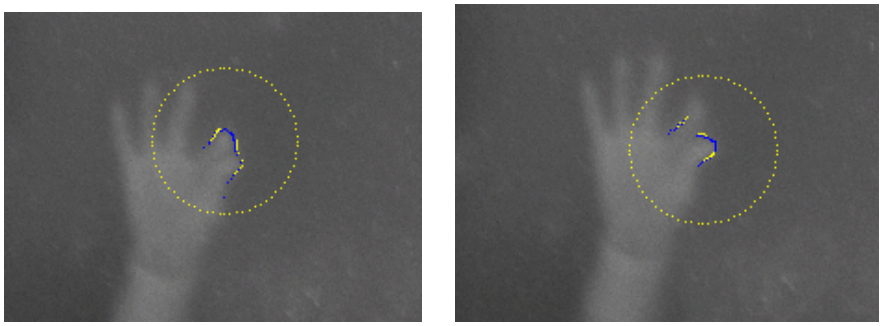

Fig. 12. Error cases with wrongly detected finger contour.

Here, increasing the image resolution and image contrast improves the recognition rates of our approach. Nevertheless, the empirical data clearly shows that the presented approach detects finger orientation reliably and with high precision even with low-resolution images (1cm^2 covered by less than 64 pixels) and low contrast. Our implementation was evaluated on a PC with Core2Duo (2.83GHz) running at 30fps where it added only marginal latency (~0.19ms per finger contact for *Contourtrack* + regression line). Hence, latency does not have a negative impact on the perceived speed of user interaction.

In Wang's approach [21], few limitations exist which are handled correctly by our approach. First, their algorithm usually detects a wrong finger orientation if users touch the surface with the side face of the thumb due to the center displacement of the contact area that is different from the other fingers. Second, if users perform a "sliding down" gesture while touching the surface, the center displacement is again different from the center displacement of an index finger's landing process resulting in wrong detection of finger orientation. Our approach is based on the finger contour which shows the correct finger orientation in such cases. The Microsoft Surface [9] tabletop also supports detection of finger orientation and would have been a suitable candidate for performance comparisons. Unfortunately, it lacks of application support for providing unprocessed raw images from each single camera, which is required for a meaningful performance comparison. Therefore, we have chosen to utilize a self-made multi-touch table employing the Diffused Illumination technique.

7 Conclusions

Our work enriches natural interaction by considering finger orientation for user interface design as well as for interaction techniques. As we have shown, both can be extended in a meaningful sense regarding manipulation, occlusion, selection, adaptation and also for high-level tasks. Natural interaction will benefit from human's inherent understanding of direction, in particular finger orientation, that they practice every day. A basic requirement to enable further research in this topic is that the detection of finger orientation has to be reliable, precise and stable for continuous interaction. Therefor, our presented approach (*Contourtrack*) to detect finger orientation has proven to be accurate enough with recognition rates over 93%. Because of the approach's low algorithmic complexity, it suits the needs of time-critical processing thus foster availability of finger orientation in tracker software.

Acknowledgments. The work described in this paper is partially funded by the EU under research grant eCUTE (Reference: 257666).

References

1. Takeoka, Y., Miyaki, T., Rekimoto, J.: Z-touch: an infrastructure for 3d gesture interaction in the proximity of tabletop surfaces. In: Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces. ITS '10, pp. 91—94. ACM, New York (2010)
2. Han, J.Y.: Low-cost multi-touch sensing through frustrated total internal reflection. In: UIST '05: Proceedings of the ACM symposium on User interface software and technology, pp. 115—118. ACM, New York (2005)
3. Hofer, R., Naeff, D., Kunz, A.: Flatir: Ftir multi-touch detection on a discrete distributed sensor array. In: Proceedings of the 3rd International Conference on Tangible and Embedded Interaction, pp. 317—322. ACM, New York (2009)
4. Peltonen, P., Kurvinen, E., Salovaara, A., Jacucci, G., Ilmonen, T., Evans, J., Oulasvirta, A., Saarikko, P.: It's mine, don't touch!: interactions at a large multi-touch display in a city centre. CHI '08, pp. 1285—1294. ACM, New York (2008)
5. NUIGroup: Multitouch techniques, http://nuigroup.com/forums/viewthread/1982
6. Dietz, P., Leigh, D.: Diamondtouch: a multi-user touch technology. In: UIST '01: Proceedings of the ACM symposium on User interface software and technology, pp. 219—226. ACM, New York (2001)
7. Rekimoto, J.: Smartskin: an infrastructure for freehand manipulation on interactive surfaces. CHI '02, pp. 113-120. ACM, New York (2002)
8. Apple inc., iphone, ipad, magic trackpad, http://www.apple.com
9. Microsoft surface, http://www.surface.com
10. Smart table, Smarttechnologies, http://www.smarttech.com
11. Archimedes solutions, http://www.multi-touch.de
12. Evoluce, Multitouch lcd monitors, http://www.evoluce.com
13. Dell, Latitude xt2, dell studio one 19, http://www.dell.com
14. HP Touchsmart tx2, touchsmart 600-1050de, http://www.hp.com
15. NUIGroup: Community core vision, http://ccv.nuigroup.com
16. Moscovich, T.: Contact area interaction with sliding widgets. In: UIST '09: Proceedings of the 22nd annual ACM symposium on User interface software and technology, pp. 13-22. ACM, New York (2009)

17. Cao, X., Wilson, A.D., Balakrishnan, R., Hinckley, K., Hudson, S.E.: Shapetouch: Leveraging contact shape on interactive surfaces. In: Tabletop 2008, pp. 129--136. (2008)
18. Benko, H., Wilson, A.D., Baudisch, P.: Precise selection techniques for multi-touch screens. In: CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems, pp. 1263—1272. ACM, New York (2006)
19. Wang, F., Ren, X.: Empirical evaluation for finger input properties in multi-touch interaction. In: CHI '09: Proceedings of the 27th international conference on Human factors in computing systems, pp. 1063—1072. ACM, New York (2009)
20. NUIGroup: Touchlib, a multi-touch development kit, http://www.nuigroup.com/touchlib
21. Wang, F., Cao, X., Ren, X., Irani, P.: Detecting and leveraging finger orientation for interaction with direct-touch surfaces. In: UIST '09: Proceedings of the 22nd annual ACM symposium on User interface software and technology, pp. 23—32. ACM (2009)
22. Vogel, D., Balakrishnan, R.: Occlusion-aware interfaces. In: Proceedings of the 28th international conference on Human factors in computing systems. CHI '10, pp. 263—272. ACM, New York (2010)
23. Vogel, D., Baudisch, P.: Shift: a technique for operating pen-based interfaces using touch. In: Proceedings of the SIGCHI conference on Human factors in computing systems. CHI '07, pp. 657—666. ACM, New York (2007)
24. Wigdor, D., Perm, G., Ryall, K., Esenther, A., Shen, C.: Living with a tabletop: Analysis and observations of long term office use of a multi-touch table. In: Horizontal Interactive Human-Computer Systems, 2007. TABLETOP'07. Second Annual IEEE International Workshop on. (2007), pp. 60 --67
25. Yee, W.: Potential limitations of multi-touch gesture vocabulary: Differentiation, adoption, fatigue. In: Proceedings of the 13th International Conference on Human-Computer Interaction. Part II: Novel Interaction Methods and Techniques, pp. 291—300. Berlin, Heidelberg, Springer-Verlag (2009)
26. Dang, C.T., Straub, M., André, E.: Hand distinction for multi-touch tabletop interaction. In: ITS '09: Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces, pp. 101—108. ACM, New York (2009)
27. Jennings, C.: Robust finger tracking with multiple cameras. In: Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems, 1999. Proceedings. International Workshop on. (1999)
28. Hung, Y.P., Yang, Y.S., Chen, Y.S., Hsieh, I.B., Fuh, C.S.: Free-hand pointer by use of an active stereo vision system. In: Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on. Volume 2. (1998), pp. 1244 --1246 vol.2
29. Malik, S., Laszlo, J.: Visual touchpad: a two-handed gestural input device. In: ICMI '04: Proceedings of the 6th international conference on Multimodal interfaces, pp. 289—296. ACM (2004)
30. Marquardt, N., Kiemer, J., Greenberg, S.: What caused that touch? expressive interaction with a surface through fiduciary-tagged gloves. In: Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces. ITS '10, pp. 139-142. ACM, NY (2010)
31. OpenCV, open computer vision library, http://sourceforge.net/projects/opencvlibrary
32. Tianding, C.: A solution of computer vision based real-time hand pointing recognition. In: Control Conference, 2008. CCC 2008. 27th Chinese. (2008), pp. 384--388
33. Jackson, D., Bartindale, T., Olivier, P.: Fiberboard: compact multi-touch display using channeled light. In: Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces, pp. 25—28. ACM, New York (2009)
34. Izadi, S., Hodges, S., Butler, A., Rrustemi, A., Buxton, B.: Thinsight: integrated optical multi-touch sensing through thin form-factor displays. In: Proceedings of the 2007 workshop on Emerging displays technologies: images and beyond: the future of displays and interacton. EDT'07, ACM, New York (2007)