Finger and Hand Detection for Multi-Touch Interfaces Based on Maximally Stable Extremal Regions

Philipp Ewerling **Alexander Kulik** **Bernd Froehlich**
Bauhaus-Universität Weimar
Bauhausstrasse 11, 99423 Weimar, Germany
philipp.ewerling@gmail.com, kulik@uni-weimar.de, bernd.froehlich@uni-weimar.de

ABSTRACT
We propose a new approach for touch detection on optical multi-touch devices that exploits the fact that the camera images reveal not only the actual touch points, but also objects above the screen such as the hand or arm of a user. Our touch processing relies on the Maximally Stable Extremal Regions algorithm for finding the users' fingertips in the camera image. The hierarchical structure of the generated extremal regions serves as a starting point for agglomerative clustering of the fingertips into hands. Furthermore, we suggest a heuristic supporting the identification of individual fingers as well as the distinction between left hands and right hands if all five fingers of a hand are in contact with the touch surface.

Our evaluation confirmed that the system is robust against detection errors resulting from non-uniform illumination and reliably assigns touch points to individual hands based on the implicitly tracked context information. The efficient multi-threaded implementation handles two-handed input from multiple users in real-time.

Author Keywords
Multi-touch; hand detection; finger detection.

ACM Classification Keywords
H5.2 Information interfaces and presentation: User Interfaces;

INTRODUCTION
Research and development on multi-touch interfaces has come up with an ever growing number of multi-touch input gestures. However, the expressiveness of touch input remains limited without additional context information. Most existing hardware platforms assume that all simultaneous touches belong to the same gesture, hence they support only a single user. This may be sufficient for small form factor devices like smartphones. Large touch screens, on the other hand, can accommodate more than one user and thus require hand detection at the very least in order to minimize interferences. It is therefore prudent that input processing should take into account the fact that input may be induced by different hands or even by different users. If we could additionally distinguish individual fingers, the design space of new multi-touch interaction techniques would be greatly enhanced.

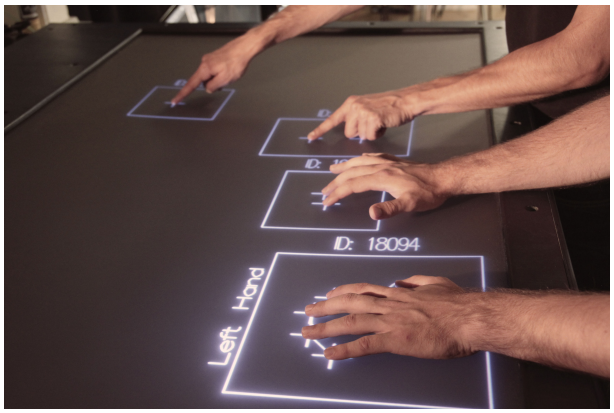


Figure 1. Detection of multiple hands and fingers. Note that the system is robust against false positives from contact with the palm of the hand.

We developed a processing pipeline for multi-touch detection on large touch screens that detects finger touches, groups finger touches into hands and distinguishes left from right hands for full-hand input (Figure 1). Our pipeline is based on the *Maximally Stable Extremal Regions* (MSER) algorithm which has previously been used primarily in stereo image matching and visual tracking. We use this algorithm to compute a hierarchy of extremal regions, which in conjunction with a set of local descriptors forms the basis for fingertip detection. Fingers are assigned to hands using agglomerative hierarchical clustering of a small subset of touch points provided by the MSER component tree. Our processing pipeline even performs hand and finger registration for situations in which all five fingers of a hand are detected.

Our approach is motivated by the observation that the captured image data of optical multi-touch systems clearly conveys the outlines of a hand even if the hand is only hovering above the surface while one or more fingers are touching the screen. Surprisingly little research exists on how to exploit this additional context information. Most existing systems perform simple contact point detection based on image filtering and thresholding that ignores any further image information.

Our main contribution is a new approach for multi-touch processing based on an adapted version of the well-established MSER algorithm in combination with on-the-fly clustering of fingertips into hands. We evaluated our prototypical implementation with four simultaneously interacting users. The evaluation confirmed the excellent detection of touch points

with over 97% true positives and the reliable assignment of touch points to individual hands with over 92% true positives while false positives fall in the range of 1% for both cases. Initial observations also indicate that our heuristic for finger and hand registration performs well. However, it is limited to situations in which all five fingers of one hand are in contact with the touch surface in a natural hand pose.

Our efficient multi-threaded implementation focuses on image regions containing potential touch points and processes up to 40 fingertips in less than 10 ms with a total latency of less than 50 ms on our hardware. While we have implemented our prototype on a system based on diffuse illumination, the processing pipeline can also be used with other touch sensing systems that provide some sort of depth map of the region above the touch screen. Examples include Thinsight[9], the current Microsoft Surface tabletop device and systems based on depth-sensing cameras.

RELATED WORK

Diffuse back-illumination is arguably the most widely used approach to multi-touch finger tracking. Nevertheless, surprisingly little literature can be found on the actual image processing. The processing pipeline, described in the following, is considered the default approach which has been extracted from prototypes described in the literature and techniques used in the open-source multi-touch community, namely the Community Core Vision framework[1].

In most setups based on infrared illumination, the detection accuracy can be negatively affected by ambient infrared light such as stray sun light. The use of high-frequency modulated light can eliminate this problem almost completely [17]. However, this is not always possible and thus the first step generally is to remove interfering noise using a *background subtraction* and *image normalization*. In certain cases the camera image is then filtered such that only objects of a certain size, typically that of a fingertip, remain. To this end, band-pass filters are applied (usually *Mid-Gauss* or *Mid-Box*). Given the assumption that in diffuse illumination setups objects appear brighter the closer they are to the surface, contact points can now be detected by a simple image thresholding.

The benefits of exploiting additional information regarding the association of touch points to individual hands and users for more expressive multi-touch interaction have been shown earlier [4, 25, 18, 19, 22, 15, 13]. Most of this research has been performed using the DiamondTouch system [4]. This commercially available multi-touch device achieves the association of touch input to respective users by coupling electric signals from the tabletop through the user's body to a distinct receiver for each user. Unfortunately, the system limits the choice of display components only to front projection and requires the users to remain in physical contact with their receiver unit. For the purpose of grasp- and posture-recognition with capacitive touch-sensing devices, Sato et al. recently proposed a novel technology called Swept Frequency Capacitive Sensing [24].

Optical systems for extracting additional context information have also been proposed. For instance, [3] and [28] use the orientation of the contact area's bounding ellipse to infer the relationship between hands and fingers, which requires users to touch the surface with the finger pad instead of the tip. Hilliges et al. [7] proposed interleaving common touchscreen interaction based on diffuse illumination with image acquisition beyond the surface. Their setup requires the projection surface to be a high frequency switchable diffuser. Other approaches utilize an overhead camera for context tracking [5, 14, 12, 8, 15]. Echter at al. proposed an additional ceiling mounted light source to enable shadow tracking of hands hovering above the input surface [6]. Both approaches require additional hardware which thwarts the compactness of the setup while also causing significant additional processing overhead to the pipeline [5].

It has been suggested to use a depth sensing camera as an all-in-one sensor both for touch detection with defined interaction surfaces and context awareness [29, 11]. Since the same sensor that is exploited for touch sensing covers the entire surrounding area in depth, touch data could directly be associated with tracked users. A drawback of this concept is that the relatively low resolution of current depth cameras impede accurate multi-touch input.

Roth et al. propose attaching small IR-emitting devices to the users' hands for cryptographically sound user identification [23]. This is obviously the most secure implementation of user tracking for tabletop interaction, but requires the users to wear an electronic device, which interferes with the walk-up-and-use paradigm of most tabletop applications.

Proximity sensors embedded to the frame of the tabletop device may also be used for integrated context tracking at tabletop interaction devices [26, 1]. This approach provides rough user tracking and even hand distinction for many purposes. Based on an heuristic, touch points can be assigned to the individual hands of several users.

With the objective of registering hands and fingertips in optical multi-touch interfaces, Walther et al. recently proposed a decision tree to classify fingertip configurations [27]. To the best of our research and knowledge, their system is currently unique in providing fingertip registration with less than five fingers present. However, their approach can only be considered a starting point for further research as the accuracy of approximately 80% is still too low for productive use.

PROCESSING PIPELINE

At the core of our processing pipeline lies an algorithm introduced by Matas et al. called Maximally Stable Extremal Regions that reveals the hierarchical structure of extremal regions in an image. An extremal region is a particular type of distinguished region and can be defined as a set of connected pixels that are all either of higher or lower intensity than pixels on the region's outer boundary. Given the fact that objects in diffuse front illumination setups appear brighter the closer they get to the surface, the representation of image content in terms of extremal regions provides two compelling advantages:

- An extremal region only defines a relative relationship between the contained pixels and those surrounding the region. Since it is independent from absolute intensity values,

[1] see `http://ccv.nuigroup.com/`

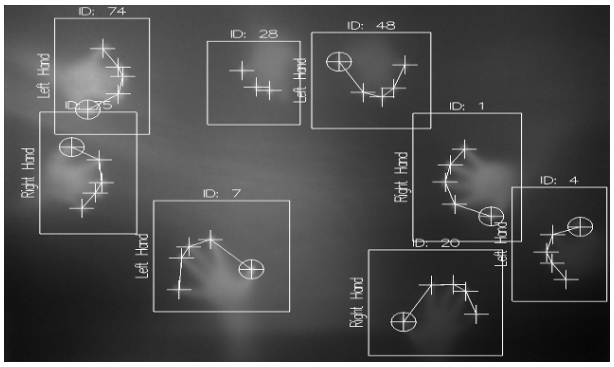


Figure 2. Finger and hand detection with our novel processing pipeline.

it is more robust in the presence of non-uniform illumination than approaches relying on a global threshold.

- Each extremal region can include further extremal regions, thereby organizing distinguished image regions in a tree structure. That structure is later used to reveal relationships between objects such as grouping contact points belonging to the same hand.

A processing pipeline usually starts with *distortion correction* and *illumination correction* in order to normalize the camera image. However these steps are considered basic image processing operations and have therefore been omitted for the sake of clarity. Our novel processing pipeline (Figure 2) comprises the following steps:

1. **Region of Interest Detection** to reduce the computational cost of subsequent processing steps
2. **Maximally Stable Extremal Regions** analysis to create a hierarchical structure
3. **Fingertip Detection**
4. **Hand Distinction** to group all revealed fingertips from the same hand into clusters
5. **Hand and Fingertip Registration** if all fingers of a hand are simultaneously touching the surface and
6. **Tracking** to follow fingers and hands across multiple frames

Step 1: Region of Interest Detection

Usually only a limited area of the tabletop surface is utilized at a time, making it appear as unnecessary overhead to process the complete camera image at every frame. Performing a simple region of interest detection first provides the following advantages:

- Applying the fingertip detection algorithm only to the relevant image areas that might represent an object or a hand on or above the surface significantly reduces computational costs.
- Several resulting regions of interest are independent and can be processed in parallel, in order to take full advantage of the threading capabilities of modern CPUs.

As the tabletop surface is illuminated from below, objects above the surface appear brighter the closer they get to the surface. Hence a simple *thresholding* operation on the intensity values seems appropriate in order to reveal regions of interest in the image. We apply a very small brightness threshold, rejecting all pixels less than 2% brighter than the background. In our setup, hands hovering less than 20 cm above the interaction surface are on average 5% brighter than the background and can thus still be detected. A drawback of this conservative approach is that illumination changes may erroneously be interpreted as user interaction. However, this only reduces the performance advantage of our processing pipeline without affecting the actual detection accuracy.

Step 2: Maximally Stable Extremal Regions

Maximally Stable Extremal Regions (MSER) is a widely used blob detection algorithm first described by Matas et al. in [16]. It robustly detects distinguished regions (i.e. regions that can be detected in image sequences or multiple views of the same scene with high repeatability). In their algorithm, Matas et al. introduced a new type of distinguished region called a maximally stable extremal region. This concept is an extension of extremal regions which are defined as follows:

Extremal Region

Given a region R in an image I, this region is called extremal if and only if all pixels within this region are either of higher or lower intensity than the pixels on the region's outer boundary Ω_R:

$$(\forall p \in R,\, \forall q \in \Omega_R \mid I(x_p, y_p) < I(x_q, y_q)) \quad \vee$$
$$(\forall p \in R,\, \forall q \in \Omega_R \mid I(x_p, y_p) > I(x_q, y_q))$$

The concept of maximally stable extremal regions additionally includes a stability criterion based on a region's relative growth.

Maximally Stable Extremal Region

Be R_{i-1}, R_i, R_{i+1} regions in an image I such that $R_{i-1} \subset R_i \subset R_{i+1}$. Region R_i is a maximally stable region if and only if R_{i-1}, R_i, R_{i+1} are extremal regions and the following stability property attains a local minimum in i:

$$s_\Delta(i) = \frac{\mid R_{i+\Delta} \mid - \mid R_{i-\Delta} \mid}{\mid R_i \mid}$$

with Δ defined as a user-defined constant.

Matas et al. describe the detection process of these regions informally as follows (Figure 3). Assume I a gray-level image thresholded on all possible intensity values $i \in \{0, \ldots, 255\}$ resulting in 256 binary images B_i. B_0 contains only black pixels while in B_{255} all pixels are white. Hence, when iterating through B_i with ascending i, the initial black image gradually turns into a complete white image with new white blobs appearing and existing blobs increasing in size. Considering the connected components from all thresholded images, each of them corresponds to an extremal region. Furthermore, each extremal region R_i at intensity threshold i is contained by exactly one extremal region from each B_j with $j > i$, since connected components do not decrease in size with increasing intensity threshold. As a result, extremal regions can be organized in a tree-like structure where the root node consists of an extremal region that covers the entire image area.

This structure is called a *component tree*. Their proposed algorithm runs in quasi-linear time with respect to the number of pixels.

In 2008 Nistér et al. proposed a different algorithm that reveals maximally stable extremal regions in true linear time [21]. Unlike the original algorithm, it does not resemble a rising waterfront continuously filling holes in the intensity height image, but rather a flood-fill spreading across the image.

In layman's terms, the algorithm can be described as pouring water into a height field that is derived from the brightness values in the camera image. Water is poured on an arbitrarily chosen point from which water starts flowing downhill. Once the lowest point of a sink is reached, water starts filling up. The filling of a sink would correspond to the growing of a connected component. Whenever the water surface of the sink did not notably change in size after filling to a certain amount, this connected component can be considered stable according to the above definition. Once a sink is fully filled, water starts pouring into adjacent sinks. The algorithm is finished after the whole image has been flooded.

Linear-time algorithm

The algorithm relies on the following data structures that describe the current state of execution:

- A binary mask marking the already visited pixels
- A priority queue of pixels on the outer boundary of the currently flooded image region. The queue is prioritized based on the pixels' intensities (lower intensities first).
- A stack of components. A component corresponds to a sink that is currently being filled with water. Therefore a component is not always necessarily an extremal region. The topmost component on the stack corresponds to the currently considered component. Whenever a sink is filled and adjacent sinks exist, a new component is pushed onto the stack and the neighboring sink is explored. Therefore the maximum stack size is equal to the number of gray levels in the image, which at most would total 256.

The algorithm proceeds as follows [21].

1. Choose an arbitrary starting point, make it the current pixel and mark it as visited.
2. Initialize a new component with the current pixel's intensity and push it onto the stack.
3. Examine all neighboring pixels of the current pixel and mark them as visited.
   - If the intensity is higher than the current pixel's intensity, then the neighboring pixel is added to the priority queue of boundary pixels.
   - If the intensity is lower, then the current pixel is added to the priority queue and the neighboring pixel is made the current pixel. Continue with 2.
4. All neighboring pixels of the current pixel have either been visited or have higher intensity. Therefore accumulate the component with the current pixel.
5. Retrieve the next pixel from the priority queue.
   - If its intensity is equal to the current pixel's intensity, continue with 3.
   - If its intensity is higher than the current pixel's intensity, all components on the stack are merged until the intensity of the topmost component is greater or equal to the next pixel's intensity. Continue with 3.
   - If the queue is empty, the algorithm terminates.

Modifications of the algorithm

In order to accelerate the subsequent processing steps, the algorithm has been modified to gather additional information on extremal regions and their spatial structure during its execution.

Extension to reveal all extremal regions

The standard algorithm only reveals extremal regions that are maximal according to the stability criterion. Hence a number of intermediate extremal regions are being discarded and do not appear in the component tree. However, these additional regions might convey important information as to the spatial relationship among regions that will be exploited extensively in the hand distinction processing step. Thus, the algorithm design has been changed in order to identify all available extremal regions.

Characterization of extremal regions

In order to be able to use the identified regions for fingertip recognition, a set of local descriptors will be used. Due to the incremental growth of components, the following features can be efficiently computed during execution of the algorithm.

Intensity statistics

The mean and variance of the intensity distribution characterize the pixel's intensity values within a component. During the sequential growth of components through accumulation of new pixels and merging of existing components, these statistics can be efficiently computed.

Image moments

In [10], Hu described a widely used set of shape descriptors based on image moments that are invariant to translation, rotation and scale. In our algorithm the first of his 7 shape descriptors will be used:

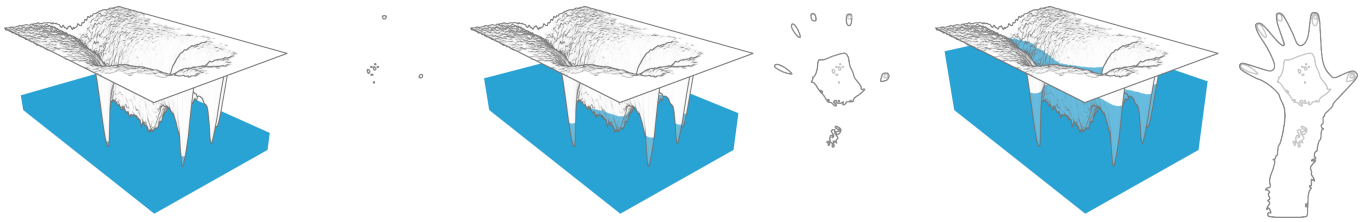
$$\phi_1 = \nu_{20} + \nu_{02}$$

where ν_{pq} denotes the *normalized moment* of order $(p + q)$ with $p, q \in \mathbb{N}$ (see [20] for a detailed description).

Bounding volumes

Image moment based shape descriptors provide a performant yet unintuitive way of describing a region's shape; however, it is sometimes useful to find a simpler representation based on geometric primitives. Therefore a region's shape will be additionally described using the oriented bounding ellipse. A non-optimal approximation can easily be derived using *image moments* of zeroth, first and second degrees [20].

Step 3: Fingertip Detection



(a) (b) (c)

Figure 3. Visualization of the *Maximally Stable Extremal Region* algorithm as a rising waterfront(from a to c). On the left hand side of each illustration, the raw camera image is represented as a height map with the waterfront indicating the level of the current intensity threshold. The isolines on the right hand side show the outlines of the image at the current and previous threshold levels.

The component tree resulting from the MSER processing serves as the foundation of the fingertip detection and the following processing steps. This tree structure has two important properties with respect to its contained *extremal regions*:

- The darkest and simultaneously largest extremal region forms the root of the component tree.
- For all child-parent relationships maintains the property that the child is smaller in size and has brighter intensity than its parent.

As fingertips in contact with the surface create the brightest spots in an image, it follows that only leaf nodes of the component tree can be considered fingertip candidates.

These fingertip candidates are evaluated as follows:

1. Identify the extremal regions that represent the finger of a fingertip candidate. This applies to all parent regions of the candidate up to a maximum size and as long as these do not have siblings (i.e. the candidate region is the only leaf contained in the parent's subtree). The largest of these regions will subsequently be referred to as *finger*.
2. For each candidate region a feature vector is compiled containing the following properties:
   (a) Bounding ellipse of the candidate region.
   (b) Bounding ellipse of the finger region.
   (c) The maximum value of the first of the seven shape descriptors introduced by Hu for all regions below the finger region.
   (d) The depth of the subtree having the finger region as its root node.
   (e) The maximum relative growth in pixel size of any child-parent relationship between the candidate and the finger regions.
   (f) The relative growth in pixel size of the finger region with respect to the candidate region.
   (g) The ratio of the intensity ranges of the candidate and finger regions respectively.
   (h) The number of pixels within a defined distance that have an intensity of less than 10% of the candidate region's mean intensity. This feature is based on the observation that pixels within a fingertip's contact area are close to a certain number of background pixels, hence resulting in a significant intensity fall-off within the vicinity.

   Properties 2a, 2b and 2c are used to discard candidate regions that do not exhibit the typical, rounded shape and average size of fingertips. Using property 2h, the algorithm is able to discern fingertip contacts from bright regions at the wrist and palm in case the hand is resting on the surface during interaction. The remaining properties describe the characteristic intensity fall-off around the contact area due to the oblique or vertical finger orientation. They are hence able to distinguish fingertip contacts from contact areas of similar size and shape.
3. Calculate a confidence score C using the weighted sum of the feature vector:
$$C = \sum_{f_i} w_i \cdot f_i \tag{1}$$
   The weights have been chosen such that range differences between features are compensated for without awarding any of the features too high of an influence on the final score. We adjusted these values manually and achieved convincing results. A machine learning approach would certainly ease setting up the system under varying conditions.
4. Two thresholds have been defined on the confidence score to classify candidate regions either as *no confidence*, *low confidence* or *high confidence*. However, having a *high confidence* score in a single frame is not sufficient for a region to be considered as a fingertip. A temporal hysteresis has been implemented that requires a candidate region to achieve at least once a *high confidence* and never a *no confidence* score in three consecutive frames in order to be classified as a fingertip. Note that this does not require the finger to be stationary during these three frames as candidates are continuously tracked while moving.

Step 4: Hand Distinction

In this approach, the grouping of fingertips is robustly established solely using the information from the previous processing steps, that is the component tree and the identified contact points that to a given degree of confidence correspond to fingertips. These fingertips may or may not belong to the same hand or even the same user. As the respective extremal regions are leaves in the component tree, they are contained in a number of regions of higher level until all of them are contained in the root region. Hence there already exists a spatial clustering

of these regions based on intensity values. Therefore, fingertips could simply be grouped gradually while ascending in the component tree until a homogeneity criterion such as the maximum distance between fingertips or the maximum size of the parent region is violated. While that approach generally works reasonably well, problems arise under certain conditions. In the case of excessive ambient light such as from stray sun light, the amount of contrast in the camera image is heavily reduced. However, with reduced contrast less extremal regions are revealed resulting in a sparse component tree. That loss of spatial information might lead to an erroneous clustering or a dead-lock situation in which too many fingertips are children of the same region.

Therefore a more robust approach will be presented here that combines the advantages of the idea described above combined with agglomerative hierarchical clustering. Hierarchical clustering itself would be unsuited in our scenario because of its high computational complexity. Tabletop surfaces continually grow in size, hence enabling the simultaneous interaction of multiple users. Due to the large number of resulting touch points, unaided nearest neighbor clustering would be highly inefficient for real-time application.

The idea is to only cluster a small subset of touch points at a time using agglomerative hierarchical clustering thereby reducing the impact of the polynomial complexity. The spatial clustering provided by the component tree, although erroneous at times, can serve as a reasonable input. The outline of the algorithm is as follows:

1. Traverse the component tree in postfix order. This ordering ensures that all children have been processed before the node itself is processed. For each node do:
   - If the node is a leaf node and has been classified with at least low confidence as a fingertip, create a new cluster containing this node.
   - Otherwise create the current set of clusters from all child nodes and process these as follows:
     (a) Compute the distance matrix among all pairs of clusters of this node using a distance function $d(C_1, C_2)$.
     (b) Find the pair of clusters C_1, C_2 with the lowest distance $d_C(C_1, C_2)$ that still satisfies the distance criterion $\mathcal{D}(C_1, C_2, d_C)$.
       – If such a pair of clusters exists, merge the two and continue with 1a
       – Otherwise the clustering is finished.

The *complete-link* (furthest neighbor) distance function is used in this instance as it takes into account the distance to the fingertip that is furthest away. It is defined as follows:

$$d_C(C_1, C_2) = \max_{u \in C_1, v \in C_2} d(u, v) \tag{2}$$

with d being a distance measure. Here Euclidean distance is used as distance measure. Combining *complete link* with an appropriate *distance criterion* $\mathcal{D}$ that places a constraint on cluster growth has been found to work best to cluster fingertips into hands.

$\mathcal{D}$ is set to the maximum expected distance between any two fingertips of the same hand. The maximum distance is usually the one between the thumb and the little finger. However, as user interaction using only these two fingers is rather rare, the criterion has been restricted further based on the combined number of fingertips in the clusters. The lower the number of fingertips contained in both clusters, the smaller the maximum allowed distance is. Hence fingertips that are at an unusually large distance from each other will only be merged once fingertips in intermediate positions have been found. Therefore this extension makes the criterion more robust as it alleviates negative effects of the chosen maximum finger distance as being considered too large or too small.

As a result of this image analysis, every detected touchpoint is assigned to a certain parent region that corresponds most probably to a single hand. The hand distinction is robust until individual hands physically touch each other. In case this accidentally happens, the problem can be eliminated based on frame-to-frame coherence. If instead this state of touching or crossing hands remains present over several frames, all involved touch events will be assigned to the same cluster. Given that the situation of touching hands is also very salient to users and generally not desired during interaction, we consider this to be a minor issue.

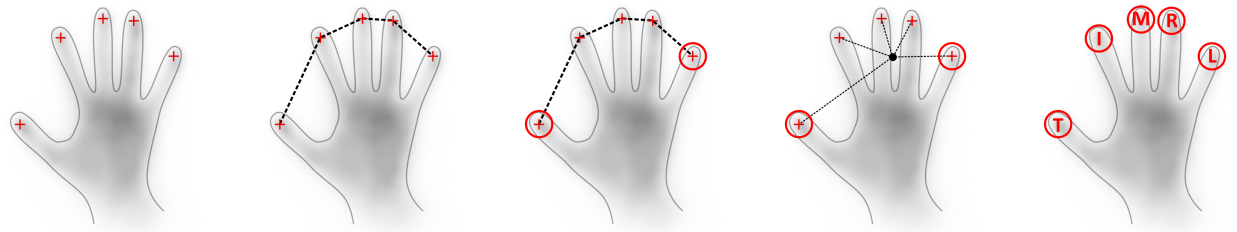
As a result of this step, every node in the hierarchical structure has a unique ID assigned that remains valid until the system looses track of it. Our system is able to keep track of hovering hands up to almost 20 cm above the screen due to their relatively large size. Applications allowing the users to choose between different tools should thus assign the selected interaction tool to the parent node that constitutes the hand rather than the selecting finger in order to keep the association when the user removes the finger from the screen.

Step 5: Hand and Fingertip Registration

The hand and fingertip registration comprises the following steps (Figure 4):

1. Order the five fingertips along the hand such that either the thumb or the little finger comes first.
2. Identify the thumb from the set of fingertips. The fingertip registration of the remaining fingertips follows from the previous ordering.
3. Infer from the fingertip registration whether the considered fingertips are from a left or right hand.

The first step is crucial as an erroneous ordering cannot be corrected at later stages of the registration process. Au et al. for instance propose an angular ordering around the centroid of the five fingertips [2]. However, the centroid usually deviates too much from the perceived center position as the index, middle, ring and little fingers are generally located very closely to each other. While this approach is robust if fingertips are aligned as an arc on the surface, considering if one of the fingers is moving (e.g. the index finger performs a sliding gesture downwards), the ordering might swap the thumb and index fingers. Hence another ordering will be proposed



(a) Initial fingertip positions of a hand in resting position.

(b) Fingertip ordering along shortest path.

(c) Endpoints are identified as either thumb or little finger.

(d) Identify thumb as the finger furthest away from the centroid of all five fingertips

(e) Fingertips can now all be identified given the thumb (T) and the ordering.

Figure 4. Steps of the fingertip registration process. The hand shape and fingertip positions used in the drawing were copied from an actual camera screen shot of the prototype.

here based on the shortest path along the fingertips which is computed as follows:

1. Be $\mathcal{F}$ the set of all fingertips f_i. The set $\mathcal{D}$ is defined as the distances among all possible pairs of fingertips:

$$\{\{d(f_i, f_j), f_i, f_j\} \mid (f_i, f_j) \in F \times F\}$$

   with $d(f_i, f_j)$ denoting the Euclidean distance between f_i and f_j.

2. For each fingertip f_i assign a unique label $L(f_i)$.

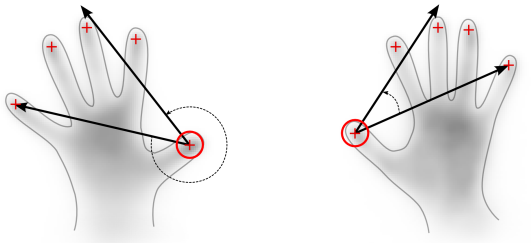
3. Iterate $\mathcal{D}$ in increasing order of $d(f_i, f_j)$. Be $\mathcal{C}$ the set of all edges forming the fingertip contour. For each pair (f_i, f_j) do

   - If $L(f_i) \neq L(f_j)$:
     (a) Add (f_i, f_j) to $\mathcal{C}$.
     (b) Assign a common unique label to all fingertips f_k that either fulfill $L(f_k) = L(f_i)$ or $L(f_k) = L(f_j)$.
   - If $\mid \mathcal{C} \mid = 4$:
     (a) All fingertips have been included in the contour. (f_i, f_j) denotes the pair of fingertips on the contour that are furthest apart from each other. f_i and f_j define the endpoints of the contour which by definition of the ordering correspond either to the thumb or the little finger.

The above ordering relies on the assumption that along the shortest path the thumb and little finger are the two fingertips that are furthest apart from each other. Since a rather unnatural hand pose would be required to violate that property, it was deemed a valid assumption for regular use.

The next step in the registration process is to uniquely identify the thumb. As the previous ordering already reduced the number of candidate fingertips, the task is equivalent to distinguishing thumb from little finger. Since the index, middle, ring and little finger influence the position of the centroid as mentioned above, thumb and little finger can be distinguished with respect to their distance to the centroid. The little finger is the one located closer to the centroid while the thumb is the one positioned further away.

Finally the registration of the set of fingertips as belonging to the left or right hand remains to be done (Figure 5). Be



(a) (b)

Figure 5. Hand registration using the angle between the little finger and the remaining fingertips with respect to the thumb.

f_T, f_I, f_M, f_R and f_L the thumb, index, middle, ring and little finger respectively. Based on the angle between the the vectors $\overrightarrow{f_T f_L}$ and $\overrightarrow{f_T f_I}$, left and right hand can be easily distinguished. If that angle is smaller than $180°$, the set of fingertips is classified as right hand, otherwise as left hand. In order to make the registration more robust in presence of finger movement such as the aforementioned sliding down gesture of the index finger, the vector $\overrightarrow{f_T f_{IMR}} = \overrightarrow{f_T f_I} + \overrightarrow{f_T f_M} + \overrightarrow{f_T f_R}$ will be used instead of $\overrightarrow{f_T f_I}$ only.

Hence the registration is defined as follows:

$$\begin{array}{rcl} \mid \overrightarrow{f_T f_L} \times \overrightarrow{f_T f_{IMR}} \mid & < & 0 \Rightarrow \text{Left Hand} \\ \mid \overrightarrow{f_T f_L} \times \overrightarrow{f_T f_{IMR}} \mid & > & 0 \Rightarrow \text{Right Hand} \end{array} \quad (3)$$

where $\times$ denotes the cross product of two vectors.

Note that finger and hand identification is only possible if the hand touches the screen in a relaxed posture with all fingers in direct contact. However, we consider this feature a mere add on to our processing pipeline providing robust multi-touch tracking with finger and hand detection. We believe that the additional information on involved fingers and hands, although limited to certain postures, can be beneficial in many applications like system control (cf. [2]).

Step 6: Tracking

The clustering of fingertips greatly reduces the complexity of matching fingertips in consecutive frames. Instead of computing all possible combinations, this step is being accelerated by only considering fingertips for nearest-neighbor matching whose containing clusters intersect. The tracking of clusters

is inferred from their contained fingertips. In case fingertips have been clustered erroneously, clusters from previous frames will be merged or split if that condition persists over a small number of frames. A window of 5 frames has worked well in our implementation.

EVALUATION AND RESULTS

In a pilot study we tested the limits of the proposed processing pipeline in terms of execution time and finger detection rate. Four students (all male, aged 24 - 27) from our research group were asked to perform multi-touch gestures on the surface. However they were reminded to not only use common gestures involving a small number of simultaneous touch points but to also use both hands with randomly changing configurations of fingers touching the surface, hence resulting in a maximum number of 40 simultaneous finger touches (and possibly even more evaluated touch points considering that users were allowed to rest their palm on the surface). We did not include intensive variability of background illumination as an independent variable as our laboratory is generally artificially illuminated. More than 4000 camera frames were captured during the interaction process that now could be replayed to the processing pipeline in order to properly measure execution times and finger detection rate.

Execution Time

As performance is dependent on the number of simultaneous user actions performed on the surface and surfaces continuously increase in size, processing pipelines have to deal with an ever increasing number of simultaneous touches. Therefore, evaluating the detection performance for up to 40 simultaneous fingers touching the surface is being considered a well-suited performance indicator of the processing pipeline.

We evaluated the performance of our algorithm on a machine with an Intel Core i7-940 CPU (2.93GHz, 4 cores) and 5.8GB of available memory running Ubuntu Linux 10.04. The camera delivered frames at a resolution of 640 x 480 at 60Hz. The reported execution times include all steps of the pipeline as well as required image preprocessing. Measuring was started just after the image had been loaded into memory and stopped when all processing and memory clean-up steps had been executed. The recorded user interaction was processed five times and we averaged the corresponding performance measures in order to minimize the influence of external factors such as the operating system.

Given the measurements shown in Figure 6, the most important result is that even for single-threaded execution, a processing performance of at least 60 frames per second was achieved. Most notably that performance was achieved even during the simultaneous interaction of 4 users, 8 hands and up to 40 fingers. Furthermore using four processing threads, processing time is capped at around 10ms regardless of how many fingers of the 8 hands simultaneously touch the surface.

Although the performance measurements prove very satisfying, it is important to keep in mind that this is only a part of the processing that is performed between a user action and the corresponding visual feedback. What determines the user's

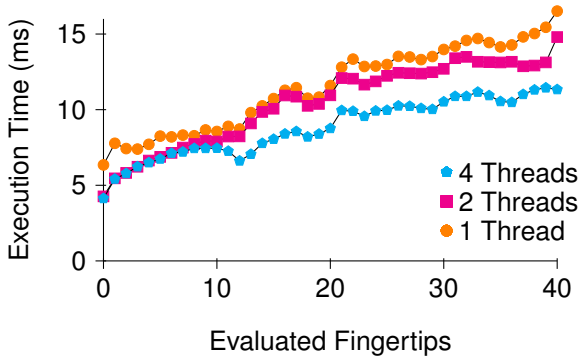


Figure 6. Execution times with respect to the simultaneous number of finger touches for different numbers of threads involved in the processing.

impression of responsiveness of our prototype is the end-to-end latency that stems from the following contributing factors (values from our prototype in brackets):

- The image acquisition, in particular the exposure time (∼16 ms)
- The transmission of image data from the camera to the computer (∼16 ms w. a Firewire camera).
- The processing of camera images (∼10 ms w. 10 threads).
- The protocol used to communicate detection results over a network interface (TUIO[2])
- The response of the application software that updates the display.

Latency introduced by the individual components can also temporarily vary due to unfavorable process scheduling as a result of the operating system. We achieved an overall latency of under 60 ms with our prototype. Further improvements can be achieved with a faster camera interface such as USB3.0 and the impact of the exposure time could be reduced by pulsing the infrared light. Since the light's intensity can be much higher during the pulse, the exposure time could be significantly shortened. This would additionally reduce the influence of ambient illumination.

Finger Detection Rate

Measuring the finger detection rate is particularly difficult as it requires a ground truth as a basis for comparison. In this case the first 1500 frames from the interaction process were considered. Given the extensive manual work required to label all visible fingertips in these camera images, this number has been further reduced to only consider every fifth frame. Hence, a total of 300 camera images were analyzed that contained on average more than 21 visible fingertips.

In order to quantify the finger detection rate, two basic measures are used. Firstly, the number of visible fingertips which have been correctly detected by the pipeline will be considered. This measure is usually called the *true positive* or *hit rate*. Secondly, given the total number of detected fingertips,

[2]Open protocol for tangible multi-touch surfaces, see `http://www.tuio.org`

how many of these have been falsely considered to be in contact with the surface will be evaluated. This second measure is usually referred to as *false positive* rate. We achieved the following results:

True Positive Rate

$$\frac{\text{Correctly detected fingertips}}{\text{Total number of visible fingertips}} = \frac{6449}{6628} \approx 0.973$$

False Positive Rate

$$\frac{\text{Wrongly detected fingertips}}{\text{Total number of detected fingertips}} = \frac{88}{6537} \approx 0.0135$$

The high *true positive rate* shows the potential of the proposed pipeline. The *false positive* rate also seems reasonably low given the constraints of the prototype, in particular the significantly uneven illumination (Figure 2). We assume that the *false positive* rate as well as the *false negative* rate could be further reduced by applying machine learning techniques to the fingertip detection.

The accuracy of the hand distinction was analyzed as well. For this purpose, more than 450 camera images were manually classified which were extracted from the interaction process at five frame intervals. The evaluation measured the success rate of the hand distinction process, defined as the percentage of hands that were correctly clustered. Hand distinction was regarded as successful if all detected fingertips from the same hand were attributed to the same cluster. However, if these fingertips were contained in more than one cluster, the clustering for this hand was considered invalid. Furthermore, if two hands were erroneously contained in the same cluster, both were regarded as unsuccessful. The results are as follows:

Success Rate

$$\frac{\text{Correctly clustered hands}}{\text{Total number of visible hands}} = \frac{2681}{2909} \approx 0.922$$

The hand distinction usually failed in the presence of *false positives* from the fingertip detection. Since cluster size was capped at five fingers, *false positives* resulted in surpassing that limit and hence interrupted the merging of clusters. This in turn reduced the *false positive* rate of the fingertip detection step, which would have positive effects on hand distinction in terms of *success rate*.

The accuracy of the hand and fingertip registration has only informally been tested. In total eight students, all male aged in their mid-twenties, from our research group were asked to place both of their hands on the surface. While no restrictions were imposed on the exact hand posture, the students were asked to adopt a position that felt natural to them. In these short tests the hand and fingertips were correctly registered for all participants. Hence the assumptions used to infer these properties appear to be reasonable. Although only a static hand position was tested, it currently seems to be the most convincing use case in which a gesture is started in this position allowing proper registration.

CONCLUSION

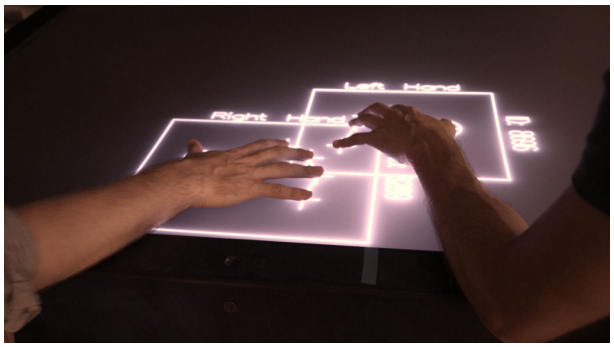


Figure 7. Close hands can still be differentiated.

We presented a novel processing pipeline for optical multitouch surfaces that relies on Maximally Stable Extremal Regions for fingertip detection rather than on simple thresholding. Extremal regions are defined as a relative relationship between an image region and its border, which renders the detection robust in the presence of non-uniform illumination. Furthermore, extremal regions can be organized into component trees which represent the spatial relationship among surface contacts and also consider further information of the captured camera images such as the hand and the arm. The leaf nodes of these trees are candidates for fingertips, which are evaluated by a set of image features and an appropriate confidence score. Using the component tree and agglomerative clustering, fingertips are merged into hands. Hand and fingertip registration for these clusters of fingertips can be performed if all five fingers of a hand are on the touchscreen in a natural pose.

We found the system to be very stable even in situations where palms and elbows were resting on the table (Figure 7) since the detection relies on the identification of the fingertips and the hand regions. Our evaluation confirms this reliability of the finger and hand detection with high true positive rates. While our approach is more computationally intensive than basic blob detection, our efficient implementation indicates that it can perform in real-time on current multi-core CPUs. We believe that advanced touch processing is at the core of further development in the realm of multi-touch interfaces. Our results clearly demonstrate novel interaction opportunities that can be achieved with smarter processing of input data from multi-touch systems which acquire some sort of depth information above the surface.

ACKNOWLEDGMENTS

This work was supported by the German Federal Ministry of Education and Research (BMBF) under grant 03IP704 (project Intelligentes Lernen). We thank the participants of our study. We also thank the reviewers of this paper for their constructive comments.

REFERENCES

1. Annett, M., Grossman, T., Wigdor, D., and Fitzmaurice, G. Medusa: a proximity-aware multi-touch tabletop. In *Proc. UIST 2011*, ACM Press (2011), 337–346.

2. Au, O., and Tai, C. Multitouch finger registration and its applications. In *Proc. OZCHI 2010*, ACM (2010), 41–48.

3. Dang, C., Straub, M., and André, E. Hand distinction for multi-touch tabletop interaction. In *Proc. ITS 2009*, ACM Press (2009), 101–108.

4. Dietz, P., and Leigh, D. Diamondtouch: a multi-user touch technology. In *Proc. UIST 2001*, ACM Press (2001), 219–226.

5. Dohse, K., Dohse, T., Still, J., and Parkhurst, D. Enhancing multi-user interaction with multi-touch tabletop displays using hand tracking. In *Proc. ACHI 2008*, IEEE (2008), 297–302.

6. Echtler, F., Huber, M., and Klinker, G. Shadow tracking on multi-touch tables. In *Proc. AVI2008*, ACM Press (2008), 388–391.

7. Hilliges, O., Izadi, S., Wilson, A. D., Hodges, S., Garcia-Mendoza, A., and Butz, A. Interactions in the air: adding further depth to interactive tabletops. In *Proc. UIST 2009*, ACM Press (2009), 139–148.

8. Hirsch, M., Lanman, D., Holtzman, H., and Raskar, R. Bidi screen: a thin, depth-sensing lcd for 3d interaction using light fields. In *Proc. SIGGRAPH Asia 2009*, ACM Press (2009), 159:1–159:9.

9. Hodges, S., Izadi, S., Butler, A., Rrustemi, A., and Buxton, B. Thinsight: versatile multi-touch sensing for thin form-factor displays. In *Proc. UIST 2007*, ACM (2007), 259–268.

10. Hu, M. Visual pattern recognition by moment invariants. *Information Theory, IRE Transactions on 8*, 2 (1962), 179–187.

11. Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A., and Fitzgibbon, A. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proc. UIST 2011*, ACM Press (2011), 559–568.

12. Jota, R., Marquardt, N., Greenberg, S., and Jorge, J. A. The continuous interaction space: Interaction techniques unifying touch and gesture on and above a digital surface. In *Proc. INTERACT 2011* (2011), 5–9.

13. Klinkhammer, D., Nitsche, M., Specht, M., and Reiterer, H. Adaptive personal territories for co-located tabletop interaction in a museum setting. In *Proc. ITS 2011*, ACM Press (2011), 107–110.

14. Leibe, B., Starner, T., Ribarsky, W., Wartell, Z., Krum, D. M., Singletary, B., and Hodges, L. F. The perceptive workbench: Toward spontaneous and natural interaction in semi-immersive virtual environments. In *Virtual Reality* (2000), 13–20.

15. Martínez, R., Collins, A., Kay, J., and Yacef, K. Who did what? who said that?: Collaid: an environment for capturing traces of collaborative learning at the tabletop. In *Proc. ITS 2011*, ACM Press (2011), 172–181.

16. Matas, J., Chum, O., Urban, M., and Pajdla, T. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing 22*, 10 (2004), 761–767.

17. Moeller, J., and Kerne, A. Zerotouch: an optical multi-touch and free-air interaction architecture. In *Proc. CHI 2012*, ACM Press (2012), 2165–2174.

18. Morris, M. R., Huang, A., Paepcke, A., and Winograd, T. Cooperative gestures: multi-user gestural interactions for co-located groupware. In *Proc. CHI 2006*, ACM Press (2006), 1201–1210.

19. Morris, M. R., Ryall, K., Shen, C., Forlines, C., and Vernier, F. Beyond "social protocols": multi-user coordination policies for co-located groupware. In *Proc. CSCW 2004*, ACM Press (2004), 262–265.

20. Mukundan, R., and Ramakrishnan, K. *Moment functions in image analysis: theory and applications*. World Scientific Publishing, 1998.

21. Nistér, D., and Stewénius, H. Linear time maximally stable extremal regions. *Computer Vision–ECCV 2008* (2008), 183–196.

22. Ringel, M., Ryall, K., Shen, C., Forlines, C., and Vernier, F. Release, relocate, reorient, resize: fluid techniques for document sharing on multi-user interactive tables. In *Ext. Abstracts CHI 2004*, ACM Press (2004), 1441–1444.

23. Roth, V., Schmidt, P., and Güldenring, B. The ir ring: authenticating users' touches on a multi-touch display. In *Proc. UIST 2010*, ACM Press (2010), 259–262.

24. Sato, M., Poupyrev, I., and Harrison, C. Touche: enhancing touch interaction on humans, screens, liquids, and everyday objects. In *Proc. CHI 2012*, ACM Press (2012), 483–492.

25. Shen, C., Vernier, F. D., Forlines, C., and Ringel, M. Diamondspin: an extensible toolkit for around-the-table interaction. In *Proc. CHI 2004*, ACM Press (2004), 167–174.

26. Teichert, J., Herrlich, M., Walther-Franks, B., Schwarten, L., and Krause, M. User detection for a multi-touch table via proximity sensors. *Proc. ITS 2008* (2008).

27. Walther-Franks, B., Herrlich, M., Aust, M., and Malaka, R. Left and right hand distinction for multi-touch displays. In *Proc. Smart Graphics 2011*, Springer (2011), 155–158.

28. Wang, F., Cao, X., Ren, X., and Irani, P. Detecting and leveraging finger orientation for interaction with direct-touch surfaces. In *Proc. UIST 2009*, ACM Press (2009), 23–32.

29. Wilson, A. D. Using a depth camera as a touch sensor. In *Proc. ITS 2010*, ACM Press (2010), 69–72.