Interacting beyond the Screen

A Multitouchless Interface
Expanding User Interaction

Philip Krejov, Andrew Gilbert, and Richard Bowden • University of Surrey

Multitouch technology is now commonplace across modern devices and provides direct interaction with an application’s GUI. However, interaction is limited to the 2D plane of the display. The detection and tracking of fingertips in 3D, independent of the display, opens up multitouch to new application areas, ranging from medical analysis in sterile environments to home entertainment and gaming. Furthermore, the additional dimension of interaction provides new avenues for UI design.

Unfortunately, using gloves, specialized hardware, or multi-camera stereo rigs to accurately track the hands limits the technology’s applications. Alternatively, detecting and tracking fingertips in real-time video is difficult due to the variability of the environment, the fast motion of the hands, and the high degree of freedom of finger movement.

The Leap Motion, an infrared stereo sensor tailored for hand interaction, accurately tracks fingertips, enabling interaction with existing multitouch applications. However, it has a limited workspace allowing only the hands to be tracked.1 Alternatively, using a generic depth sensor allows both the body and hands to be tracked for interaction. Cem Keskin and his colleagues created a means of full hand pose estimation that can also classify hand shapes.2 This approach, however, requires significant quantities of synthetic data. (For more on previous research on hand pose estimation and interactive displays, see the related sidebars.)

In contrast, our approach detects only fingertips but does so without machine learning and a large amount of training data. Its real-time methodology uses geodesic maxima on the hand’s surface instead of its visual appearance, which is efficient to compute and robust to both the pose and environment. With this methodology, we created a “multitouchless” interface that allows direct interaction with data through finger tracking and gesture recognition.3

Extending interaction beyond surfaces within the operator’s physical reach (see Figure 1) provides greater scope for interaction. We can easily extend the user’s working space to walls or the entire environment. Our approach also achieves real-time tracking of up to four hands on a standard desktop computer. This opens up possibilities for collaboration, especially because the workspace isn’t limited by the display’s size.

Our primary use of multitouchless interaction has been in the Making Sense project (www.making-sense.org), which employs data-mining and machine-learning tools to allow the discovery of patterns that relate images, video, and text. It combines multitouchless interaction and visualization to let analysts quickly

- visualize data,
- summarize the data or subsets of the data,
- discriminatively mine rules that separate one dataset from another,
- find commonality between data,
- project data into a visualization space that shows the semantic similarity of the different items, and
- group and categorize disparate data types.
Related Work in Hand Pose Tracking and Estimation

Extensive research has focused on analyzing hands for use in computer interaction. This includes systems that identify fingertip locations, because hand tracking is often the first stage in finding the fingertips.

Model-based tracking is a popular way to infer fingertip locations and account for large-scale occlusion. Such approaches can achieve 15 fps for a single hand, but this is still too slow for smooth interaction. Our approach (see the main article) can achieve 30 fps for four hands.

Many researchers optimized model fitting with the assistance of devices attached to the user. Motion capture is a common example of this; it uses markers at key locations on the hand, sometimes including the wrist. Andreas Aristidou and Joan Lasenby employed markers to reduce the complexity of mapping a model to the full structure of the hand.

After obtaining depth information from a structured-light camera, Jagdish Raheja and his colleagues removed the background. Then, they used a large circular filter to remove the palm and obtain the fingers’ masks. They located each fingertip by finding the point closest to the camera under each finger mask. To find fingers, Georg Hackenberg and his colleagues exploited the understanding that fingers comprise tips and pipes. Employing features previously applied to body pose estimation, Cem Keskin and his colleagues trained a random forest using a labeled model. This forest classified the regions of a captured hand on a per-pixel basis. They then used a support vector machine to estimate the pose.

Using a time-of-flight camera, Christian Plagemann and his colleagues applied geodesic distances to locate the body skeleton. We also employ geodesic extrema, but for finger detection, and we use Dijkstra’s algorithm to efficiently identify the candidate regions.

References

Visualization provides an intuitive understanding of large, complex datasets, while interaction provides an intuitive interface to complex machine-learning algorithms. Figure 1 shows the system in use.

Tracking Fingertips
A Microsoft Kinect serves as the sensor. First, we capture the depth image and calibrate the point cloud. We locate and separate hand blobs with respect to the image domain. Processing each hand in parallel, we build a weighted graph from the real-world point information for each hand’s surface. An efficient shortest-path algorithm traverses the graph to find candidate fingertips. We then filter these candidates on the basis of their location relative to the center of the hand and the wrist, and use them in a temporally smoothed model of the fingertips locations.

Hand and Forearm Segmentation
For each color frame from the Kinect, we locate the user’s face using a standard face detector. This provides the subject’s location for each frame. Using the point cloud derived from the calibrated depth information (the distance from the Kinect), we find the bounding box of the closest face, which we as-
Related Work in Interactive Displays

John Underkoffler at MIT designed the gesture-based computer interface featured in the 2002 film Minority Report. An interface based on that concept is available from Oblong Industries (www.oblong.com), but its reliance on specialized hardware limits it to custom installations.

However, the Microsoft Kinect has brought new levels of interaction to consumers. In an early example of applying the Minority Report interaction paradigm to the Kinect, Garratt Gallagher, from the MIT Computer Science and Artificial Intelligence Laboratory, used the Point Cloud Library (http://pointclouds.org) to track hands (see Figure A1 and www.csail.mit.edu/taxonomy/term/158). The HoloDesk used head and hand tracking to let users interact with 3D virtual objects through the screen. 1 The Space Top extended this by developing a desktop interface (see Figure A2). 2 6D Hands used two webcams to perform pose estimation and tracking, enabling a design-and-assembly application with virtual object manipulation. 3

OmniTouch was a shoulder-mounted interface that displayed graphics on users’ hands, held objects, walls, and tables. 6 It captured user interaction through a depth sensor and used a pico projector to project a multitouch overlay on the target surface. It also tracked the surface’s plane, letting users adjust the projected image to maintain the overlay.

Because the collaboration of users is important, approaches that let multiple people use a system in real time are desirable. In research by Alexander Kulik and his colleagues, six people are able to interact with the same projective warping screen. 4 The T(ether) application used infrared markers to track multiple hands interacting with a virtual environment. 5 Users viewed this environment on a tablet computer, which was also tracked through infrared markers.

Projectors are the typical means of presenting information to multiple users and allow collaborative work. Andrew Wilson and his colleagues developed a steerable projector that presents information on any surface in a room. 7 They also used it with a Kinect camera to provide reactive projection.

References


Figure A. Examples of hand- and gesture-based interaction. (1) An early Kinect-based version of an interface developed by MIT researchers. (Source: Garratt Gallagher, MIT Computer Science and Artificial Intelligence Laboratory; used with permission.) (2) Use of 3D hand tracking in 3D interaction. (Source: Microsoft Applied Sciences Group; used with permission.)
sume is the user. We smooth the depth over a 20-ms window to define the back plane for segmentation, which lets us remove the body and background from the depth image. The remaining depth space is the workspace for gesture-based interaction.

We cluster any points in this space into connected blobs. We ignore blobs smaller than potential hand shapes; the remaining blobs are candidates for the user’s arms. We then classify the points in each blob as part of a hand subset or wrist subset. This classification uses a depth threshold of one-quarter of the arm’s total depth.

**Hand Center Localization**

The hand’s center serves as the seed point for distance computation. Simply using the centroid of points would result in the seed point shifting when the hand is opened and closed. We use the hand’s chamfer distance, measured from the external boundary to find a stable center. (The chamfer distance is a transform that computes the distance between sets of points. In this context, it computes each point’s distance from the closest boundary point.) A hand’s center is the point with the greatest distance in the chamfer image.

**Candidate Fingertip Detection**

By mapping the hand’s surface and searching for extremities, we can find the fingertips, excluding the closed fingers. We conduct this search by mapping the distance from the hand’s center to all other points on the hand’s surface. The geodesic distances with the greatest local value correspond to geodesic maxima. We search for up to five extrema to account for all open fingertips.

In practice, however, the wrist forms additional extremities with a similar geodesic distance. So, we greedily compute the first seven extremities; this accounts for each fingertip, including two false positives. When the fingers are closed, the tips aren’t of interest because the fingers contribute to the fist’s formation. The extremity normally associated with a folded finger forms an additional false positive that we filter out at a later stage.

To find the candidate fingertips, we first build a weighted undirected graph of the hand. Each point in the hand represents a vertex in the graph. We connect these vertices with neighboring vertices in an 8-neighborhood fashion, deriving the edge cost from the Euclidean distance between their world coordinates.

Using the hand’s center as the seed point, we compute the first geodesic extremity of the hand graph. We use an efficient implementation of Dijkstra’s shortest-path algorithm and search for hand locations with the longest direct path from the hand’s center. (For a given source vertex, Dijkstra’s algorithm finds the lowest-cost path to all other vertices in the mesh. The geodesic extremity is the vertex with the greatest cost.) This uses the hand’s internal structure to find the extrema and, as such, is robust to contour noise, which is frequent in the depth image and would cause circular features to fail. We then find the remaining extrema iteratively, again using Dijkstra’s algorithm but with a noninitialized distance map, reducing the computational complexity.

Figure 2 shows the seven extremities found for various hand shapes. Each extremity is associated with its shortest path, as Figure 2f shows.

**Nonfingertip Rejection**

Next, we filter the candidate fingertips’ points to a subset of valid fingertips. We combine a penalty metric derived from the path taken during Dijkstra’s algorithm with a candidate’s position relative to the hand’s center.

Finding the penalty for each candidate requires the covariance of the hand and arm points, translated to the wrist location. (The covariance models the variation in pixels for the chosen body part.) Using the covariance to map the wrist in this manner takes into consideration the variability of hand shapes. We translate the covariance to form an elliptical mask centered around the wrist. If a pixel has a Mahalanobis distance within three standard deviations of the wrist, we mark it as 1; otherwise, we mark it as 0 (see Figure 3).
To find the penalty, we use both the paths from the extrema to the hand’s center and the elliptical mask. We increment a path’s score for each vertex with increasing depth through the mask, then normalize the score using the complete path’s length. When moving along this path, we include a vertex in the penalty only if its depth is less than the next vertex’s depth. (In Figure 4, the red vertices contribute to an increased score.) This heuristic is consistent with the understanding that the wrist has a greater depth than the hand’s center.

We only consider vertices that traverse with increasing depth in the penalty. The elliptical mask allows for gestures in which the fingers have a greater depth than the hand’s center—for example, someone pointing at his or her own chest. After finding the penalty for each candidate, we remove the candidate with the greatest penalty.

We reduce the remaining candidates using the Euclidean distance of the fingertip to the hand’s center, rejecting fingers with a distance less than 7 mm. This forms a sphere around the user’s hand. We consider any fingertip outside this sphere a true positive.

**Finger Assignment and Tracking**

To track and assign the fingertips between frames, we use a Kalman filter. (A Kalman filter takes a series of noisy measurements observed over time and combines these estimates to give a more accurate prediction. It often includes assumptions about the type of motion and noise that might be observed, which it uses to smooth the predictions.) We update this model using the point correspondence that minimizes the change between consecutive frames. As tracking is in 3D, we must check all possible permutations when matching points.

When assigning five or fewer points, searching all permutations requires fewer operations than using the Hungarian algorithm (an optimization algorithm that efficiently solves the assignment problem). The hand has five fingers, resulting in only 120 permutations in the worst case.

We use each detected fingertip’s world position to update a bank of 3D Kalman filters that persist from one frame to the next. To update this model, we pair detected extrema with the Kalman filter predictions by selecting the lowest-cost assignment between pairs.

Any unmatched extrema initialize a new Kalman tracker to account for the presence of new fingers. Kalman filter predictions derived from the previous frames that were not matched are updated, using the previous predicted position. This blind update is performed on the condition that the prediction’s confidence does not diminish considerably, at which point we remove it from the model.

We use the smoothed model to output each fingertip as a 3D coordinate in millimeters. For visualization, we can project these coordinates back to the image domain (see Figure 5).

It’s more accurate to track the fingertips in 3D than in the image domain because the additional depth information and filter estimation provide subpixel accuracy. If we tracked the points in the image plane, the resulting fingertips would be prone to jitter, owing to the quantization of the image.

**Tracking Validation**

Our tests, written in C++, were performed on a standard desktop PC with a 3.4-GHz Intel Core i7 CPU. To validate tracking performance, we used sequences captured from a range of users. We labeled these videos with approximately 15,000 ground truth markers, against which we measured the tracked fingers.

Over all the sequences, the average error was 2.48 cm. However, Figure 6 shows that most fingertips were accurate to within 5 mm. Our approach operated at 30 fps for each hand, with up to four hands processing in parallel.
Recognizing Gestures

Gestures for interaction typically are common across multitouch devices and provide familiarity that lets users easily perform complex tasks. However, 3D interaction differs fundamentally, and 2D gestures don't necessarily translate well to 3D. The development of gestures requires special consideration because they're integral to the user experience. They should be simple and semantically similar to the task they represent; a good example would be using a pinching gesture to zoom. However, for large-scale displays, a single-handed, two-finger pinch zoom does not scale well to the visualization's size; a two-handed, single-finger zoom is more appropriate.

To provide flexible multitouchless gesture recognition, we place middleware between the finger-tracking and application levels. We chose the gesture vocabulary to generalize across applications. To minimize the complexity new users experience when the interaction modality changes, we adapted the most common 2D gestures.

Basic Gestures

From hand and finger tracking, we know the presence and position of the user's head, hand, and fingers (see Figure 7a; additional skeletal information is available from the Kinect. We use a representation of a hand's velocity to detect a zoom gesture (see Figure 7b) and horizontal and vertical swipes (see Figure 7c), single- or double-handed. As we know the number of fingers visible on each hand, our system can recognize these gestures on the basis of specific finger configurations. For example, at the application level, users employ one finger per hand for the zoom gesture, but the system detects swipes regardless of the number of fingers.

Users can also employ an extended thumb and index finger to control a cursor. Then, to select an object placed under the index finger, the user retracts the thumb to lie alongside the palm. The system also easily detects a grasping gesture as the rapid transition from five to zero fingers.

Figure 7d shows a gesture that translates directly to a variable-size selection window in the application layer. The hands’ position controls the window’s size and position. Our demonstration system uses this particularly intuitive gesture to select different subsets of data in the visualization.

Because our approach tracks the hands in 3D, it can interpret rapid motion toward the camera as a push gesture. Users can employ this gesture with an open hand to open a menu or with the index finger to select an item. They can rotate a hand to control a circular context menu (also known as a pie or radial menu).

We can extend the system to recognize additional types and combinations of gestures. However, more complex gestures involving transitions—for example, a horizontal swipe followed by a downward swipe—could be better represented using a probabilistic approach such as hidden Markov models.

Limitations

Because the Kinect sensor is designed for full body pose estimation, it has inherent drawbacks for our approach. For example, it has difficulty mapping the hand’s surface when fingers are directed toward it. This forms a hole in the depth information that our approach does not account for. So, we avoid gestures in which users point at an onscreen object.

In addition, we parametrically tuned the distance threshold described in the section “Nonfingertip Rejection” to be invariant to adult users. In the event of a child using the system, this might lead to failure owing to the smaller hand. Our approach
could derive a more appropriate value at runtime from the chamfer distance. The maximum value at the hand’s center, once normalized by the depth, would provide the palm width.

**The Making Sense System**

The Making Sense system employs a curved projection screen that forms the main display and a secondary tabletop display (see Figure 8). Two high-definition projectors are predistorted to correct for the screen’s curvature and stitched together to form a 2,500 × 800-pixel display that presents timeline data.

Because the user stands some distance from the screen, conventional interaction such as with a mouse and keyboard isn’t applicable or required. So, the system employs the previously described gesture vocabulary. The user’s face serves as the back plane for gesture recognition; any objects behind this are ignored. This removes background distractions but allows the workspace to move with the user.

The tabletop display, which is directly in front of the user, employs a traditional multitouch overlay on a high-definition display. However, the space above the tabletop display is within the workspace for multitouchless recognition. So, by sensing the area containing the user’s arms, the system uses gestures as input for interacting with both the screen and tabletop.

The tabletop display visualizes data in the semantic space, determined through machine learning. The distance between objects represents the semantic similarity of documents in the dataset, projected through multidimensional scaling into an interactive finite-element simulation. (A finite-element simulation consists of a set of nodes connected by edges. Each node and edge has synthetic physical properties defining how they interact. The simulation constantly evaluates these interactions to minimize the overall system’s energy.) Users can select datasets and mine the data to extract rules linking the data. These rules provide the semantic similarity.

**Data Collection**

An example combining data mining and multitouchless interaction was analysis of the content of hard drives. To generate a synthetic dataset, a team of psychologists generated scripts for four individuals, each with different profiles and interests. The scripts contained each subject’s atypical online behaviors. For a month, the scripts were performed on a day-by-day basis, with the scripted behaviors alternating with normal filler behaviors. This generated a hard drive for each subject, containing text, images, and videos from webpages, emails, YouTube videos, Flickr photos, and Word documents—a truly mixed-mode dataset. Digital-forensic scientists then reconstructed this data from the hard
drives and produced high-level event timelines of user activity, along with supporting digital evidence from the hard-drive images.

**Semantic Grouping of Data**

The traditional method for clustering or grouping media is to use a large training set of labeled data. Instead, Making Sense lets users find natural groups of similar content based on a handful of seed examples. To do this, it employs two efficient data-mining tools originally developed for text analysis: min-Hash and APriori. MinHash quickly estimates set similarity. APriori, often called the shopping-basket algorithm, finds the frequency and association rules in datasets. It’s typically used in applications such as market basket analysis (for example, to determine that people who bought item A also bought item B).

The user guides the correct grouping of the diverse media by identifying only a small subset of disparate items—for example, Web, video, image, and emails. The pairwise similarity between the media is then used to project the data into a 2D presentation for the tabletop display that preserves the notion of similarity via proximity of items. More recent research has looked at automatically attaching linguistic tags to images harvested from the Internet.

The approach can also summarize media, efficiently identifying frequently recurring elements in the data. When all the events are used, this can provide an overall gist or flavor of the activities. Furthermore, we can employ discriminative mining, which mines a subset of (positive) events against another subset of (negative) events. Positive and negative sets are chosen by the user, and discriminative mining attempts to find rules that vary between them. This can be used to highlight events or content that’s salient in the positive set, letting users identify trends that, for example, describe activity particular to a specific time of day or day of the week.

Figure 9a shows the timeline view of data from the data collection example we described earlier. The horizontal axis is the date; the vertical axis is the time of day. Figure 1 shows a user employing the selection gesture on the timeline view to choose the data subset to mine. As we mentioned before, the Making Sense system presents the data on the tabletop display in terms of semantic similarity (see Figure 9b). To form semantic groups, we use a fast multiscale community detection algorithm on the distance matrix to find natural groupings. A video demonstrating the system is at www.ee.surrey.ac.uk/Personal/R.Bowden/multitouchless.
System Validation

We employed the Making Sense system on a multiclass video dataset with 1,200 videos. With only 44 user-labeled examples, our approach increased the class grouping’s accuracy from 72 to 97 percent, compared to the basic Min-Hash approach.\(^5\) (This contrasts with the popular use of equal amounts of training data and test data.)

Making Sense users have reported that they found the gestures intuitive and responsive. While mining data, they used the gestures for in-depth analysis. They used the zoom gesture to look at tight groupings of data in the timeline visualization. They used scrolling both to explore the data, moving between clusters of activity, and to navigate a thumbnail summary of the data.

The users found the timeline representation enlightening because it visualized trends in data usage. For example, the horizontal grouping in Figure 9a indicates repeated evening use of the computer. Also, the real-time data mining let users drill down into the data for increased detail, so that they could fully explore the data.

We plan to improve the system in several ways. Instead of relying on face detection, we’ll use the Kinect’s skeletal tracker to establish more accurate hand and forearm segmentation. We’ll also incorporate a probabilistic framework in which we can combine several hypothesis of fingertips, improving detection and accuracy.

Another area of investigation is the classification of more complex hand shapes, moving toward full hand pose estimation. A wider range of detected hand shapes would also allow recognition of more gestures, extending the system’s capabilities. \(^\star\)

Acknowledgments

The UK Engineering and Physical Sciences Research Council Project Making Sense (grant 13 EP/H023135/1) funded this research. We thank Chris Hargreaves at Cranfield for the digital-forensics work and Margaret Wilson at the University of Liverpool for the psychology work.

References


Philip Krejov is a PhD student at the University of Surrey’s Centre for Vision, Speech and Signal Processing. His research focuses on human–computer interaction, with interests in head tracking for anamorphic projection and hand pose estimation using graph-based analysis and machine learning. Krejov received a BEng (Hons) in electronic engineering from the University of Surrey and received the prize for best final-year dissertation. Contact him at p.krejov@surrey.ac.uk.

Andrew Gilbert is a research fellow at the University of Surrey’s Centre for Vision, Speech and Signal Processing. His research interests include real-time visual tracking; human activity recognition; intelligent surveillance; and exploring, segmenting, and visualizing large amounts of mixed-modality data. Gilbert received a PhD from the Centre for Vision, Speech and Signal Processing. He’s a member of the British Machine Vision Association’s Executive Committee and coordinates the association’s national technical meetings. Contact him at a.gilbert@surrey.ac.uk.

Richard Bowden is a professor of computer vision and machine learning at the University of Surrey, where he leads the Cognitive Vision Group in the Centre for Vision, Speech and Signal Processing. His research employs computer vision to locate, track, and understand humans, with specific examples in sign language and gesture recognition, activity and action recognition, lip reading, and facial feature tracking. Bowden received a PhD in computer vision from Brunel University, for which he received the Sullivan Doctoral Thesis Prize for the best UK PhD thesis in vision. He’s a member of the British Machine Vision Association, a fellow of the UK Higher Education Academy, and a senior member of IEEE. Contact him at r.bowden@surrey.ac.uk.