Group Touch: Distinguishing Tabletop Users in Group Settings via Statistical Modeling of Touch Pairs

Abigail C. Evans,[1] Katie Davis,[1] James Fogarty,[2] Jacob O. Wobbrock[1]
[1]The Information School, [2]Computer Science & Engineering
DUB Group | University of Washington
Seattle, WA, USA 98195
{abievans, kdavis78, wobbrock}@uw.edu, jfogarty@cs.washington.edu

ABSTRACT
We present *Group Touch*, a method for distinguishing among multiple users simultaneously interacting with a tabletop computer using only the touch information supplied by the device. Rather than *tracking* individual users for the duration of an activity, Group Touch *distinguishes* users from each other by modeling whether an interaction with the tabletop corresponds to either: (1) a new user, or (2) a change in users currently interacting with the tabletop. This reframing of the challenge as distinguishing users rather than tracking and identifying them allows Group Touch to support multi-user collaboration in real-world settings without custom instrumentation. Specifically, Group Touch examines pairs of touches and uses the difference in orientation, distance, and time between two touches to determine whether the same person performed both touches in the pair. Validated with field data from high-school students in a classroom setting, Group Touch distinguishes among users "in the wild" with a mean accuracy of 92.92% (*SD*=3.94%). Group Touch can imbue collaborative touch applications in real-world settings with the ability to distinguish among multiple users.

Author Keywords
Tabletop; modeling; distinguishing users; "in the wild."

ACM Classification Keywords
H.5.3. Group and Organization Interfaces: Collaborative Computing.

INTRODUCTION
Interactive tabletops can support collaboration because of the large, shared interface that multiple people can interact with together [9,22]. However, with the exception of two systems, DiamondTouch [8] and Fiberio [24], tabletops are limited by their inability to identify users. The capacity to distinguish among users is desirable for a number of reasons, ranging from basic usability (e.g., resolving conflicting gestures carried out by different people [32]), to enabling certain application features (e.g., allowing users to separate personal and shared space on the screen [27,40]), to tracking and evaluating an individual's contributions to a group task [29]. For example, in the domain of collaborative classroom education [12], distinguishing among users can enable interventions that help students improve overall group collaboration (Figure 1).

Figure 1. Group Touch can distinguish among users "in the wild" with a mean accuracy of 92.92% (*SD*=3.94%) using only touch information. Pictured above are high school students in a classroom. The challenge of *distinguishing* among users (i.e., knowing one user's touches are different from another's) is a different challenge than *identifying* them (i.e., knowing to whom each touch belongs).

A number of methods have been proposed for distinguishing among tabletop users, but most existing approaches either rely on external sensors (e.g., [1,29,30,35,36]) or constrain interaction (e.g. [4,21,43]) so that users are prevented from taking full advantage of the tabletop's multi-touch capabilities. In our prior investigation of tabletop collaboration in high school classrooms [12], the need arose for distinguishing among users engaged in unconstrained multi-touch interaction without using external sensors, which were not practical due to the physical constraints of the classroom. No existing approach could meet this need.

This paper describes *Group Touch*, a novel user-independent approach to distinguishing among users that utilizes only the built-in capabilities of the tabletop computer and does not impose restrictions on users. Importantly, existing approaches have generally taken on the goal of trying to *identify* and *track* each user. In this work, we relax and reframe that goal, instead focusing on *distinguishing* one

CHI 2017, May 06-11, 2017, Denver, CO, USA
 ISBN 978-1-4503-4655-9/17/05…$15.00
DOI: http://dx.doi.org/10.1145/3025453.3025793

user from the next by determining when subsequent touches are from the same user or different users. We therefore do not achieve the ability to uniquely identify a user (e.g., for authentication purposes), but we do gain the ability to distinguish among multiple users during periods of simultaneous interaction. This reframing of the objective is a key insight that simplifies the problem but still retains many capabilities important for multi-user tabletop interaction scenarios. Use cases for Group Touch include: (1) resolving conflicting input that occurs when multiple people are simultaneously interacting with the same interface; (2) modeling the collaborative processes of small groups working at tabletop computers; and (3) evaluating and informing the design of software to better support small group collaboration at tabletop computers.

We explain how Group Touch works, give example scenarios in which it works, and demonstrate its effectiveness on a collection of field data acquired by groups of high school students collaborating at a Microsoft PixelSense in a classroom setting. Our results show that Group Touch can distinguish among 4-6 simultaneous users of a tabletop computer "in the wild" with an accuracy of 92.92% (*SD*=3.94%).

The contributions of this work are: (1) a novel approach to distinguishing among users at a tabletop computer that does not restrict multi-touch interaction or require additional sensors; and (2) an empirical study of the effectiveness of this approach using touch data collected entirely in a chaotic classroom field setting.

RELATED WORK

Common tabletop computers are vision-based, using either frustrated total internal reflection (FTIR) [19] or diffuse illumination (DI) [38] to detect touches. Existing approaches to distinguishing among users at vision-based tabletops fall into three categories: (1) approaches that augment the tabletop with additional sensors; (2) approaches that require the user to wear or hold external sensors; and (3) approaches that use only the built-in capabilities of the tabletop hardware. We review each of these in turn.

Approaches that augment the tabletop with additional sensors are typically reliable, accurate, and unobtrusive to the user. These approaches make it possible to track individual users for the duration of their time at the tabletop. For example, proximity sensors have been added to the edges of tabletop computers in order to track users' locations around the screen [2,41]. Multiple approaches have used cameras mounted above the tabletop to track users' arms and hands [6,29,34]. Bootstrapper [35] uses a depth camera mounted under the table to track users by their shoes and then match touches to individuals using the touch orientation detected by the tabletop's vision system.

Other approaches augment users with additional sensors, such as rings [36], wristbands [30], gloves [28], or cards [25] that communicate with tabletops' built-in sensors to identify users. Ackad et al. [1] used a combination of users' personal mobile devices and an overhead depth camera.

However, relying on external sensors is not always practical or desirable. For example, our work is motivated in part by the need to distinguish among users in the context of a study of collaborative learning at tabletop computers in authentic classroom settings [12]. The physical constraints of the study sites mean that the tabletop needs to be set up and then removed for each class session, making it impractical to use external sensors such as a depth camera that would need to be fixed in place. This problem has been encountered by other researchers studying tabletops in classroom settings [26]. Augmenting users (high school students) with wearable sensors was ruled out simply because such sensors require additional setup, storage, and maintenance, all of which increase time spent administering a learning activity and the burden of classroom management on the teacher. These factors go against evidence-based best practices and guidance on integrating new technologies into classrooms [3,10,11]. Therefore, deployments like this need an approach to distinguishing among users that does not rely on external sensors.

Several studies have investigated the use of finger orientation and hand contours, captured by vision-based tabletops' on-board cameras, to match touches to hands and users [7,13,42,43,44]. Ewerling et al. [13] developed an image processing technique to match touches to individual hands with four users simultaneously interacting with a tabletop computer. They did not explore the application of their technique to distinguishing users. Dang et al. [7] and Zhang et al. [44] both developed heuristic methods using touch orientation and distances between contact points to map touch points to hands. Mapping touches to hands could potentially improve gesture detection and thereby enhance the interactive experience. Both methods were highly effective at differentiating between the hands of single participants, but neither method was tested with groups or used to extrapolate from hands to users.

Zhang et al.'s [43] *See Me, See You* algorithm used hand contours captured by the vision system to detect finger orientation and handedness. Using touch orientation along with a touch's coordinates, Zhang et al. trained a support vector machine to predict a user's location at the tabletop, thus distinguishing among different users. Their approach, however, does not support multi-touch input, purposefully constraining users to single-point touches in order to make identification possible.

For touchscreens with capacitive sensing capabilities, Harrison et al.'s [21] capacitive fingerprinting technique takes advantage of the natural differences in electrical current passing through individual users. This approach does not restrict how users touch the screen, but a limitation of the prototype tested in Harrison et al.'s study is that it can only handle one touch at a time.

Blažica et al. [4] enable identification of users across all types of tabletop computers using hand biometrics. However, their approach is only able to distinguish among multiple simultaneous users when they place a hand on the screen in a specific position—all five fingers splayed and touching the screen. Therefore, it is not a suitable approach for distinguishing users during "in the wild" tabletop interaction, such as in a high school classroom.

Finally, application-based approaches take advantage of assumed or enforced social protocols. For example, an approach that automatically segments a tabletop interface into territories belonging to individual users [16] draws upon Scott et al.'s [39] findings that groups tend to divide the tabletop into personal and shared territories without any explicit coordination. However, social protocols can vary greatly depending on the context, limiting their generalizability as a means of distinguishing users. For example, research with adults in a lab setting has shown that participants are reluctant to reach into one another's perceived personal space or territory on the tabletop [37], supporting the notion that it is possible to distinguish users by identifying those personal territories. In contrast, children engaged in collaborative learning around a tabletop often break or ignore boundaries of personal space, actually enhancing collaboration [15,33]. Given that our context is tabletop collaboration in authentic high school classrooms, these findings suggest that assuming young users will follow adult social protocols may be unwise.

All of the approaches described above that use only the built-in capabilities of the hardware to distinguish users also impose restrictions on how users can interact with the tabletop. Some researchers argue that there are scenarios where users will be willing to sacrifice unconstrained interaction for the ability to track users [43]. However, this will only hold true when the value added by tracking users outweighs the inconvenience of artificial constraints on interaction. When this is not the case, such as when an application requires true multi-touch input, the above approaches are not suitable. Additionally, these approaches are at risk of failure with young users because it can be difficult to get children and adolescents to comply with behavioral constraints. With Group Touch, we address the problem of distinguishing users without utilizing external sensors and without constraining interaction.

DESIGN AND EVALUATION OF GROUP TOUCH

In order to support unconstrained multi-touch interaction, we forego the goal of *identifying* and *tracking* users (knowing to whom each touch belongs), which has been the focus of many of the aforementioned approaches. Instead, our aim is *distinguishing* among users (knowing one user's touches are different than another's). This means that our approach cannot enable sophisticated personalization of interfaces or authentication, but it can be used to address core usability problems that arise when multiple people interact with a single shared interface, such as determining whether simultaneous touches are by a single person performing a single multi-touch gesture or multiple people performing separate gestures (e.g., to resolve conflicts). Our approach could also be used for modeling collaboration in order to understand group working practices and inform the design of collaborative tabletop applications.

Most approaches to distinguishing among users have tackled the problem at the individual level by trying to match touches to users. By relaxing the goal of tracking and identifying, we reframe the problem at the group level; after all, the need to distinguish among individuals only arises when there are multiple users in a group. This reframing leads us to compare touches to each other instead of attempting to match touches to specific individuals.

Group Touch has two components. The first is a multilayer perceptron (MLP) model trained on touch data collected entirely "in the wild." The model predicts, given a pair of touches, whether they were carried out by the *same* person or *different* people. The second component is an algorithm that uses the predictions of the MLP model to group touches that were likely to have been carried out by the same person. The following sections describe the dataset that was used to develop and evaluate Group Touch and the design of each component in turn.

Creating the Touch Dataset

Group Touch was developed and evaluated using touch data collected "in the wild" during a study of collaborative learning at tabletop computers in classroom settings [12]. Data were collected from high-school students using five distinct multi-touch applications (Figures 2 and 3) on a Microsoft PixelSense in two different educational programs.

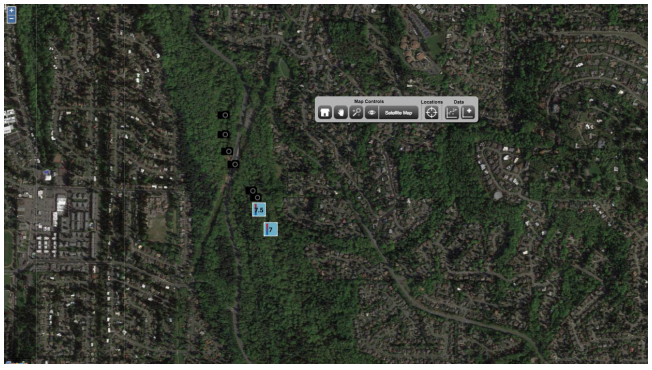

Figure 2. The mapping application used in the classroom. The two blue squares represent locations with available water quality data. Touching the squares opened up graphs and charts of the data. The shown toolbar provided access to additional navigation and settings.

The participants in the first study setting—an after school science program—were eleven 9th to 12th grade students (6 male, 5 female). Figure 2 shows the custom-built mapping application used in this setting. The teachers of the science program had requested an application that would enable students to see stream-monitoring data they had collected laid out on a map of their study area that could be panned and zoomed to different levels of detail. Students

could use the map to navigate to various data collection locations and to interact with tables, graphs, and images of data. Students interacted with the application using standard multi-touch gestures such as pinch-to-zoom, swipe-to-pan, and multiple-finger rotation. Four sessions were recorded across two class periods with the students using the computer in groups of five to six for around 30 minutes a session. A total of 1,072 touches were logged in this setting.

Participants in the second study setting—a user-centered design course—were sixteen 10th and 11th grade students (6 male, 10 female). Figure 3 shows screenshots of the applications that were custom-built for this course. The students used each application for 10-15 minutes in groups of three to four over four class sessions. One application (Figure 3, top left) enabled the students to draw on wireframes for their design project. A second application (Figure 3, top right) helped groups brainstorm questions for a usability test. A third application helped students improve their search skills (Figure 3, bottom left) by finding and comparing resources for their projects. These applications were built using the PixelSense's native interface components, which fully support multi-touch. The final application (Figure 3, bottom right) was a multi-touch-enabled browser plugin for finding and annotating, via drag and drop, real-world examples of usability heuristics. Thirteen group sessions were logged in this setting, amounting to 9,255 touches.

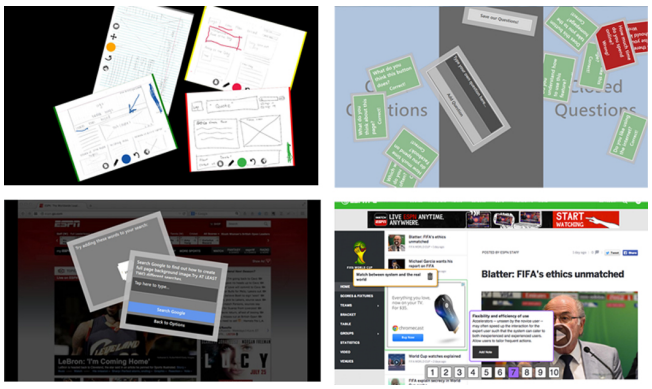

Figure 3. The applications used in the second study setting. From left to right, top to bottom, the first three applications used the computer's native SDK. The fourth (bottom right) application was browser-based. All applications supported multi-touch. (See the text for details on each application.)

In both classroom settings, the teacher determined the activities that the students carried out, and students were not given any instructions on how they should interact with the tabletop computer. The students elected to stand while using the computer and chose how to arrange themselves around the screen. They touched freely and rampantly, without any rules imposed on them to take turns or "be polite" to their collaborators. In this way, the environment paralleled many authentic classroom situations.

All 17 sessions across both study sites were video recorded with a single camera and every touch was logged by software on the tabletop. The camera was mounted on a tripod to the side of the table, angled down so that the whole tabletop screen was visible. The purpose of the camera was to enable us to manually label the author of every single touch using the video to obtain ground truth. Manually labeling the touches took around 200 hours.

The touch data collected in both settings proved to be as "messy" as might be expected from unconstrained field settings with high school students. We observed considerable variation in how students carried out standard multi-touch gestures, such as drag and rotation, echoing Hinrich and Carpendale's [23] findings from their study of multi-touch gestures on tabletops "in the wild." Students were free to move around the tabletop and often did, occasionally pushing and shoving each other. Figure 1 shows multiple students simultaneously interacting with the table, which was characteristic of much of the students' time with the table, even though taking turns and adhering to the social protocols of effective collaborative work would have been beneficial for the learning activities. Additionally, a notable proportion of student interactions appeared to be carried out without an obvious task-related purpose. For example, students regularly spun or flicked on-screen objects while engaged in off-task conversation. These behaviors have also been noted in other reports of tabletop use in classroom settings [12,26].

The MLP Model and Feature Selection

The first component of Group Touch, the MLP model, predicts whether a pair of touches was carried out by the same person or different people based on three features (Figure 4): (1) the difference in touch orientations; (2) the pixel distance between the two touches; and (3) the time difference between touches, in milliseconds.

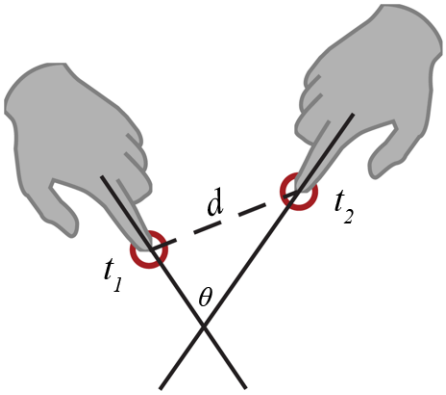

Figure 4. The touch features used to predict whether a pair of touches was carried out by the same user are: (1) the difference θ between the touches' orientations, (2) the distance d between the touch points, and (3) the time between t_1 and t_2.

The three features were selected based on the following reasoning. Consider Figure 5, a raw image captured by the PixelSense's on-board cameras. Even without any context, a human viewer can determine that the two touch points detected by the computer (the brightest points in the image)

are being carried out by two different people—we can clearly see two hands in a configuration which would be very difficult for a single person to achieve given the size of the screen (40"). The primary features of the two touch points that enable us to infer that we are seeing the hands of two separate people are the difference in their orientations and the distance between the touch points. Touch orientation refers to the orientation of the finger's contact area detected by the computer's vision system.

Figure 5. A raw image captured by Microsoft PixelSense on-board cameras showing two different people touching the screen.

In Figure 5, the touch points are concurrent, which helps us determine that we are seeing two different people. If the touch points were not concurrent—for example, if the touch on the left of the screen took place 30 seconds after the touch at the top—we would be less confident that we were seeing two people rather than one person who has taken a few steps to their right in order to reach a different area of the screen. Therefore, a third feature, the time that has passed between the touches, is also necessary to make a prediction that the two touches were carried out by the same person or two different people.

Based on the above examples, the three features included in the model are all needed to make a prediction of "same" or "different" but the exact nature of the relationship between them is not clear and is likely non-linear. Although including just three features makes for a simple model, the non-linearity adds complexity. The MLP classifier has been demonstrated to be well-suited for complex problems where the relative importance of each feature is unknown [17], making it a good choice for this work. Our main requirement for classifier selection was that the model output a probability with each classification, as this was necessary for our grouping algorithm, described below. We selected MLP as our classifier after using leave-one-out nested cross validation in Weka [18] to test a number of classifiers that output a probability. The Weka classifiers tested included Logistic Regression, LibSVM, and BayesNet among many others. Our use of a classifier learned from data, instead of a heuristic approach, also means that future versions of Group Touch could incorporate additional touch features found to be predictive.

The MLP Model, Pre-processing, and Training

The purpose of our MLP model is to predict, given a pair of touches, whether or not those touches are carried out by the same or different people. To prepare a touch log to train our model, each touch was processed sequentially.

Pre-processing of the raw touch logs began by converting individual touches into pairs of touches. Each touch pair consisted of a given touch and the previous touch by each user in the session. The values of each of the features—difference in orientation, distance, and time between touches (see Figure 4)—were calculated to form an *instance* with a class label of either "same" or "different." A single instance comprised the new touch and the last touch carried out by the same user (with the class label, "same"). *N*-1 instances comprised the new touch and the last touch carried out by each of the other users in the group (with the class label, "different"). So in a group of *N* users, each new touch resulted in *N* touch pairs, or instances.

Consider, for example, that several of our user groups comprised four students. In these groups, each new touch resulted in four instances (or touch pairs)—one instance formed from the differences between the new touch and the last touch by the *same* person, and three instances describing the differences between the new touch and the last touch by each of the *other* three people.

Creating touch pairs in the aforementioned manner helped to account for differences in how groups of users interact with any given application, as it always produced a similar distribution of instances with the "same" versus "different" class label for a group of *N* users regardless of the activity or the users' working style. For example, a group of four users will always produce three instances with the class label "different" for every instance with the class label "same." This consistency is important to prevent over-fitting the MLP model to a particular application or group working style (e.g., turn-taking versus parallel interactions).

Before training the model, each feature was normalized to the range [0, 1]. For two of the three features, normalization was straightforward as the possible range of values is inherently limited—the difference in orientation can only fall between 0º and 180º, and the maximum distance between two touch points is determined by the length of the screen diagonal. The elapsed time between touches, however, does not have a clear upper limit. We set the upper limit for time between touches to be the 90th percentile for our full dataset, or about 307 seconds, in order to remove outliers. Instances with a greater time between touches were dropped from the training data. Although this upper limit may seem high at just over five minutes, it was fairly common in our dataset for groups to take breaks from interacting with the screen to discuss something, or for some individual users to refrain from interacting for extended periods of time, therefore leading to lengthy gaps between touches by particular users.

We trained and optimized Group Touch's MLP models in Weka using leave-one-out nested cross validation to prevent overfitting. This method resulted in 17 models—each one trained and optimized on the data from 16 of the study sessions and evaluated on the unseen data from the remaining study session.

For every touch logged, as described above, there were more instances generated with the class label "different" than "same," so the data were heavily weighted towards the "different" label (77.40% of instances across all sessions). To avoid overfitting the models to the majority class label, we applied the SMOTE filter [5] to balance the training data, resulting in an almost 50%-50% split between the two labels. MLP models can potentially be sensitive to the order of training instances, so we applied the Randomize filter to shuffle instances before training.

Touch Grouping Algorithm

After a touch pair is categorized with the "same" or "different" label, the next step is to establish groups of touches that are likely to have been carried out by the same user. The goal of Group Touch is to detect when additional users begin touching the screen and *distinguish* among multiple users as they are concurrently interacting with the computer; it is not intended to *track* or *identify* individual users for the duration of an activity. Key to our approach is that a "group" is not synonymous with a "user"—it is a *group of touches* that belong to a single user, representing a period of sustained interaction. Figure 6 illustrates the algorithm used to place touches into groups.

The first touch creates the first group (Figure 6A). For every subsequent touch, a touch pair is created with the last touch in each group established so far. When the MLP model is queried, it returns the probability that the last touch belonged to the same user. If P(same) is less than a specified threshold (p), a new group is started (Figure 6B); otherwise, the touch is assigned to the group with the highest P(same) (Figure 6C).

The value of threshold p can vary between 0.5 and 1.0. Higher thresholds mean that it is more likely a touch group will be made up of touches carried out by only one user because a more confident "same" prediction is required for a touch to be added to an existing group. However, higher thresholds also lead to new groups being created more often, with each group containing fewer touches and lasting a shorter period of time. In contrast, lower thresholds lead to touch groups that last for longer and contain more touches, but also increase the likelihood that touches by different users will be mistakenly grouped together because a weaker "same" prediction is sufficient to add a touch to an existing group.

To evaluate the grouping algorithm and establish a suitable value for threshold p, we tested it on each session of touch data using a leave-one-out procedure. This testing involved

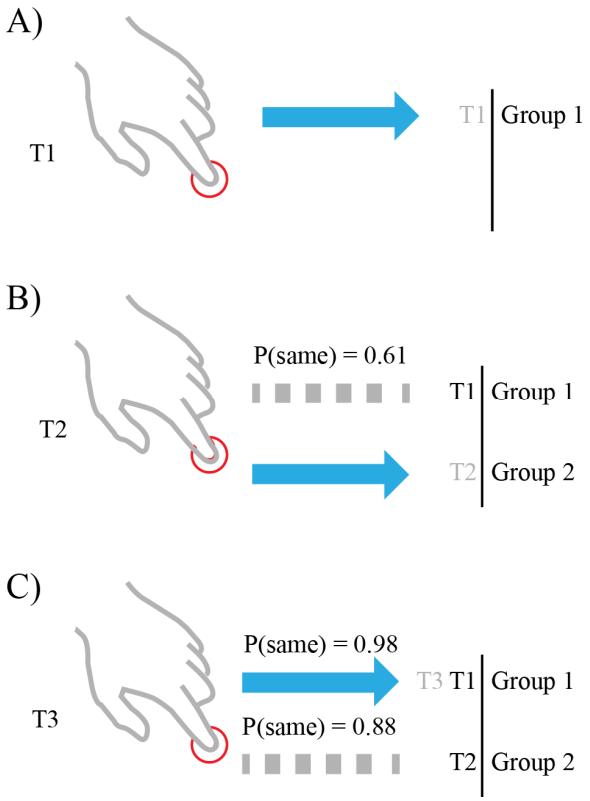

Figure 6. An example of the algorithm to place touches into groups. A) The first touch creates the first group. B) For subsequent touches, we create touch pairs with the last touch of each group, then query the model for each pair. We create a new group if no pairs have a same-author with probability ≥ p, set to 0.80 in this example. C) If multiple touch pairs have a same-author probability ≥ p, we add the touch to the group with the highest probability.

using the algorithm to group the touches in a session according to predictions from the MLP model trained on touch data from the other 16 sessions. As well as being dependent on the performance of the MLP model, the outcome of the grouping algorithm would also be affected by the value of the probability threshold, p. Therefore, we tested the algorithm with a range of probability threshold values from 0.5 to 0.9, in increments of 0.1. This test was repeated with each of the 17 sessions of touch data in order to determine an optimal value of p. We determined that a threshold of 0.80 was suitable for our purposes, a decision that we discuss in more detail in the next section, together with implications of this decision.

Approaches that use touch orientation to match touch points to hands [7,44] have shown thumb touches to be a source of error as the orientation of the thumb can be different from other fingers on the same hand. We anticipated this would also be an issue for Group Touch. Thumb touches are relatively infrequent in general interaction and do not feature in most standard gestures, with the exception of pinch-to-zoom. Thumb touches, however, are heavily used when typing on a virtual keyboard—typically when pressing the spacebar. To remedy this potential problem,

(a) Session - Application	(b) # of Touches	(c) # of Users	MLP Model		(f) Grouping Accuracy (%)
			(d) Accuracy on Test Data (%)	(e) Area under ROC Curve	
1 - Brainstorm	601	5	96.87	0.99	99.44
2 - Heuristics	105	4	89.32	0.96	96.30
3 - Heuristics	328	4	91.49	0.96	95.74
4- Heuristics	241	5	94.29	0.97	98.40
5- Heuristics	240	5	87.81	0.91	89.89
6 - Map	104	5	92.19	0.93	94.74
7 - Map	262	6	91.85	0.94	91.84
8 - Map	519	5	93.85	0.95	91.76
9 - Map	187	5	84.96	0.89	88.97
10 - Resources	285	4	90.50	0.95	94.04
11 - Resources	448	4	89.27	0.95	95.78
12 - Resources	271	5	89.66	0.92	90.10
13 - Resources	296	5	88.28	0.92	89.50
14 - Wireframes	465	5	89.53	0.96	93.07
15 - Wireframes	435	5	89.45	0.96	95.16
16 - Wireframes	241	5	87.54	0.92	83.43
17 - Wireframes	295	4	87.80	0.95	91.49
Mean:	**313.12**	**4.76**	**90.27**	**0.94**	**92.92**
***SD*:**	**138.32**	**0.56**	**2.91**	**0.03**	**3.94**

Table 1. Group Touch results by session. Columns (d) and (e) show how just the MLP model performed on the test data for each session. Column (f) shows the performance of the grouping algorithm with threshold p=0.80. Column (f) is therefore the performance of Group Touch.

groups of touches on a virtual keyboard were merged when the same keyboard instance was touched and the touches occurred within 384 ms of each other. This threshold was chosen based on the typing speeds of participants in a study of typing on virtual keyboards [14]; it represents the time between keystrokes for the slowest participant. Touches to the same keyboard within this time could be assumed to have been carried out by the same user.

Our evaluation of Group Touch was carried out offline but was set up as if it was grouping touches interactively at run-time—touches were processed in the order they were received and logged. Accuracy was calculated as the percentage of touches added to a group where the preceding touch correctly had the same user, using the manual labels assigned from the videos as ground truth.

RESULTS

Group Touch's overall accuracy for a session ranged from 83.43% to 99.44%, with a mean of 92.92% (*SD*=3.94%). However, only 2.66% of all touches logged were incorrectly added to a group of touches by a different user. This discrepancy between the overall accuracy and the number of touches added to incorrect groups occurs because our selection of a p threshold of 0.80 means the algorithm favors creating a new group of touches in the absence of a strong prediction of "same user" for any existing group. Therefore, groups of touches associated with a particular user persist for short periods of time but are highly accurate. Table 1 shows the complete results by session.

To test the first component of Group Touch in isolation, the MLP model that predicts whether a pair of touches was authored by the same person or different people, we used leave-one-out cross-validation to evaluate the model, withholding a different session's touch data each time. Within the training data for each fold in the leave-one-out cross validation, we conducted a 10-fold cross validation of the MLP model. These scores were highly consistent across sessions—the mean cross validation score was 89.22% (*SD*=0.29%). When the trained models were then tested on the withheld touch data, the mean accuracy was 90.27% (*SD*=2.91%). Leave-one-out test results for the MLP model are shown in Table 1(c). Figure 7 shows the ROC curves.

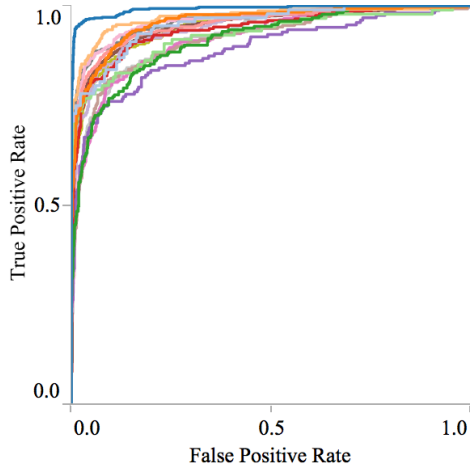

Figure 7. The ROC curves for each MLP model. The area under the curve, listed in Table 1, ranged from 0.89 to 0.99.

The second component of Group Touch, its grouping algorithm, adds a touch to the group for which the MLP model has returned a prediction of "same" person with the highest probability above a threshold p (see Figure 6). We tested threshold values between 0.5 and 0.9 in 0.1 increments and found that mean accuracy increased from 88.83% to

94.23% (Figure 8). At the same time, the median duration of a group—the time elapsed from the first touch in a group to the last—decreased from 7,207 ms to 2,383 ms. Therefore, Group Touch can distinguish among simultaneous users for longer periods of time when the value of *p* is lower but with a tradeoff of lower accuracy within touch groups.

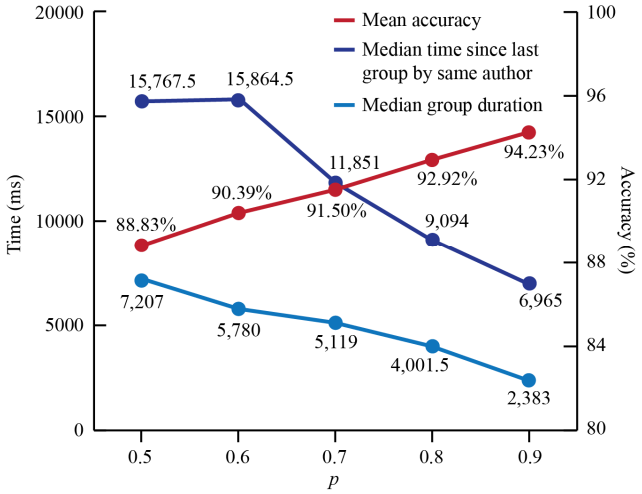

Figure 8. As the value of *p* increases, the accuracy of Group Touch increases. However, the median time since the last group of touches with the same user and the median duration of groups decreases.

To evaluate the threshold values, we also tracked another measure: the time between groups of touches *belonging to the same person* (i.e., the time that elapsed since that person last touched the screen). For example, Figure 8 shows that when p=0.8, the median time since the last group by the same user was 9,094 ms. This means that when a new group is started with a touch by user X, a median of 9,094 ms have passed since X last touched the screen. When the time between groups of touches belonging to the same person is brief, the system is being conservative in assigning new touches to existing groups. It therefore becomes more likely that sustained input by a single person will be ascribed to multiple people. Figure 8 shows that, as the value of p increased, the median time between groups of touches belonging to the same person decreased. Therefore, although accuracy within groups will be highest when p=0.9, longer sequences of related gestures by a single user are more likely to be split into multiple groups than when p is lower.

Based on Figure 8, we chose 0.80 as the value of p. Although the average accuracy increases to 94.23% when p=0.90, both time metrics decrease to a point where Group Touch would only be reliably able to distinguish among users for brief interactions. At p=0.90, Group Touch still has valuable use cases (see the Discussion) but at p=0.80, Group Touch retains high accuracy and is able to distinguish among simultaneous users over longer periods of time, making it useful for a wider range of applications. When p was set to 0.80, the midspread (middle 50%) of groups lasted between 0.65 and 12.02 seconds and contained 2 to 6 touches. The longest group recorded lasted 77.10 seconds and contained 80 touches. Table 1(d) shows the results of the evaluation of the grouping algorithm with the threshold, p, set to 0.80.

To understand why Group Touch creates new groups for touches by an existing user, we investigated the differences between touches that resulted in a new group and those that were added to an existing group. Figure 9 shows the average features of touch pairs by the MLP model's probability that a pair was carried out by the same user, P(same). The "New groups created" region shows the average features of false negatives—touch pairs that were actually by the same user (based on the labels assigned manually) but were incorrectly labeled by Group Touch as being by different users, leading to the creation of a new group. For these touch pairs, the MLP model returned a probability, P(same), below our threshold of 0.80 (see Figure 6B). Moreover, 62.41% of false negatives occurred when the MLP model returned P(same) < 0.1, a strong prediction that the touches were carried out by different people. Clear trends emerged for each feature as P(same) increased. Typically, touch pairs with the lowest values of P(same) had large values for time between touches (M=162.54 s, SD=76.70 s). Difference in orientation (M=69.86°, SD=47.14°) and distance between touch points (M=601.37 px, SD=347.47 px) also tended to be higher but there was more variation for these features. These results suggest that extended time between touches results in the creation of a new group when an existing group of touches by the same user is available.

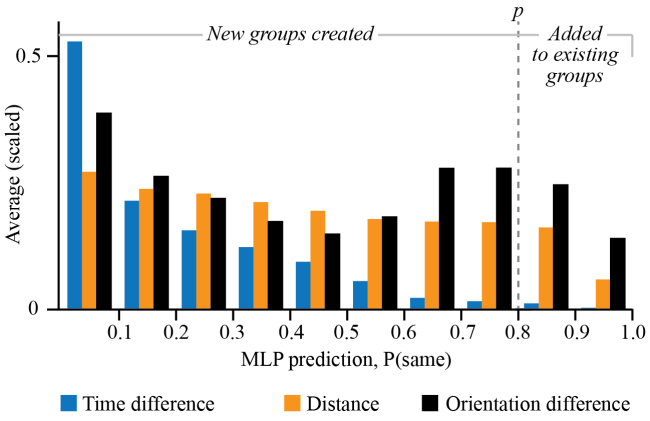

Figure 9. Average features of touch pairs by the probability they were carried out by the same user according to the MLP model.

For comparison, Figure 9, "Added to existing groups," shows the average features of touch pairs that resulted in a touch being added to an existing group. These are pairs where the MLP model returned the highest value of P(same) above our 0.80 threshold (see Figure 6C). 93.58% of these touch pairs were true positives that were correctly assigned—both touches were carried out by the same user. 91.16% of these touch pairs had a P(same) of at least 0.90. The time between touches for pairs with a P(same) of at least 0.90 was brief (M=1.03 s, SD=1.54 s). Difference in orientation (M=25.25°, SD=24.75°) and distance between

touches (M=130.35 px, SD=130.79 px) were also at their lowest when P(same) was at least 0.90. This means that a touch was more likely to be correctly added to an existing group when it was close to the last touch in the group in terms of all three features.

DISCUSSION

Group Touch's overall accuracy (Table 1(d)) shows that it was successful at grouping touches that belonged to the same user. There was little variability in the accuracy of both the MLP model (Table 1(c)) and the grouping algorithm (Table 1(d)) across the 17 sessions of touch data collected in five different applications. These results indicate that Group Touch is user-independent and can perform consistently across a range of applications.

Group Touch favors creating a new group of touches when the MLP model does not return a strong prediction of "same user" for any touch pair. This means that group assignments are highly accurate, with only 2.66% of all touches in our dataset being added incorrectly to a group by another user. However, this also means that it is important to understand when and why the model is unable to return a strong prediction of "same user," leading to the creation of new groups. The trends shown in Figure 9 suggest that extended time between touches is a strong reason that new groups are created when an existing group of touches by the same user is available. Therefore, Group Touch is likely to perform better and create groups that persist for longer when users are regularly interacting with the screen, so time between touches is brief. Given that Figure 9 also shows that touches are correctly added to groups when the time and distance between touches are low, Group Touch is probably not suited for applications where users frequently jump around different areas of the screen. This might explain why session 16 had a much lower accuracy than all other sessions at 83.46%, 6.46% lower than the next lowest result. The students in this session spent much of the time flicking the on-screen objects at other objects to make them bounce around the edges, then reaching across the full length of the screen to retrieve the object.

The value of the threshold p used to assign touches to groups of touches by the same user impacted the group duration and time between groups by the same user, as well as the accuracy. We selected 0.80 as our value of p because it resulted in a high grouping accuracy and the decrease in group duration was much steeper with a p above 0.80. However, the value of p and the resulting tradeoff between group duration and accuracy affects the potential use cases for Group Touch and therefore should be adjusted depending on how Group Touch is to be used.

The simplest and most generalizable use case for Group Touch is resolving conflicting input that occurs when multiple people are simultaneously interacting with the same interface. For example, without the ability to distinguish among users, the touch input of multiple people attempting to drag an object in different directions will look very much like that of a single user carrying out a pinch-to-zoom action. Another example would be a user attempting to close a window that another user is actively working in [31]. Group Touch makes it possible to resolve these and other conflicts and determine how the application should respond because it is not necessary to know the identity of the users, just whether or not multiple users are interacting with the application concurrently. In this scenario, Group Touch would affect the application's behavior in ways that are visible to users. Therefore, the high accuracy of the top value of p, 0.90, would likely be favored over groups that persist for longer, particularly as conflicting gestures will occur with a small number of overlapping touches.

In addition to resolving conflicting gestures, there are numerous application scenarios where the ability to detect when multiple people are attempting to interact with the same object would be useful. For example, in a collaborative document editing application, it would be useful to prevent concurrent interaction in some situations (e.g., one user scrolling a page while another is making edits). Online collaboration tools, such as Google Docs, handle this situation by allowing each user to control navigation of their own view of the document—they can see others' edits as they occur but they can move around their own view of the document independently. This approach would not transfer to a similar application on a shared screen. Group Touch could help in this scenario by making it possible to temporarily block interaction from users other than the person actively working in the document. Additionally, tracking concurrent versus individual edits to a document could make document revision history more useful.

The ability to temporarily block other users from interacting with an object that a user is currently holding could also be used to enforce turn-taking in a collaborative learning environment or as a game mechanic in a competitive game where players have to steal objects from their opponents. Taking the opposite approach to the same incident—only responding to input if multiple people are simultaneously interacting with an object—would make it possible to support cooperative gestures. Cooperative gestures have been shown to be useful for increasing awareness of important functionality and increasing participation in applications for collaborative work and play [31, 32].

Group Touch could also be used to model the collaborative processes of small groups working at tabletop computers. For example, the ability to distinguish when multiple people are interacting with an application makes it possible to detect when users are taking turns, interfering with the actions of others, or working in parallel, all of which have been shown to impact collaborative learning [12,15,20,26,33]. For this use case, it would be important to capture longer interactions that contain more than just a small number of isolated gestures. Therefore, a lower p threshold, which creates groups that persist for longer would be preferable to increasing accuracy.

Application designers could also use information about when and where conflicting gestures occur, and how groups use their applications, to evaluate and refine their designs to enable better collaborative use. For example, Group Touch would make it possible to automatically identify areas of the screen where there is typically only one user interacting at a time and those where there tend to be multiple users interacting. This information could be used to determine placement of shared interface elements and how to divide the screen into individual and shared workspaces.

Group Touch offers an enhancement to existing touch-detection on large, multi-user touchscreens that adds a new capability without breaking existing touch functionality when errors occur. For example, Group Touch false positives may cause an application to interpret the touches of multiple users as a single gesture. This outcome is typical of most current commercial tabletops. Therefore, when Group Touch produces a false positive, behavior will be the same as in current systems.

In the case of false negatives, the primary causes are extended time between touches, or touches close together in time that have very different orientations or locations. How false negatives impact a user will largely depend on what an application does with information provided by Group Touch. For example, when detecting and resolving conflicting gestures, false negatives due to touches that are far apart in time are unlikely to produce a conflict in the interface and therefore unlikely to affect the user. In some cases, however, a false negative could cause a zoom gesture to look like two separate pan gestures. These cases occur infrequently and application-level design could help to mitigate any impact.

Limitations

The main limitation of Group Touch is that it does not track or identify users for the duration of an activity, and therefore cannot enable personalization, such as color-coding each user's touches [43], or tracking individual contributions to the group effort [29]. It also cannot be used for authentication [4,24]. However, existing approaches to distinguishing users that are able to provide these features either rely on external sensors or artificially constraining interaction. As stated, Group Touch is intended for situations where such methods are impractical or undesirable.

Additionally, the applications we used to design and evaluate Group Touch all featured highly collaborative tasks in which group members worked with shared objects. We expect accuracy would be about the same for other tasks of this nature. However, we expect that Group Touch may have lower accuracy in situations where individual users frequently interact with objects in very different areas of the screen (e.g., rapidly reaching from the top left corner to the bottom right). Based on our analysis of touch group characteristics (Figure 9), these types of interactions would look like touches carried out by different people. In applications requiring this functionality, Group Touch may not be the best option for distinguishing among users.

Finally, in this work we have not yet identified the exact situations where Group Touch performs poorly. For example, we have not investigated how good Group Touch is at recognizing two-handed input by a single person versus single-handed input. This limitation is because all of our touch data were collected "in the wild" in uncontrolled field settings and extracting these types of specific interactions would be an extremely laborious process. A controlled lab study would likely be necessary to gain this kind of insight.

FUTURE WORK

One avenue for future work would be to address some of the limitations of Group Touch by testing the approach with data collected in less collaborative applications and to conduct a controlled study that can identify the exact types of low-level input that Group Touch is best able to handle.

Additionally, it would be worthwhile to investigate ways to improve Group Touch by combining it with other approaches that focus on finger and hand detection (e.g. [7,44]), which may help make Group Touch even more robust.

In our future work, we intend to use Group Touch to model the processes of collaborative learning with tabletop computers in high school classroom settings using the touch patterns described by Evans et al. [12]. To date, identifying indicators of the quality of collaborative learning processes has required manually labeling touch data using video recordings. Group Touch will make it possible to detect and respond to these touch patterns in real-time.

CONCLUSION

We have presented Group Touch, an approach to distinguishing among multiple simultaneous users by detecting when the users interacting with a vision-based tabletop computer change. Our approach achieved a mean accuracy of 92.92% (*SD*=3.94%). Group Touch is the first approach to distinguishing users that was designed and evaluated using data collected entirely "in the wild." It uses only the built-in capabilities of the tabletop hardware and does not require the use of extra sensors, making it more flexible and easier to deploy than many existing approaches.

Group Touch is not intended to replace existing approaches to distinguishing users, nor is it a suitable approach if individual users need to be identified and tracked for the duration of a session. Instead, it is intended to bring the capability of distinguishing among concurrent users to new settings that other approaches cannot support, namely, settings where use of external sensors is impractical and artificially constraining interaction is undesirable.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation under awards IIS-0952786 and IIS-1053868. Any opinions, findings, conclusions or recommendations expressed in this work are those of the authors and do not necessarily reflect those of the National Science Foundation.

REFERENCES

1. Christopher James Ackad, Andrew Clayphan, Roberto Martinez Maldonado, and Judy Kay. 2012. Seamless and continuous user identification for interactive tabletops using personal device handshaking and body tracking. *Extended Abstracts on Human Factors in Computing Systems (CHI EA '12)*, 1775–1780.
2. Michelle Annett, Tovi Grossman, Daniel Wigdor, and George Fitzmaurice. 2011. Medusa: a proximity-aware multi-touch tabletop. *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '11)*, 337–346.
3. Katerine Bielaczyc. 2009. Designing social infrastructure: critical issues in creating learning environments with technology. *Journal of the Learning Sciences* 15, 3: 301–329. http://doi.org/10.1207/s15327809jls1503
4. Bojan Blazica, Daniel Vladusic, and Dunja Mladenic. 2013. MTi: a method for user identification for multitouch displays. *International Journal of Human-Computer Studies* 71: 691–702. http://doi.org/10.1016/j.ijhcs.2013.03.002
5. Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2011. SMOTE: Synthetic minority over-sampling technique. *Journal Of Artificial Intelligence Research* 16: 321–357.
6. Andrew Clayphan, Roberto Martinez-Maldonado, Christopher Ackad, and Judy Kay. 2013. An approach for designing and evaluating a plug-in vision-based tabletop touch identification system. *Proceedings of the Australian Computer-Human Interaction Conference (OzCHI '13)*, 373–382. http://doi.org/10.1145/2541016.2541019
7. Chi Tai Dang, Martin Straub, and Elisabeth André. 2009. Hand distinction for multi-touch tabletop interaction. *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS '09)*, 101–108. http://doi.org/10.1145/1731903.1731925
8. Paul Dietz and Darren Leigh. 2001. DiamondTouch: a multi-user touch technology. *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '01)*, 219–226. http://doi.org/10.1145/502348.502389
9. Pierre Dillenbourg and Michael Evans. 2011. Interactive tabletops in education. *International Journal of Computer-Supported Collaborative Learning* 6, 4: 491–514.
10. Pierre Dillenbourg and Patrick Jermann. 2010. Technology for classroom orchestration. In *New Science of Learning: Cognition, Computers and Collaboration in Education*, M. S. Khine and I. M. Saleh (eds.). Springer, New York, NY, 525–552.
11. Pierre Dillenbourg, Guillaume Zufferey, Hamed Alavi, Patrick Jermann, Son Do-lenh, and Quentin Bonnard. 2011. Classroom Orchestration : The Third Circle of Usability Why is Paper Highly Usable in Classrooms ? *Proceedings of the International Conference on Computer-Supported Collaborative Learning (CSCL '11)*, 510–517.
12. Abigail C. Evans, Jacob O. Wobbrock, and Katie Davis. 2016. Modelling collaboration patterns on an interactive tabletop in a classroom setting. *Proceedings of the ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW '16)*, 860–871.
13. Philipp Ewerling, Alexander Kulik, and Bernd Froelich. 2012. Finger and hand detection for multi-touch interfaces based on maximally stable extremal regions. *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS '12)*, 173–182.
14. Leah Findlater, Jacob O Wobbrock, and Daniel Wigdor. 2011. Typing on flat glass: examining ten-finger expert typing patterns on touch surfaces. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*, 2453–2462. http://doi.org/10.1145/1978942.1979301
15. Rowanne Fleck, Yvonne Rogers, Nicola Yuill, et al. 2009. Actions speak loudly with words: unpacking collaboration around the table. *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS '09)*, 189–196. http://doi.org/10.1145/1731903.1731939
16. Fernando Garcia-sanjuan, Javier Jaen, and Alejandro Catala. 2013. Evaluating heuristics for tabletop user segmentation based on simultaneous interaction. *Expert Systems With Applications* 40, 14: 5578–5587. http://doi.org/10.1016/j.eswa.2013.04.011
17. M. W. Gardner and S. R. Dorling. 1998. Artificial Neural Networks (The Multilayer Perceptron)—A Review of Applications in the Atmospheric Sciences. *Atmospheric Environment* 32, 14/15: 2627–2636.
18. Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA data mining software: an update. *SIGKDD Explorations* 11, 1: 10–18.
19. Jefferson Y Han. 2005. Low-cost multitouch sensing through frustrated total internal reflection. *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '05)*, 115–118.
20. Amanda Harris, Jochen Rick, Victoria Bonnett, et al. 2009. Around the table: Are multiple-touch surfaces better than single-touch for children's collaborative interactions? *International Conference on Computer-Supported Collaborative Learning (CSCL '09)*, 335–344.

21. Chris Harrison, Munehiko Sato, and Ivan Poupyrev. 2012. Capacitive fingerprinting: exploring user differentiation by sensing electrical properties of the human body. *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '12)*, 537–543.
22. Steven E Higgins, Emma Mercier, Elizabeth Burd, and Andrew Hatch. 2011. Multi-touch tables and the relationship with collaborative classroom pedagogies: a synthetic review. *International Journal of Computer-Supported Collaborative Learning* 6, 4: 515–538.
23. Uta Hinrichs and Sheelagh Carpendale. 2011. Gestures in the wild: studying multi-touch gesture sequences on interactive tabletop exhibits. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*, 3023–3032.
24. Christian Holz and Patrick Baudisch. 2013. Fiberio: a touchscreen that senses fingerprints. *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '13)*, 41–50. http://doi.org/10.1145/2501988.2502021
25. Paul Jermann, Guillaume Zufferey, and Pierre Dillenbourg. 2008. Tinkering or sketching: apprentices' use of tangibles and drawings to solve design problems. *Proceedings of the European conference on Technology Enhanced Learning (EC-TEL '08)*, 167–178.
26. Ahmed Kharrufa, Madeline Balaam, Phil Heslop, David Leat, Paul Dolan, and Patrick Olivier. 2013. Tables in the wild: lessons learned from a large-scale multi-tabletop deployment. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*, 1021–1030. http://doi.org/10.1145/2470654.2466130
27. Daniel Klinkhammer, Markus Nitsche, Marcus Specht, and Harald Reiterer. 2011. Adaptive personal territories for co-located tabletop interaction in a museum setting. *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS '11)*, 107–110. http://doi.org/10.1145/2076354.2076375
28. Nicolai Marquardt, Johannes Kiemer, David Ledo, Sebastian Boring, and Saul Greenberg. 2011. Designing user-, hand-, and handpart-aware tabletop interactions with the TouchID toolkit. *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS '11)*, 21–30. http://doi.org/10.1145/2076354.2076358
29. Roberto Martinez, Anthony Collins, Judy Kay, and Kalina Yacef. 2011. Who did what? Who said that?: Collaid: an environment for capturing traces of collaborative learning at the tabletop. *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS '11)*, 172–181. http://doi.org/10.1145/2076354.2076387
30. Tobias Meyer and Dominik Schmidt. 2010. IdWristbands: IR-based user identification on multi-touch surfaces. *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS '10)*, 277–278. http://doi.org/10.1145/1936652.1936714
31. Meredith Ringel Morris, Anqi Huang, Andreas Paepcke, and Terry Winograd. 2006. Cooperative gestures: multi-user gestural interactions for co-located groupware. *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '06)*, 1201–1210. http://doi.org/10.1145/1124772.1124952
32. Meredith Ringel Morris, Kathy Ryall, Chia Shen, Clifton Forlines, and Frederic Vernier. 2004. Beyond "social protocols": multi-user coordination policies for co-located groupware. *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, 1–4. http://doi.org/10.1145/1031607.1031648
33. Taciana Pontual Falcão and Sara Price. 2010. Interfering and resolving: how tabletop interaction facilitates co-construction of argumentative knowledge. *International Journal of Computer-Supported Collaborative Learning* 6, 4: 539–559. http://doi.org/10.1007/s11412-010-9101-9
34. Raf Ramakers, Davy Vanacken, Kris Luyten, Karin Coninx, and Johannes Schoning. 2012. Carpus: a non-intrusive user identification technique for interactive surfaces. *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '12)*, 35–44.
35. Stephan Richter, Christian Holz, and Patrick Baudisch. 2012. Bootstrapper: recognizing tabletop users by their shoes. *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, 1249–1252. http://doi.org/10.1145/2207676.2208577
36. Volker Roth, Philipp Schmidt, and G Benjamin. 2010. The IR ring: authenticating users ' touches on a multi-touch display. *Proceedings of the ACM symposium on User Interface Software and Technology (UIST '10)*, 259–262.
37. Kathy Ryall, Clifton Forlines, Chia Shen, and Meredith Ringel Morris. 2004. Exploring the effects of group size and table size on interactions with tabletop shared-display groupware. *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW '04)*, ACM Press, 284–293.
38. Johannes Schöning, Jonathan Hook, Nima Motamedi, et al. 2009. Building interactive multi-touch surfaces. *Journal of Graphics, GPU, and Game Tools* 14, 3: 35–55.
39. Stacey D. Scott, Carpendale Sheelagh, and Kori. M. Inkpen. 2004. Territoriality in collaborative tabletop workspaces. *Proceedings of the ACM Conference on*

Computer Supported Cooperative Work (CSCW '04), 294–303. http://doi.org/10.1145/1031607.1031655

40. Masanori Sugimoto, Kazuhiro Hosoi, and Hiromichi Hashizume. 2004. Caretta: a system for supporting face-to-face collaboration by integrating personal and shared spaces. *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '04)*, 41–48. http://doi.org/10.1145/985692.985698
41. Cristian Tanase, Radu-daniel Vatavu, Ş Pentiuc, and Adrian Graur. 2008. Detecting and tracking multiple users in the proximity of interactive tabletops. *Advances in Electrical and Computer Engineering* 8, 2: 50–53.
42. Feng Wang, Xiang Cao, Xiangshi Ren, and Pourang Irani. 2009. Detecting and leveraging finger orientation for interaction with direct-touch surfaces. *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '09)*, 23–32. http://doi.org/10.1145/1622176.1622182
43. Hong Zhang, Xing-Dong Yang, Barrett Ens, Hai-Ning Liang, Pierre Boulanger, and Pourang Irani. 2012. See me, see you: a lightweight method for discriminating user touches on tabletop displays. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*, 2327–2336. http://doi.org/10.1145/2207676.2208392
44. Zhensong Zhang, Fengjun Zhang, and Hui Chen. 2014. Left and right hand distinction for multi-touch tabletop interactions. *Proceedings of the 19th international conference on Intelligent User Interfaces (IUI '14)*, 47–56.