# CircGR: Interactive Multi-Touch Gesture Recognition using Circular Measurements

**Ruben Balcazar**
rbalc001@fiu.edu

**Francisco R. Ortega**
fortega@cs.fiu.edu

**Katherine Tarre**
ktarr007@fiu.edu

**Armando Barreto**
barratoa@fiu.edu

**Mark Weiss**
weiss@cs.fiu.edu

**Naphtali D. Rishe**
ndr@acm.org

School of Computing and Information Sciences
Florida International University
Miami, FL 33199

## ABSTRACT

CircGR is a multi-touch non-symbolic gesture recognition algorithm, which utilizes circular statistic measures to implement linearithmic ($\mathrm{O}(n \log n)$) template-based matching. CircGR provides a solution to gesture designers, which allows for building complex multi-touch gestures with high-confidence accuracy. We demonstrated the algorithm and described a user study with 60 subjects and over 12,000 gestures collected for an original gesture set of 36. The accuracy is over 99% with the Matthews correlation coefficient of 0.95. In addition, early gesture detection was successful in CircGR as well.

### Author Keywords

multi-touch; gesture recognition; template matching

### ACM Classification Keywords

H.5.2. User Interfaces: Input Devices and Strategies

## INTRODUCTION

Multi-touch has become pervasive since the introduction of the iPhone in 2007. In the past few years, more notebooks (e.g., Microsoft Surface) and tablets (e.g., iPad) have begun to take off with multi-touch as one of the input modalities. Users have already internalized some multi-touch gestures [24]. However, when going beyond the simple pinch, swipe, and rotate gestures, there is still a need for a generalized (non ad-hoc) interactive gesture recognizer that is responsive and easy to implement. We provide a solution, named CircGR.

CircGR is an interactive template-based recognition algorithm, which uses circular design primarily for non-symbolic gestures (e.g., rotate is a non-symbolic gesture, versus the letter A). Our algorithm, tested with 60 subjects with a set of 36 gestures (for a total of 12960 gestures collected), yielded an accuracy (ACC) of 99% and Matthews correlation coefficient (MCC) of 0.95. One of the main features of our algorithm is the ability for early detection by classifying while the gestures are in progress.


*ISS'17*, October 17–20, 2017, Brighton, United Kingdom.

https://doi.org/10.1145/3132272.3134139

### Motivation: Yet Another Multi-Touch Classifier?

The ubiquitous state of multi-touch surfaces may prompt the false assumption that multi-touch gesture recognition is a solved problem; however, challenges still exist. One of the challenges is incorporating early detection of multi-touch gesture with interactive systems and detection using template-based matching. One of the advantages of template-based matching is that it doesn't require large training sets (one template per gesture is sufficient). For a system that requires multiple gestures (without using ad-hoc methods), a highly responsive interactive recognizer with high-accuracy is needed. Our proposed method addresses this particular research problem.

When working towards an interactive multi-touch classifier, we tried different methods, including the extension of known recognizers (listed in the Related Work Section), and we started looking at Circular Statistics. While we didn't use the typical circular statistical models, we looked at the methods used in circular statistics, and noticed that we could represent the gestures through circular measurements. One of the reason for this occurrence is the angularity of some gestures.

### Contributions

CircGR is a novel way to recognize non-symbolic gestures for interactive (i.e., real-time: the output of the overall sequence of processes appears to be generated without any appreciable delay with respect to the overall input) systems using circular measures as a way to represent gesture templates. The contributions detailed in this paper include: (1) demonstration of a linearithmic algorithm for multi-touch detection with a high recognition accuracy (ACC = 99% and MCC = 0.95); (2) demonstration of early-detection of gestures (without the need for gesture completion) with ACC=99% and MCC = 86% for the first window (64 points); (3) demonstration of the need for only one user or developer-provided

training sample per template; and (4) a 60-subject experiment with over 12,000 gestures collected for a gesture set of 36. The novelty of CircGR is the simplicity seen in previous Dollar ($) family algorithms, coupled with an improved running time and the ability to recognize a wide variety of non-symbolic gestures for multi-touch displays. As with the $ Family, CircGR provides the ability of gesture designers to create gestures by example and quickly prototype gesture sets.

Note that CircGR can run in linear time if sorting is not used during the classification, with minimal decrease in accuracy. Finally, we provide information on how to obtain our source code and data repository in our supplemental material (see `http://CircGr.com`), as well as additional algorithm listings. The algorithms are listed in the appendix, part of the supplementary material.

## RELATED WORK

Gesture recognition approaches fall into two main categories: formal gesture definition languages and specification via examples [9]. Formal gesture definition languages define gestures for specific domains [18], using dialects such as XML [1, 23], JSON [12], regular expressions [20, 19], logical rules [30, 27, 15], and semiotics [17]. Gesture recognition proceeds from formal languages by verifying that input matches declared definitions. Formal languages allow broad expression of potentially unlimited types of gestures, but suffer from either sensitivity or specificity issues. A gesture outside of a definition will not be considered as part of that definition regardless of how close it is, which can be problematic when the expression of a certain gesture varies by user (an issue of sensitivity). This definition can be "loosened" to include variation, but then might incorrectly include other gestures (an issue of specificity). To overcome these hurdles and ensure good classification, the burden of defining a mutually exclusive gesture set (i.e. each gesture is uniquely different from the others) as well as creating gesture definitions that strike the right balance of sensitivity and specificity, falls on the developer. Furthermore, there is no inherent priority or measure of rank when an input gesture meets the definition of multiple gestures. Ad-hoc solutions are usually offered by each system to address this limitation.

For example, gesture representation in Proton [20] and Proton++ [19] revolve around a stream of gesture events encoded in the form of a string of symbols consisting of three properties: the type of action, the identification of the trace, and the trace's target object. The set of all symbol permutations forms a language, and each gesture is then a regular expression denoting a subset of that language, which allows the input to be classified by testing whether it is a part of that subset. Ambiguities must be resolved by either refining the definition or via an ad-hoc score function – again – placing the burden on developers to ensure proper functionality. The work by Proton and Proton++ presents an innovative avenue in gesture recognition and specification via regular expressions, but we believe that this approach increments the complexity of gesture set design.

Specification via examples involves building classifiers from a training set of gestures. Often, these methods involve feature extraction to reduce the gesture to a specific feature space or derive a statistical model capable of classification. Popular methods include Hidden Markov Models (HMM) [11, 31], statistical approaches [29, 28], finite state machine (FSM) generation [14, 21], dynamic programing [22, 37], Petri nets [32], Bayesian networks [35], fuzzy logic [8], Neural Networks (NN) [5, 25], and nearest-neighbor approaches [36, 4, 3, 34]. Specification approaches are most commonly limited by the need to collect training data. Classifiers based on statistical models that require training (e.g., HMM,FSM, NN, Fuzzy Logic) require numerous examples of each gesture. HMM-based model by Dittmar et al. achieved good performance utilizing 120 examples over 6 gestures to train the model [11]. Furthermore, a statistical approach such as Rubine's classifier [28], or Schmidt and Weber's template matcher [29] require feature engineering, which ultimately has trouble separating gestures across features that are not linearly separable. Bayesian networks, fuzzy logic, and Petri nets also require a degree of expert knowledge that make such systems suitable in the domain of that expert knowledge. Nearest-neighbor is simple to understand and implement but also requires quantization via features as it is often unfeasible to utilize the full breadth of gesture data as input.

Template-based approaches employ nearest-neighbor-like methods to classify input as the "closest" template via a distance metric. The $-family are particularly notable template based matchers, supporting multiple strokes and featuring rotational invariance without the need for large datasets or very specific domain knowledge [36, 4, 3, 34]. While the $-family rely on distances between re-sampled and superimposed gestures as a measure of similarity, a very direct nearest-neighbor approach, their approach is focused on supporting single or multi-stroke interaction rather than our approach, that focuses on multi-touch interaction. Other template matchers utilize a combination of statistical models to define the distance metric [29, 28] as a hybrid approach. Template matchers often require additional preprocessing of input to minimize template-input gesture differences due to scale and rotation, incurring additional processing time.

Regardless of approach, multi-touch gestures themselves can be of symbolic or non-symbolic nature. Symbolic gestures include the articulation of alphanumeric characters or symbols (e.g., $\leftarrow$, $\theta$, $\gg$) with a meaning, while non-symbolic gestures express movement and action (e.g., swipe, pinch, double-tapping). The type of gesture involved affects both domain and recognition. Symbolic gestures are suitable as activators to some functionality (e.g., drawing "S" to save a document) therefore, recognition occurs after the execution of the gesture. Non-symbolic gestures have no predefined end-point, and are more suited for interactive or manipulative tasks (e.g., using pinch to zoom) which must occur during gesture execution with recognition occurring in parallel.

CircGR is a non-symbolic template based matcher that represents gestures as a set of circular measures generated from touch data. Input is converted to circular measures and recog-

nition is performed on these measures as the gesture is being performed. Circular measures are descriptive statistics utilized in directional statistics. Circular statistics are involved in the analysis of circular data such as direction, time, and their distributions. Circular data usage in gesture recognition has largely been focused on HMM or probabilistic models using the von Mises distribution in domains such as sign language recognition [33], hand drawn gestures [6], and action recognition [7].

## CIRCULAR MEASURES

Circular statistics is primarily focused on data that represents time, direction, and orientation. These measures have no particular sense of "magnitude". Therefore, they can be represented by unit vectors. A set of such vectors in 2-dimensional space lie on the surface of a unit circle and are thus circular data. Further information on circular data and its statistical usage can be found in [16]. Below we denote the basic measures that are in use by CircGR.

A circular observation $\alpha$ consists of a polar angle $\alpha \in [0, 2\pi)$ which represents a 2-dimensional unit vector. In other words, it is a polar coordinate $(1, \alpha)$. It is trivial to convert such a vector between its polar angle representation and its rectangular coordinates. A set of angles $A = \alpha_1, \alpha_2, \ldots, \alpha_N$ has a mean resultant vector $R$ represented by polar coordinate $(||R||, \alpha_R)$, where $||R|| = \sqrt{C^2 + S^2}$ and $\alpha_R = atan2(S, C)$. The values $S$ and $C$ are the sum of the sine of all angles and the cosine of all angles, respectively. The circular distance $d$ between two observations $\alpha$ and $\beta$, is the smallest arc length between them and can be directly calculated via equation $d_C(\alpha, \beta) = \pi - |\pi - |\alpha - \beta||$.

### Why Circular Measures?

Template matching algorithms typically preprocess input to maximize classification performance. Preprocessing can include resampling, scaling, rotation, and translations of input, as well as additional technique-specific transformations. These are usually applied in a sequential manner, which means that points are iterated over multiple times. Modern improvements in template matchers often come in the form of additional preprocessing steps. For example, Rekik et al. introduced Match-Up, a simple clustering preprocessing technique with two steps that is invariant to how users articulate multi-touch gestures [26]. Match-Up, paired with $P [34], significantly increased recognition over several gestures when articulated with multiple fingers compared to $P. However, each preprocessing step added to the algorithm produces more runtime overhead, leading to slower classifications.

Circular measures are just a series of angles along with their descriptive statistics (e.g., mean, resultant vector). A series of angles can be used to represent any given geometric curve by segmenting the curve into equal-length vectors [13] and such representation has been used as an input technique for magnetized styluses [2]. In CircGR, input traces are converted to series of angles to be used explicitly for recognition, which has several important consequences: (1) because a trace is just a series of angles in this representation, there is no need for translation or scaling, (2) rotation can easily be done by increasing each angle by the desired rotation angle in the moment that they are calculated (however, as we are not aiming to be rotation invariant, this is not done), (3) resampling, which is very common in template-matching approaches to make input comparable to stored templates, naturally segments an input trace into a series of equal-length vectors, meaning that this preprocessing step is all that is needed to generate the angles, and (4) each angle is treated by CircGR as a polar angle to a unit vector, which makes distance calculation very simple arithmetic. Furthermore, using this circular framework lends itself to additional notions such as representing temporal sequence of gesture events over the input interval where each event is a label that is easily derived from the angles themselves during resampling. These labels facilitate classification without additional preprocessing.

## CircGR: GESTURE REPRESENTATION

CircGR uses standard definition of a multi-touch gestures; they begin when the first finger contacts the screen and end when the last finger ceases contact with the screen. Each finger produces a series of 2D points called a *trace*. Gestures can have an arbitrary amount of traces. Traces belonging to fingers that are not moving during the gestures are referred to as *anchors*. CircGR detects anchors automatically and creates a single point, the centroid of all the points in that trace, to represent that anchor. From this point on, we will use the word "traces" to refer to traces that are explicitly not anchors.

CircGR defines the *center point*, $p_C$ of the gesture depending on the presence of anchors. If there are anchors present, the center of the gesture is the centroid of all anchors. If there are no anchors present, then the center is the centroid of the first point in each trace. Anchors, by extension, also represent traces with very few points, such as taps; hence a two finger tap can be expressed as a gesture with two anchors and no traces.

A CircGR gesture is two sets of $n$ angles $S = \{\alpha_i^S | i = 1..n\}$ and $T = \{\alpha_i^T | i = 1..n\}$ that represent spatial and temporal properties, respectively. The angles are derived from input traces consisting of a set of points $P = \{p_i = (x_p^i, y_p^i, id_p^i, t_p^i) | i = 1..N\}$, where $id$ is stroke ID, and $t$ is the timestamp. Additional data includes the number of anchors and traces, which add up to number of fingers.

The process of calculating the spatial angles of $S$ is illustrated in Figure 1. Gesture resampling allows input gestures to have the same amount of points as their template counterpart, simplifying template comparison. The resampling process was adapted from the one specified in [36], which involves recreating a trace to have a fixed amount of interpolated points, referred to as *sampling resolution*. The sampling resolution used for CircGR was 64 points for all tests. Resampling a trace requires its length, which is divided by sampling resolution, to give an interval length. If the interval length is less than a limit $\iota$, the trace is considered an anchor, is not re-sampled, and contributes to the center of the gesture. Empirical observation yielded that the value of $\iota = 0.5$ pixels was a good minimum interval length of

a moving trace in our environment. As the points are interpolated for each trace, each angle $\alpha_i^S$ is calculated between every two interpolated points $p_i$ and $p_{i-1}$ in a trace as $\alpha_i^S = atan2(p_i.x - p_{i-1}.x, p_i.y - p_{i-1}.y)$, and adding $2\pi$ if $\alpha_i^S < 0$.

Temporal angle $\alpha_i^T$ calculation immediately follows $\alpha_i^S$ calculation during resampling, thus, there is an angle in $T$ for every angle in $S$ (i.e. $|T| = |S|$). The timestamp of the first and last points of the gesture determine the start time $t_s$ and end time $t_f$ of that gesture and its duration $\Delta_G = t_f - t_s$. The timestamp of interpolated points $p_i.t$ is used to generate $\alpha_i^T$ as shown in Equation 1. Interpolated points inherit the timestamp of the input point used to interpolate them. As each angle is calculated, the resultant vector for the gesture is also calculated, both for overall and each label.

$$\alpha_i^T = \frac{p_i.t - t_s}{\Delta_G} \times 2\pi \quad (1)$$

### Labels

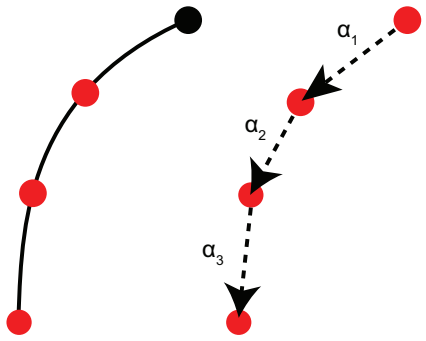
Each angle, spatial or temporal, is assigned 3 labels: a propagation label, a centripetal label, and a clock label. Temporal angles $\alpha_i^T$ inherit all the labels assigned to spatial angle $\alpha_i^S$. Propagation labels consist of 8 directions that are based on the cardinal and intercardinal directions as shown in Figure 2. The value of $\alpha^S$ and where the location of it falls determines the propagation label. A buffer called the "directional margin" is defined around each axis direction by $\Theta \in [0, \frac{\pi}{4}]$, as shown by the dotted lines in Figure 2. Under this format, $\alpha^S$ classified as "Down(D)" if it belongs to the set $\{\alpha^S | (\frac{\pi}{2} - \Theta) \leq \alpha^S \leq (\frac{\pi}{2} + \Theta)\}$ assuming the radians increase clockwise[1] from direction R as 0 rad. The direction designated as 0 rad is arbitrary as long as it's consistent and $\Theta$ values larger than $\frac{\pi}{4}$ are undefined. All testing for CircGR was done with $\Theta = \frac{\pi}{18}$ rad which yielded good results.

The centripetal labels consist of two directions: *I* (in) and *O* (out) and are assigned relative Euclidean distance, $d_E(p, q)$ to the gesture's center point $p_C$. Given two points $p_i$ and $p_{i-1}$ that generated $\alpha_i^S$, *I* is assigned to $\alpha_i^S$ if $d_E(p_i, p_C) \leq d_E(p_{i-1}, p_C)$; *O* otherwise.

Clock labels are assigned to $\alpha_i^S$ relative to the previous angle $\alpha_{i-1}^S$: *CW* (clockwise) and *CCW* (counter-clockwise). Clockwise label is assigned to $\alpha_i^S$ if tracing the smallest arc from $\alpha_{i-1}^S$ to $\alpha_i^S$ results in clockwise movement; the reverse would yield CCW. Depending on the chosen orientation, one label is assigned if $\alpha_{i-1}^S < \alpha_i^S \leq (\alpha_{i-1}^S + \pi)$ and the other label otherwise. Care has to be taken when $(\alpha_{i-1}^S + \pi) > 2\pi$.

Labels are a simplified form of definition elements typically found in gesture definition languages. GeForMT, for example, utilizes labels such propagation(referred to as "direction" in that work) and clock (refereed to as "rotation") as grammar elements to define a given gesture [17]. However, GeForMT is a formalization which requires an implementation that supports the identification of these elements as well as other numerous elements of its grammar. CircGR utilizes labels for simple-to-implement and fast clustering of angles in order to facilitate comparison during classification as well as represent temporal features in gestures without requiring further interpretation by a syntax engine.

[1]The inversion from the typical counter-clockwise orientation is due to touch-screens usually setting the origin at the upper left corner with the positive y-axis pointing downward.




**(a) Re-sampled Trace.** **(b) Spatial Angles.**

**Figure 1. Spatial Observation Generation.**

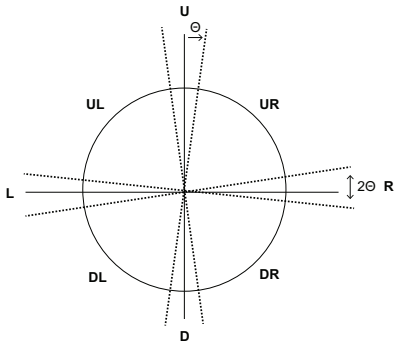



**Figure 2. The 8 Propagation Labels.**

### Input Capturing

CircGR receives input gesture as a series of traces. How the traces are generated and sent to CircGR can vary. In our approach, a variable defines the minimum amount of points $n$ collected before classification. The number of fingers influence the amount of points generated by the hardware during an interval, so the minimum number of points (64 in our experiments) to trigger classification was set at $n \times f$ where $f$ is the number of fingers. Once that minimum was met, the traces collected up to that point represented an *input window* and were classified by CircGR while the next input window buffered and appended to the original window. If the gesture stopped before meeting the minimum amount of points, the input window was sent as is. The result was periodic classification executed as the users entered performed the gesture over a series of windows in real time. Setting the window to a smaller minimum $n$ increases the rate of classification at the expense of giving the classifier less information per window.

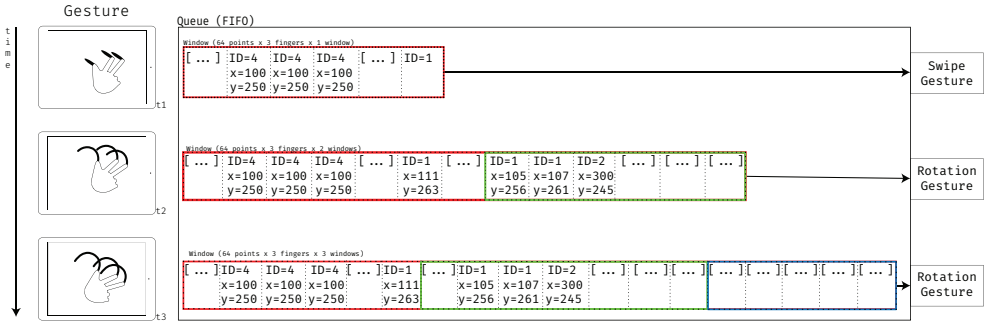



**Figure 3. Early Gesture Detection Example with Multiple Windows (over time $t_1$, $t_2$, and $t_3$).**

### Early Gesture Detection

An important feature of CircGR is that it provides early gesture detection for multi-touch gestures. In many detection algorithms (not including ad-hoc detection algorithms), such as $P [34], detection is completed at the end of the trace (or gesture). However, for gestures that require immediate feedback, this option is not ideal. As explained in the previous paragraph, CircGR performs continuous detection once each window is completed (in our case every 64 resample points) or once the input of data points stops. That means that for gestures that have multiple windows we can start performing (early) detection as soon as the window of size N is completed. Our experiments yielded ACC=99% and MCC = 86% for the first window. It is important to note that some gestures may have less than 64 resample points (as is the case for short interaction gestures) and each user may produce a different number of points for similar gestures. Nevertheless, this recognition rate, while lower compared to our overall recognition accuracy at the end of the gestures (ACC = 99% and MCC = 0.95), it is still fairly accurate when one considers the few data points which are considered. An example is provided in Figure 3. In this example, a gesture generates multiple windows. When the first window is captured, there is immediate recognition. With most gestures, the classification is accurate in the first window (ACC=99%, MCC=86%). As more data is collected the algorithm considers all windows up to that point. For our experiment, the window size was 64 points for each finger (trace). Therefore, in the last window in this example, there are 64 points $\times$ 3 fingers $\times$ 3 windows totaling 576 points for classification at $t_3$. Note: if the gesture ends before the last window has all N points the window gesture will be analyzed the same way

### CircGR: RECOGNITION

Recognition is simply the template that has the lowest distance toward the input. The distance between two angles, and total distance is simply the sum of these distances calculated over spatial and temporal angles. Only similarly labeled angles of the same type (spatial or temporal) are compared. Typically, this approach introduces the need to "pair" input and template angles in order to minimize the total distance. The $-family's $P approaches this problem with a greedy algorithm [34]. As circular data is one-dimensional, a simpler approach was possible without a significant loss of classification performance. In the case of spatial angles, the observations are sorted in ascending order, then paired off sequentially. Temporal angles are added in the order in which they occur, so they are paired off in that order without sorting.

Undoubtedly, there will be mismatches in the number of angles for specific labels. For example, an input gesture that contains "Up" angles exclusively might be compared to a template that contains only "Down." In this case, distance penalties are levied against the template depending on the type of angle. For spatial angles of a label, every unpaired angle in the template is compared to the resultant vector of the candidate for that label (and vise versa). If there is no resultant for a label, then the gesture resultant vector $R$ is used. For temporal angles, a penalty of $\pi$ is added to the total distance for each unmatched angle.

Potential templates are culled if the amount of fingers or the amount of anchors don't match before continuing the recognition process. As such, it is possible to obtain no classification if the user enters a gesture that is explicitly outside of the gesture set.

### CircGR: RUNTIME ANALYSIS

CircGR revolves around two basic processes, construction and classification. The construction algorithm involves the creation of a gesture representation from raw 2D input data as discussed above. Construct first acquires information on each trace (including possible anchors) through Algorithm 1, which runs at $\mathrm{O}(tn)$, with $n$ being the number of points for the largest trace and $t$ being the number of traces. In practice, the number of traces are limited to 10 or less due to human limitation, so at worst, Algorithm 1 is linear in $\mathrm{O}(10n) \approx \mathrm{O}(n)$. Construction performs anchor detection and centroid calculation at $\mathrm{O}(t)$ (see Algorithm 4 in appendix). The second notable part of construction is processing each trace (excluding anchors) through ProcessTrace (Algorithm 3) which consumes the information calculated at earlier generated angles. All algorithms within the *while* loop are $\mathrm{O}(1)$, leaving the *for* and *while* loop to contribute the bulk of the running time. Processing each trace through both these loops is done in $\mathrm{O}(kn)$, where $k$ is some constant that depends on the exact nature of the points in the trace. If a trace consists of equally spaced points and |trace| is equal to the sampling resolution, then $k \approx 1$ because that trace would be similar to its re-sampled counterpart. Deviations

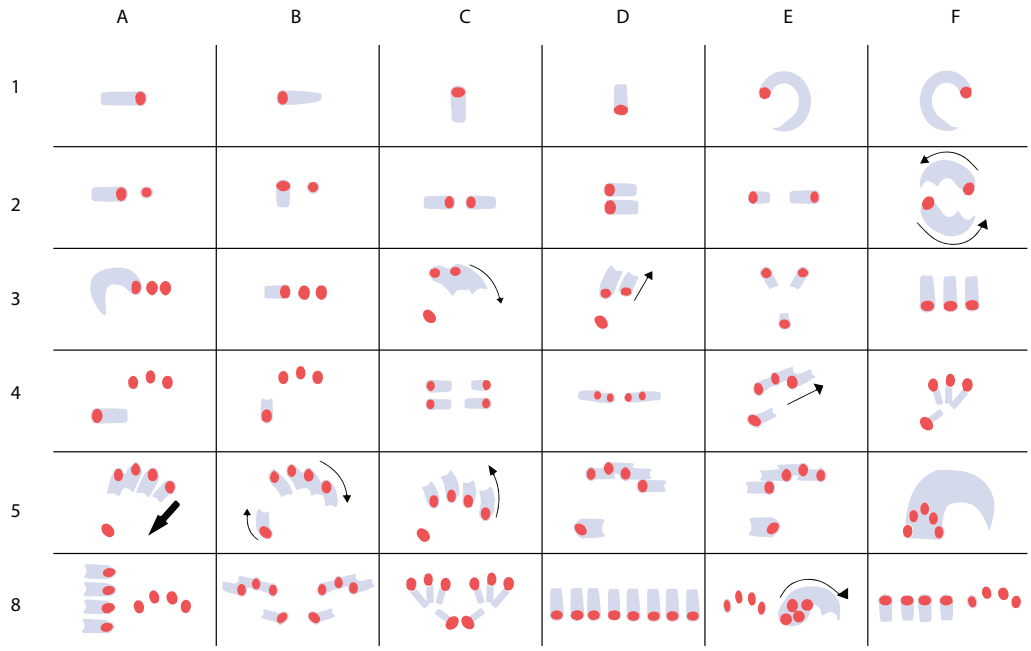



**Figure 4. Complete Gesture Set.**

from this counterpart would increase $k$ to some larger constant. Nonetheless, the method scales with number of points passed as a parameter, $O(n)$ over all. ProcessTrace is called $t$ times for a run time of $O(10n)$. Calculation of resultants is $O(1)$ for a total construction time of linear $O(n)$ overall with a variable constant depending on amount of trace.

Classification is implemented in Algorithm 2 (see appendix). Each template is compared to the candidate input in all labels through CalculateDistance in Algorithm 5 (see appendix). CalculateDistance involves sorting spatial angles for that label and then calculating the circular distance. Sorting dominates the run time, making Algorithm 5 $O(n \log n)$. In the worst case scenario, all templates are compared for a running time of $O(Tn \log n)$ where $T$ is the amount of templates. Furthermore, given that templates are culled if they do not have the same number of traces or the same number of anchors, the average run time constant for $T$ will typically be lower depending on the gesture set.

## EXPERIMENT DESIGN

CircGR was evaluated via 12,960 gestures collected from 60 participants over the 36 gesture specified in Figure 4. Each gesture in Figure 4 is displayed as a contact point and a propagation direction after contact (e.g., A1 is a one finger left swipe). Contact points without movement represent anchors. The gesture set utilized here was expanded from a set of 20 gestures specified in [10].

### Participants

Each participant was given an entry questionnaire to establish a baseline for previous touchscreen experience. Participants Ages ranged from 18 to 47, with an average of 24.2 (SD=4.1) and a male majority (~75%). Most were students working toward a degree in Computer Science; over 90% were right handed. Participants were asked to select the frequency at which they use non-smartphone touchscreen-enabled devices as "Never", "A couple of times a year", "Few times a months", "Few times a week", "About once every day", "Multiple times a day", and "Multiple times a day for long periods of time." 39% of the participants reported using touch-enabled devices multiple times a day or multiple times a day for long hours. Roughly two-thirds of the participants reported that they use touchscreen-enabled devices at least as frequently as a "few times a week." The majority(86.1%) reported to have experienced an increase in preference for using touchscreens since obtaining their smartphone. When asked about the importance of having gestures to touchscreen usage, 66.2% felt it was important, with 88.2% feeling that it was at least "somewhat" important. The majority (58.7%) felt that multi-touch systems should allow users to specify custom gestures, with 80% reporting that this quality is at least somewhat important. The average number of touchscreen-enabled devices used per participant, not including smartphones, was 2.1 (SD=1.41).

### Environment

Experiment was run in a canvas environment created in C# application (using Windows API for multi-touch events) running on Windows 7 PC and standard (Acer FT220HQL) multi-touch display. The application logged all of the touch data as a bundle of traces made up of 2D points tagged with their touch ID and timestamps.

### Participant Groups

The templates used for gestures set in Figure 4 were created beforehand. In order to see if there was a difference between

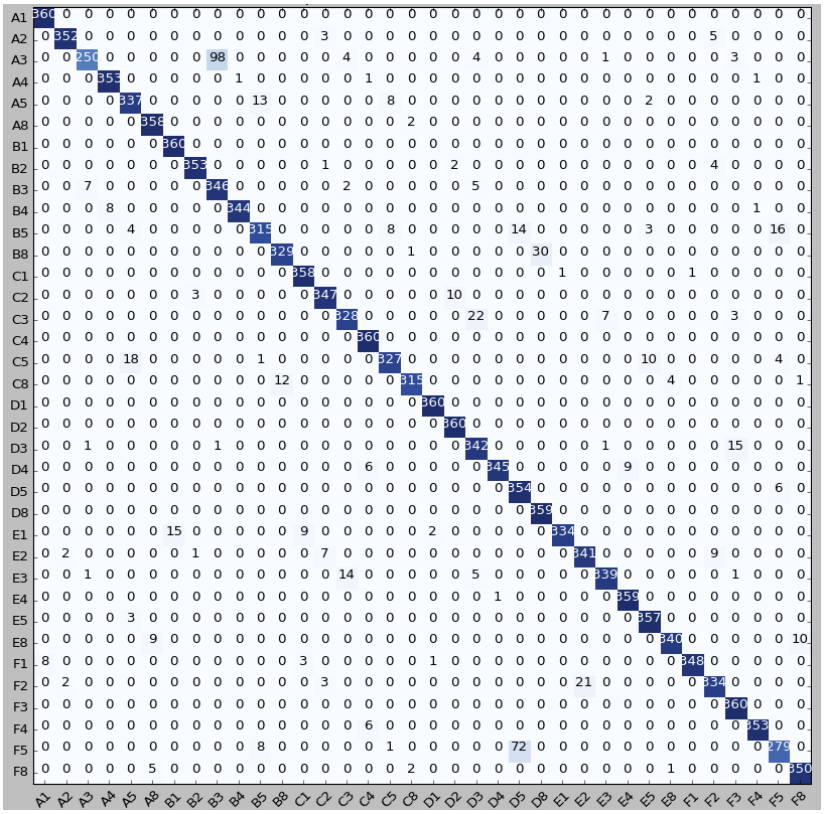


**Figure 5. Confusion matrix over all gestures captured in the experiment (true label).**

using these templates and user created templates for the gesture set, two groups of users were identified: the "developer template" group used the templates created specifically for the experiment and the "user template" group created their own templates for each gesture. Each participant was randomly assigned to a group at the start of the experiment. In order to standardize the procedure, both groups provided one template for each of the gestures in the gesture set before recognition trials began, but only the "user template" group had their templates used during the testing process.

### Testing Process

The order in which gestures were presented was randomized for each trial. Testing the classifiers on a specific gesture involved two phases, training and recognition. During the training phase, participants were asked to enter the gesture twice to familiarize themselves with the gesture before recognition. During recognition, participants entered the gesture 6 times within a bounded box that moves between gestures to stimulate articulation at different points of the screen. If the user entered an explicitly incorrect gesture (e.g, using 5 fingers when a 2 finger gesture was requested), the user was asked to redo the gesture. Classification was performed during every window interval, as well as at the end of the gesture when participant ceased all contact with the screen. Afterwards, the user was given 4 questions regarding the comfort of the gesture they just performed, the suitability of that gesture for mobile devices, the suitability of that gesture for everyday use, and the suitability of that gesture for large screen devices such as desktops. Answers were given in the form of a 7-point Likert scale. The process was repeated for all gestures for an estimated 36x8=288 gestures per participant.

## RESULTS

The confusion matrix over all the gesture captured for all participants is shown in Figure 5. The confusion matrix displays classification results at the last window of input. Results are consistent and promising through most of the gestures of the gesture set, as demonstrated by the concentration of values along the diagonal. The largest confusion for CircGR occurred for gesture A3, confused for B3. Both gestures involve 2 anchors and movement to the left, A3 featuring a curved movement versus B3 straight propagation. Interestingly, the confusion is not pronounced in the opposite (B3 to A3) direction. Inattentive or fatigued execution can easily morph gesture A3 into B3 (e.g., the user doesn't curve enough) while modifying B3's execution to A3 requires far more effort. The same pattern holds for the second and third most confused gesture, F5 (confused for D5) and B8 (confused for D8), indicating sub-optimal user execution was a contributing factor to these misclassification and underscoring the importance of incorporating human limitations and a user's liability to modify gesture execution. From the confusion matrix, CircGR averaged an ACC of over 99% demonstrating high recognition performance. The MCC metric ranges between -1 to 1, 1 indicating perfect prediction and -1 representing total dis-

agreement. MCC is usually lower than ACC due to the introduction of penalties when the different templates are unevenly represented over the data set. Regardless, CircGR managed an MCC of 0.95, or 95% on average, over all gestures.

As previously mentioned, CircGR classifies touch input as a series of input windows which represent the gesture. The confusion matrix shows the results of classification at the last input window (i.e. the point users fingers left the screen). We used Cohen's Kappa ($\kappa$) to test the correct identification of gestures by our algorithm through each input windows. Given that the number of windows varies per gesture, ACC and MCC are unsuitable metrics for this assessment. The $\kappa$ statistic, like most measures of agreement, ranges from $-1$ to $+1$, where 0 represents the amount of agreement that can be expected from random chance. The $\kappa$ statistic is specifically used when a level of agreement is already expected in the data (e.g., filtering guarantees that 1 finger gestures will not be misclassified as other multi-finger gestures). While there are no undisputed limits for defining agreement based on the magnitude of the kappa statistic it can be generally agreed that if agreement is less than that expected by chance then $\kappa \leq 0$ and if agreement is greater than that expected by chance then $\kappa \geq 0$. For values $\leq 0.4$ indicate poor agreement, values $0.4 - 0.75$ indicate moderate agreement, and values $\geq 0.75$ indicate excellent agreement. This applies to our experiment as we use $\kappa$ to measure the agreement of CircGR's classification to the gesture expected at that input window. Futhermore, $\kappa$ accounts for chance agreement giving a better estimate of the true agreement of the CircGR, where chance agreement represents the probability that CircGR will choose the correct template at random at that window. Figure 6 shows the values of the kappa statistic based on the window being considered. As we can see in all cases, value above 0.82 indicates near perfect agreement based on general guidelines per window.

When considering the first input window exclusively, CircGR obtained 99% ACC, 0.86 MCC, and $\kappa \approx 0.90$, which indicates good performance from the first input window and demonstrating promising results in terms of early detection.

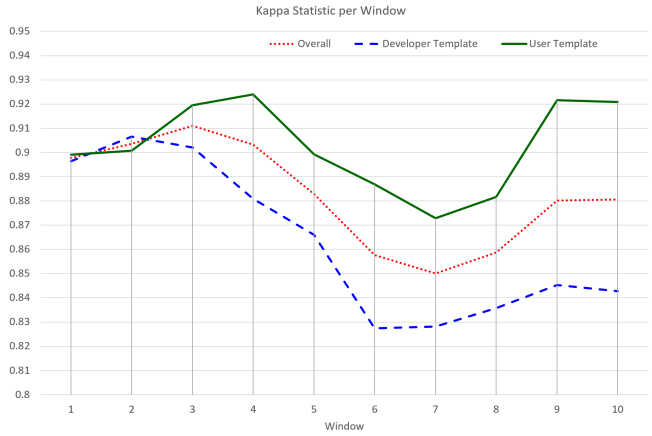



**Figure 6. Cohen's Kappa ($\kappa$) statistic per window.**

### Limitations of the Study

The study has the following limitations: (1) complex gestures are more likely to produce user-error, reducing the efficiency of early-detection of the correct gesture (a four-finger gesture may be recognized as a three-finger gesture because of user error); and (2) the objective of this study was to identify the accuracy of the algorithm for a pre-determined set of gestures which were presented to the user on a one-by-one basis.

## POST-HOC ANALYSIS: COMPARISON OF CircGR AND $P

All the gestures captured in the experiment were subject to classification by CircGR and $P in a post-hoc experiment to compare (1) recognition performance and (2) time taken to classify gestures. $P was choosen due to its significance as a simple to understand, implement, and use template matcher. Recognition performance was obtained by off-line batch classification of the 12,961 gestures captured during the experiment. Timing benchmarks where done at the batch level (timing how long it took to classify the entire batch) and at the gesture level (timing how long each gesture took to classify). Due to time variations stemming from the underlying OS, 30 trials where run on each classifier. An additional classifier $P-Filter was added to observe the classification impact of filtering out incorrect number of fingers. $P-Filter is $P with an added benefit of invalidating templates that don't match the amount of fingers used during input, an aspect that is prominent in CircGR. The same templates were provided as developer templates during the experiment were used for post-hoc. Like the experiment, only one template for each gesture in the gesture set was used as training data. All benchmarking for recognition and time were performed on the same Dell Precision T1700 with an Intel®Core™i7-4790 @ 3.60 GHz and 16 GB RAM.

Overall averages for all gestures across each benchmark are illustrated in Table 1. Testing was done using Wilcoxon-Sign Rank Sum Test, for pairwise comparison. Prior to this analysis we ran the Shapiro-Wilk test on both recognition rates, as well as, time benchmarks; in both cases we found p-values $<$ 0.01 which indicate severe deviations from normality. Based on this evidence, we proceeded to use non-parametric comparison methods. We first used the Kruskal-Wallis Rank Sum test to identify whether there were any significant differences overall. In all cases we found p-values $< 0.01$ which indicate significant differences between classifiers. The pairwise Wilcoxon-Sign Rank Test was used as a follow up to identify the different pairs. Our analysis shows that CircGR has significantly higher accuracy rates and MCC than $P for all gestures in the gesture set, and higher accuracy rates and MCC than $P Filtered in nearly all gestures, the exception being two three-finger gestures, A3 and F3. This observation is based on the results of the Wilcoxon Rank Sum Test using a 95% confidence level. For a more complete analysis we also compared $P and $P-Filter and found that in 9 gestures (all 5-finger gestures, C1, D1, and C8) their ACC and MCC were no different. $P-Filter was significantly performed significantly better in both metrics in the remaining gestures.

In comparison to time benchmark, CircGR was found to be significantly faster than $P and $P-Filter using Wilcoxon-

Sign Rank Sum Test. The time it takes to classify gestures by users who utilized developer templates is not significantly different than users who used user templates at a 5% level of significance over all classifiers using the Wilcoxon Rank Sum Test. We also compared developer versus user templates for each independent classifier. Our analysis concluded that CircGR and $P take a similar amount of time with developer and user templates while $P-Filter takes a significantly longer amount of time when compared to user templates, meaning that filtering introduces variation to running time depending on how user executes the gesture.

**Table 1.** Post-Hoc Results: CircGR and $P.

| Classifier | Running Time | ACC | MCC | Batch Time | PG‡ Time |
|---|---|---|---|---|---|
| CircGR | $O(T * n \log n)$ | 99% | 0.95 | 3.46 s | 0.5 ms |
| $P | $O(T * n^{2.5})$ | 97% | 0.45 | 125 s | 19 ms |
| $P-Filter | $O(T * n^{2.5})$ | 98% | 0.68 | 20.9 s | 3.2 ms |

Legend: ‡Per-Gesture

## CONCLUSIONS AND FUTURE WORK

We have described CircGR, a template matcher based on circular angles. We have demonstrated that CircGR can accurately classify non-symbolic multi-touch gestures in real time with significant ACC of 99%, MCC of 0.95, and significant kappa statistics of across both early and late windows highlighting its ability for early recognition as well as continuous detection. CircGR's lack of excessive training data requirements allows it to be easily expanded or supplemented by samples from the users which demonstratively shown increase performance over developer provided templates. CircGR, as presented, is simple to understand and implement without specific expert knowledge. Future work includes the possibility of expanding our work to 3D gesture recognition. In addition, we will conduct an experiment to test 3D travel with multi-touch using our algorithm. This implies that we will need to consider how early detection plays a role in the interaction of users. Finally, CircGR can be linearithmic under the assumption that spatial observation must be sorted during classification. Sorting provides the highest accuracy, but this is not required for CircGR. A future study may evaluate the degradation of accuracy when omitting sorting in CircGR.

### Acknowledgments

Support provided by the National Science Foundation: I/UCRC IIP-1338922, III-Large IIS-1213026, MRI CNS-1429345, MRI CNS-1532061, MRI CNS-1532061, MRI CNS-1429345, RAPID CNS-1507611, DUE-1643965. U.S. DOT Grant ARI73. We acknowledge Lukas Borges, Vesna Babarogic, Alain Galvan, and Jonathan Bernal. Finally, Jake Wobbrock, Lisa Anthony, and Radu-Daniel Vatavu for the many email discussions about multi-touch gesture recognition.

## REFERENCES

1. GestureML - Gesture Markup Language. `http://www.gestureml.org/doku.php/gestureml`, 2014. Accessed: 2017-3-10.
2. Abe, T., Shizuki, B., and Tanaka, J. Input techniques to the surface around a smartphone using a magnet attached on a stylus. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '16, ACM (New York, NY, USA, 2016), 2395–2402.
3. Anthony, L., and Wobbrock, J. O. A lightweight multistroke recognizer for user interface prototypes. In *Proceedings of Graphics Interface 2010*, GI '10, Canadian Information Processing Society (Toronto, Ont., Canada, Canada, 2010), 245–252.
4. Anthony, L., and Wobbrock, J. O. $n-protractor: A fast and accurate multistroke recognizer. In *Proceedings of Graphics Interface 2012*, GI '12, Canadian Information Processing Society (Toronto, Ont., Canada, Canada, 2012), 117–120.
5. Bailador, G., Roggen, D., Tröster, G., and Triviño, G. Real time gesture recognition using continuous time recurrent neural networks. In *Proceedings of the ICST 2Nd International Conference on Body Area Networks*, BodyNets '07, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering) (ICST, Brussels, Belgium, Belgium, 2007), 15:1–15:8.
6. Beh, J., Han, D., and Ko, H. Rule-based trajectory segmentation for modeling hand motion trajectory. *Pattern Recogn. 47*, 4 (Apr. 2014), 1586–1601.
7. Benabbas, Y., Lablack, A., Ihaddadene, N., and Djeraba, C. Action recognition using direction models of motion. In *2010 20th International Conference on Pattern Recognition* (Aug 2010), 4295–4298.
8. Bimber, O. Continuous 6dof gesture recognition: a fuzzy logic approach. *gesture 1* (1999), 1.
9. Cirelli, M., and Nakamura, R. A Survey on Multi-touch Gesture Recognition and Multi-touch Frameworks. *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces - ITS '14* (2014).
10. Damaraju, S., and Kerne, A. A gesture learning and recognition system for multitouch interaction design (poster). `https://goo.gl/DSakjP`, 2009. Accessed: 2017-4-2.
11. Dittmar, T., Krull, C., and Horton, G. A new approach for touch gesture recognition: Conversive Hidden non-Markovian Models. *Journal of Computational Science 10* (2015), 66.
12. Echtler, F., and Butz, A. Gispl: Gestures made easy. In *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction*, TEI '12, ACM (New York, NY, USA, 2012), 233–240.
13. Freeman, H. On the encoding of arbitrary geometric configurations. *IRE Transactions on Electronic Computers EC-10*, 2 (June 1961), 260–268.

14. Hong, P., Turk, M., and Huang, T. S. Constructing finite state machines for fast gesture recognition. *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000* (2000).

15. Hoste, L., De Rooms, B., and Signer, B. Declarative Gesture Spotting Using Inferred and Refined Control Points. In *Proceedings of ICPRAM 2013, 2nd International Conference on Pattern Recognition Applications and Methods* (Barcelona, Spain, 2013).

16. Jammalamadaka, S. R., and SenGupta, A. *Topics in Circular Statistics (Series on multivariate analysis ; v. 5)*, 1 ed. World Scientific, 2001.

17. Kammer, D., Wojdziak, J., Keck, M., Groh, R., and Taranko, S. Towards a formalization of multi-touch gestures. In *ACM International Conference on Interactive Tabletops and Surfaces*, ITS '10, ACM (New York, NY, USA, 2010), 49–58.

18. Khandkar, S. H., and Maurer, F. A domain specific language to define gestures for multi-touch applications. In *Proceedings of the 10th Workshop on Domain-Specific Modeling*, DSM '10, ACM (New York, NY, USA, 2010), 2:1–2:6.

19. Kin, K., Hartmann, B., DeRose, T., and Agrawala, M. Proton++: A customizable declarative multitouch framework. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, ACM (New York, NY, USA, 2012), 477–486.

20. Kin, K., Hartmann, B., DeRose, T., and Agrawala, M. Proton: Multitouch gestures as regular expressions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, ACM (New York, NY, USA, 2012), 2885–2894.

21. Lü, H., and Li, Y. Gesture coder: A tool for programming multi-touch gestures by demonstration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, ACM (New York, NY, USA, 2012), 2875–2884.

22. Myers, C. S., and Rabiner, L. R. A comparative study of several dynamic time-warping algorithms for connected-word recognition. *The Bell System Technical Journal 60*, 7 (Sept 1981), 1389–1409.

23. NUI Group. Gesture Definition Markup Language(GDML). `http://goo.gl/ZYuf6N`, 2009. Accessed: 2017-2-10.

24. Ortega, F. R., Galvan, A., Tarre, K., Barreto, A., Rishe, N., Bernal, J., Balcazar, R., and Thomas, J. L. Gesture elicitation for 3d travel via multi-touch and mid-air systems for procedurally generated pseudo-universe. In *2017 IEEE Symposium on 3D User Interfaces (3DUI)* (March 2017), 144–153.

25. Pittman, J. A. Recognizing handwritten text. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '91, ACM (New York, NY, USA, 1991), 271–275.

26. Rekik, Y., Vatavu, R.-D., and Grisoni, L. Match-up & conquer: A two-step technique for recognizing unconstrained bimanual and multi-finger touch input. In *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces*, AVI '14, ACM (New York, NY, USA, 2014), 201–208.

27. Renaux, T., Hoste, L., Marr, S., and De Meuter, W. Parallel gesture recognition with soft real-time guarantees. In *Proceedings of the 2Nd Edition on Programming Systems, Languages and Applications Based on Actors, Agents, and Decentralized Control Abstractions*, AGERE! 2012, ACM (New York, NY, USA, 2012), 35–46.

28. Rubine, D. Specifying gestures by example. *SIGGRAPH Comput. Graph. 25*, 4 (July 1991), 329–337.

29. Schmidt, M., and Weber, G. Template based classification of multi-touch gestures. *Pattern Recognition 46*, 9 (2013), 2487.

30. Scholliers, C., Hoste, L., Signer, B., and De Meuter, W. Midas: A declarative multi-touch interaction framework. In *Proceedings of the Fifth International Conference on Tangible, Embedded, and Embodied Interaction*, TEI '11, ACM (New York, NY, USA, 2011), 49–56.

31. Sezgin, T. M., and Davis, R. HMM-based efficient sketch recognition. *Proceedings of the 10th international conference on Intelligent user interfaces - IUI '05* (2005).

32. Spano, L. D., Cisternino, A., Paternò, F., and Fenu, G. Gestit: A declarative and compositional framework for multiplatform gesture definition. In *Proceedings of the 5th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '13, ACM (New York, NY, USA, 2013), 187–196.

33. Starner, T., Weaver, J., and Pentland, A. Real-time american sign language recognition using desk and wearable computer based video. *IEEE Transactions on Pattern Analysis and Machine Intelligence 20*, 12 (Dec 1998), 1371–1375.

34. Vatavu, R.-D., Anthony, L., and Wobbrock, J. O. Gestures as point clouds: A $p recognizer for user interface prototypes. In *Proceedings of the 14th ACM International Conference on Multimodal Interaction*, ICMI '12, ACM (New York, NY, USA, 2012), 273–280.

35. Wilson, A., and Shafer, S. Xwand: Ui for intelligent spaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, ACM (New York, NY, USA, 2003), 545–552.

36. Wobbrock, J. O., Wilson, A. D., and Li, Y. Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, UIST '07, ACM (New York, NY, USA, 2007), 159–168.

37. Zhai, S., and Kristensson, P.-O. Shorthand writing on stylus keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, ACM (New York, NY, USA, 2003), 97–104.