DANTE VISION: IN-AIR AND TOUCH GESTURE SENSING FOR NATURAL SURFACE INTERACTION WITH COMBINED DEPTH AND THERMAL CAMERAS

Elliot N. Saba¹, Eric C. Larson¹, Shwetak N. Patel¹,²

¹Electrical Engineering, ²Computer Science & Engineering
UbiComp Lab, DUB Group, University of Washington, Seattle, WA (USA)
{sabae, eclarson, shwetak}@uw.edu

ABSTRACT

Researchers have paid considerable attention to natural user interfaces, especially sensing gestures and touches upon an un-instrumented surface from an overhead camera. We present a system that combines depth sensing from a Microsoft Kinect and temperature sensing from a thermal imaging camera to infer a variety of gestures and touches for controlling a natural user interface. The system, coined Dante, is capable of (1) inferring multiple touch points from multiple users (92.6% accuracy), (2) detecting and classifying each user using their depth and thermal footprint (87.7% accuracy), and (3) detecting touches on objects placed upon the table top (91.7% accuracy). The system can also classify the pressure of chording motions. The system is real time, with an average processing delay of 40 ms.

Index Terms— depth imaging, thermal imaging, natural user interface, sensor fusion

1. INTRODUCTION

Interest in touch and gesture sensing in natural user interfaces has increased dramatically in the past five years, especially in applications that use a camera to analyze interactions on arbitrary surfaces (for example, interactions with smartphones). Because of the real time nature and complexity of these vision analysis problems, this area is gathering more interest in the signal processing community.

Much of the work in camera-based interfaces has looked at only a single sensor stream, for example RGB [1-4], depth [5] or thermal imaging [6], [7]. Recently there has been interest in combining multiple visual sensor streams to improve the robustness of the interface (for example, combining RGB and depth [8]). Similarly, we hypothesize that combining depth and thermal camera streams will provide more robust sensing and provide distinct, new sensing capabilities for the interface. To this end, we present Dante, a system which is capable of (1) robust multi-finger touch down detection, (2) multi finger or whole hand chording, (3) in-air finger tracking, (4) touch pressure classification, (5) multi-user classification, and (6) handling changes in geometry of the touch surface (such as detecting touches on arbitrary objects). The cornerstone of Dante is its ability to seamlessly fuse information from each camera stream.

2. MOTIVATION AND RELATED WORK

Traditional (RGB) cameras have seen considerable use for detecting hand gestures and touch points [1-3]. There has been substantial work in skin color matching, contour detection, and motion tracking to identify various body parts and infer multi-touch gestures [1], [4], [9], [10] as well as pressure of touch via finger deformation [11]. The use of RGB cameras, however, is fundamentally limited by the information that can be analyzed from a scene in real time.

More recently, near infrared (N-IR) depth cameras have also been applied alone or in concert with RGB cameras for gesture detection and tracking [8], [10]. The introduction of the Microsoft Kinect has recently re-spawned interest in using depth imaging for natural projected interaction [5], [12]. However, common problems with this approach include: (1) if hand motion is too fast adjacent depth values blur together, (2) sensor noise elongates the detection time for touch down events, and (3) differences in the anatomy of users’ hands makes general touch detection (such as palm presses, rather than finger presses) less reliable. These downsides can be mitigated (and in many scenarios eradicated) by fusing the depth information with that of a thermal imaging camera.

In the thermal imaging space, also known as far infrared (F-IR), the work of Oka et al. [7] and Sato et al. [13] combines an overhead RGB and thermal imaging camera for
hand segmentation and fingertip tracking. They use the trajectory of extracted finger tips as input to a hidden Markov model to identify one of six different in-air gestures. In previous work [6], we have used a thermal imaging camera to drive an interactive projected user interface. We detect the residual heat transfer from a hand to an arbitrary surface as the hand moves along the surface (e.g., chording movements). We were also able to detect more general touches, such as palm prints, and the pressure a user applies to a surface. Common problems we ran into stemmed from occlusion of the heat transfer: (1) the detection of touch down events and (2) chording motions where the hand moves over a touch point that has just been made.

The drawbacks of using RGB, depth, and thermal cameras in isolation have slowed the adoption of interactive projection systems. The fusion of these technologies, however, promises to mitigate the drawbacks of each camera system and provide a rich application space for combined real time feature integration from each video stream.

### 3. METHODOLOGY

This section introduces the hardware and algorithms used in our system. Design decisions were tailored to the real time constraints of the system—in order for the interaction to be as natural as possible, we constrain the processing lag to be no more than one frame at 20 frames per second (a delay of 50 ms). This can be assumed to be a highly responsive, real time system [14].

#### 3.1. Camera Hardware

For our depth camera we use the Microsoft Kinect running at 640x480 resolution and 11-bit depth resolution (approximately 3 mm per digital level). To interface with the depth camera, we are using the Microsoft Kinect for Windows SDK beta [15]. For our thermal imaging camera we used the RazIR NANO, which contains an un-cooled Focal Plane Array (FPA) micro-bolometer sensor with 160x120 pixel resolution [16]. The thermal sensor is tuned for wavelengths in the IR spectrum between 8 and 14 µm, and captures data with 11-bit thermal resolution (about 50 mK per digital level).

The field-of-view of the thermal camera is 15° with stock lens and, thus, the thermal imaging camera sits about 30 cm back from the Kinect as shown in Figure 1. There is no additional instrumentation on the surface or user. We note that the wooden tabletop used for projection and interaction is not special and a variety of other materials may be used such as plastics, laminates, or even paper [6].

#### 3.2. Algorithms

Our hardware offers two distinct data streams, one from thermal and another from depth, each with its own unique advantages and capabilities. For example, depth sensing provides poor segmentation when the hand is near the touch surface [5], [12] whereas thermal imaging provides an extremely robust method of hand detection, even when the hand is close to the table [6]. Figure 2 provides a flow diagram of our algorithms which are outlined in the following subsections. For the remainder of the paper, the \( t \)th frames captured from the Kinect depth sensor are represented by \( D \), and frames captured from the thermal camera are represented as \( T \). For residual heat transfer detection and pressure classification, we use the methods outlined in [6].

##### 3.2.1. Homography

We first register the thermal and depth images using a planar homography (i.e., geometric only), allowing us to transform to and from depth and thermal image spaces. This requires a 4-point calibration in both cameras and is valid for the plane of the tabletop that users interact with. We found other optical disparities from the cameras and lenses to be insignificant. We implement the homography using well known perspective transforms from OpenCV [17].

##### 3.2.2. Background Subtraction and Segmentation

In order to segment hands from our frames, we must first identify frames containing only the background (i.e., no hands). We consider a frame to contain only background if there are no connected heat sources occupying a large area in the image (i.e., a hand) for 10 continuous frames. In this case, the current frames from each camera are taken to be the “background”, denoted \( D_{\text{back}} \) and \( T_{\text{back}} \). These background frames are subtracted from subsequent frames to produce \( D_{\text{diff}} = D_{t} - D_{\text{back}} \) and \( T_{\text{diff}} = T_{t} - T_{\text{back}} \). If a large connected heat source is detected, \( D_{\text{diff}} \) is thresholded to

![Figure 2: The algorithmic flow diagram, example images shown for each stage in callouts.](image-url)
isolate objects above the table surface, and $T_{i\text{diff}}$ is segmented using Otsu’s method [18]. Because the background and hand temperatures can be assumed to occupy significantly different temperature ranges, Otsu’s thresholding method provides an efficient means of dynamically tracking the optimal point of separation in the temperature histogram.

### 3.2.3. Hand Detection
The segmentations from $T_{i\text{diff}}$ (assumed to be hands) are operated on with a binary morphological opening to fill in any gaps in the segmentation. Each connected component (denoted as $h_i$) is then processed using the Convex Hull method detailed in [19]. This results in a set of $N$ convex and $M$ convexity defect points, $\{<x_i,y_i>_c, ..., <x_N,y_N>_c\}$ and $\{<x_i,y_i>_d, ..., <x_M,y_M>_d\}$ respectively, for each $h_i$. The $M$ convexity defects can be assumed to surround the center of the hand yielding a stable estimate of the palm when averaged. This centroid of the hand is transformed into depth space (using the homography), resulting in an estimate of the height of the hand over the surface. If the hand is above a threshold (about 2 cm), the hand detection is performed on $D_{i\text{diff}}$. This is because the homography no longer reliably transforms from the thermal perspective to depth due to parallax differences between the two cameras.

### 3.2.4. Fingertip Tracking
The points along the convex hull, $\{<x_1,y_1>_c, ..., <x_N,y_N>_c\}$ can be assumed to correspond reasonably to the fingers of each $h_i$. To reduce false positives, we provide a distance constraint from the centroid of the hand and perform a local hit-and-miss transform around each point on the convex hull using a disk shaped kernel. This also has the added bonus of refining the fingertip location—the closest point found using the hit-and-miss transform gives a more stable estimate of the finger than the convex hull points alone. This results in a set of $O$ fingertip locations $\{<x_o,y_o>_1, ..., <x_o,y_o>_O\}$.

### 3.2.5. Feature Extraction
Once we have stable estimates of finger location, we extract two features: mean and median finger point depth around a neighborhood of $5\times5$ pixels, $D_{i\text{diff}}(x'_i,y'_i)$, where $i,j={1,2,...,5}$. These features are fed into a C4.5 tree classifier [20], as implemented by the WEKA Machine Learning toolkit [21]. The output of the classifier is one of two states: (1) touch down or (2) hover. This provides us with an efficient means of classifying if a user touches down onto the surface.

In addition, we can combine this with the residual heat transfer detection implemented in [6] to dynamically retrain our touch classifier. If we detect residual heat transfer in previous frames from the thermal camera, then we know the user was pressing down on the surface. By saving the depth from previous frames, we can retrain the touch classifier periodically when we detect heat transfer has occurred in the vicinity of a given finger.

### 3.2.6. Multi-User Classification
We are also able to classify a given connected component $h_i$ as a specific user using a Support Vector Machine [21]. We extract the following features, (1) average hand temperature, (2) standard deviation of hand temperature, (3) area of thermal component and (4) depth of the center of the hand, to distinguish between different users that are interacting with the surface. The joint use of the thermal area and centroid depth allows the classifier to build relationships on the fused sensor streams. This classification frees users to move around the tabletop without the system needing to track anything more than current frame level features, which allows users to interact inside overlapping spaces without any instrumentation.

### 4. RESULTS
Figure 3 shows the results of the algorithm running for a single user. The thermal difference image (red channel) shows a clear indication of the previous touch points (with pressure) and the depth difference image (blue channel) shows the current points where the finger is touching down. Notice that even if the user’s hand moves quickly enough to blur the finger depths, the thermal image still provides a robust segmentation of where the fingers are located. We can use this segmentation to transform the finger endpoints into the depth image and still get a reliable measure of their depth (even though the depth image blurs the fingers together). Also notice that the user is successfully identified as “user 1” from a pool of four candidate users.

Table 1 shows the typical response characteristics for the system collected from 2 different users. Users were asked to hold their hands at 1, 2, and 3 cm above the surface for 35 seconds each. Users were then asked to touch and swipe across the surface with one, two or three fingers. Then, a block was inserted into the frame and users were asked to repeat the tests above the block. Finally, users were also asked to draw three lines across the tabletop at three increasing pressures. From table 1 we can see that the true positive rate of in-air and touch gestures is above 90%. The majority of the undetected touch events occurred at the end of the users’ swipe, as their hands moved out of frame. The
false positives from users hovering at different distances are also shown. Notice that when the hand is more than 1 cm above the surface, the false positives are less than 5%, otherwise there are a considerable number of false positives when the user hovers above the surface at less than 1 cm.

To calibrate the user classifier, four users were asked to hold their hand in view of the cameras for 3 trials (2 seconds per trial). Based on a ten-fold cross-validation, users were successfully classified with 85.5% accuracy. After calibration, each user performed another trial which was classified as that specific user with an accuracy of 87.7%. Almost all confusions occurred between users 1 and 4, while almost no hands were classified incorrectly for users 2 and 3.

All results show that the system is capable of tracking finger down events, chording, and in-air tracking with high accuracy and with a fast response time of 200 ms. In our design constraints we aimed for less than 50 ms response time. However, we discovered that the Kinect has an inherent latency of around 160 ms (our algorithms added a 40 ms processing delay). While 200 ms can be considered real time for most applications, the responsiveness in high user throughput applications (i.e., Google Earth) may not feel “naturally” responsive. More testing is needed to investigate the naturalness of this system response time.

5. CONCLUSION

In summary, we created an interactive projection system capable of sensing a variety of finger and hand touch events and in-air movements of fingers; aware of how many and which users are interacting, even if they move around the touch surface; aware of the pressures of chording motions; and capable of sensing touches on arbitrary surfaces or touch surfaces which periodically change shape.

6. REFERENCES


<table>
<thead>
<tr>
<th></th>
<th>response time (ms)</th>
<th>true positive rate (%)</th>
<th>false positive rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>in-air finger tip tracking</td>
<td>200 ms\textsuperscript{1}</td>
<td>98.7%</td>
<td>2.2%</td>
</tr>
<tr>
<td>finger down detection on table</td>
<td>200 ms\textsuperscript{1}</td>
<td>92.6%</td>
<td>0.03, 2.7, 23.2%*</td>
</tr>
<tr>
<td>finger down detection on block</td>
<td>200 ms\textsuperscript{1}</td>
<td>91.7%</td>
<td>0.59, 4.4, 26.7%*</td>
</tr>
<tr>
<td>user identification</td>
<td>200 ms\textsuperscript{1}</td>
<td>87.7%</td>
<td>N/A</td>
</tr>
<tr>
<td>heat transfer detection</td>
<td>166 ms</td>
<td>99.1%</td>
<td>0.02%</td>
</tr>
<tr>
<td>pressure classification</td>
<td>166 ms</td>
<td>96.3%</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 1. Response time and classification rates for the system. *These false positives are measured at 3cm, 2cm, and 1cm. †Note that the Kinect has a mandatory response time of about 160 ms, our processing adds about 40 ms to the overall touch detection.