Bootstrapping Personal Gesture Shortcuts with the Wisdom of the Crowd and Handwriting Recognition

Tom Y. Ouyang [*]
MIT CSAIL
ouyang@csail.mit.edu

Yang Li
Google Research
yangli@acm.org

ABSTRACT
Personal user-defined gesture shortcuts have shown great potential for accessing the ever-growing amount of data and computing power on touchscreen mobile devices. However, their lack of scalability is a major challenge for their wide adoption. In this paper, we present Gesture Marks, a novel approach to touch-gesture interaction that allows a user to access applications and websites using gestures without having to define them first. It offers two distinctive solutions to address the problem of scalability. First, it leverages the "wisdom of the crowd", a continually evolving library of gesture shortcuts that are collected from the user population, to infer the meaning of gestures that a user never defined himself. Second, it combines an extensible template-based gesture recognizer with a specialized handwriting recognizer to even better address handwriting-based gestures, which are a common form of gesture shortcut. These approaches effectively bootstrap a user's personal gesture library, alleviating the need to define most gestures manually. Our work was motivated and validated via a series of user studies, and the findings from these studies add to the body of knowledge on gesture-based interaction.

Author Keywords
Gesture-based interaction, search, mobile computing.

ACM Classification Keywords
H.5.2 [Information Interfaces and Presentation]: User Interfaces. H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval.

General Terms
Design, Human Factors, Algorithms.

INTRODUCTION
With the ever-growing complexity of mobile devices like the Android [1] and iPhone [11], there is a pressing need for faster and more natural ways to access all of the diverse functions that are available to mobile users today. Gesture shortcuts on the touchscreen represent one promising new modality for confronting this problem. Imagine being able to draw a quick star to launch Google Sky Maps (an application for exploring stars, planets, and constellations) or a fast cursive "NY" to open your browser to The New York Times (as illustrated in Figure 1). These gestures can be quick to draw, easy to remember, and require very little cognitive load on the user.

Figure 1: When a user draws a gesture in Gesture Marks, the system shows a list of matches based on: 1) a library of learned gestures from the user, 2) gestures from other users—the crowd, and 3) a built-in handwriting recognizer.

A system that supports user-defined gesture shortcuts often employs a template-based approach [19], in which a new gesture is matched against a set of known gestures defined by the user. Compared to prior work that uses built-in, system-defined shortcuts, e.g., those based on letters [14,21], this approach enables a richer unconstrained gesture vocabulary and allows users to easily create their own personal shortcuts. A new gesture can be added to the library in an on-line manner without expensive retraining, and it can immediately contribute to future recognition. Furthermore, because such a system can continually learn from the user's gestures over time, it allows the gesture

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CHI'12, May 5–10, 2012, Austin, Texas, USA.
Copyright 2012 ACM 978-1-4503-1015-4/12/05...$10.00.

[*] This work was done while the author was an intern at Google Research.

language to adapt to the particular drawing or handwriting style of the user instead of the other way around.

However, despite the appeal and promise of user-defined gesture shortcuts, the main challenge is their scalability. It is not practical for a user to define custom gesture shortcuts for every target they might visit, e.g., accessing a large corpus such as the web. To address this challenge, our goal is to find ways to alleviate users from the burden of having to define gesture shortcuts manually.

To this end, we first conducted an exploratory study to find out whether there are any intrinsic properties or commonalities in gesture shortcuts, both within one individual as well as across individuals. The study revealed two important findings. First, we found a major portion of gesture shortcuts that users defined were character-based and mnemonically associated with the target, e.g., "NY" for The New York Times website. Second, there was a significant overlap in the gestures defined by different users for the same target, e.g., many users drew a plus symbol for Google+. These findings motivated us to develop Gesture Marks, a novel system that allows users to access applications and websites on their phone using natural unconstrained gesture shortcuts without necessarily having to define them beforehand.

Our major contribution is a novel approach for bootstrapping personal gesture shortcuts, using a combination of crowdsourcing and handwriting recognition. It alleviates the effort of defining gesture shortcuts and makes gesture-based interaction more scalable. The novelty of the system is two-fold. First, Gesture Marks leverages the "wisdom of the crowd". It infers what a user intends to access with a gesture by looking at how the gesture is commonly used by others. Second, it combines this template-based gesture recognizer with a robust cursive handwriting recognizer that can infer the user's target even when there is no matching target in the gesture library. The handwriting recognizer effectively captures the long tail of gesture shortcuts, ones that are rare or specific to an individual user. Since each gesture usage is recorded by Gesture Marks, the results of these gesture-based interactions all contribute to the user's personal library of gesture shortcuts. A validation study that we conducted showed that crowd-defined gestures and handwriting recognition effectively bootstrapped the gesture shortcut library of each individual user.

This paper focuses on applications and websites because these are two classes of targets that are likely to be shared between users, thus allowing us to better study the commonality in gesture shortcuts across the user population. However, we believe that many of the ideas presented here can be extended to other types of targets such as music and phone settings.

In the remainder of the paper, we first describe an exploratory study where we analyze the types of gestures people draw, and develop a set of guidelines based on our findings. We next present Gesture Marks, a novel gesture interface we designed and implemented based on those guidelines. We then present a set of evaluations on our system, and finally conclude with discussions of future and related work.

EXPLORING USER-DEFINED GESTURE SHORTCUTS

We conducted an exploratory study to investigate the types of gestures people tend to draw. In particular, we analyzed whether there were any visual or semantic patterns in gesture shortcuts defined by the same individual, and whether these patterns hold across individuals. We then used the findings to inform the design of a user-defined gesture shortcut system.

Data Collection Procedures

We collected gesture shortcut samples from 26 Android phone users for their personalized applications and websites. We asked these users to install a data collection tool that we developed on their Android phone. The tool automatically extracted a set of targets for which participants were asked to define gesture shortcuts. These targets consisted of 20 applications randomly selected from their device and the top 20 websites sampled from their browsing history based on visit count. The selection of these targets approximates the actual usage model of applications and websites for each user.

Participants were prompted with one target at a time, and they could not see any previous gestures they had already drawn. To make sure our gestures were representative of actual usage, participants were allowed to skip targets they did not recognize and would have no intention of using in real life. Each participant was prompted with the list of targets twice. For each round, we randomized the order of the occurrence of these targets. The procedure allowed a total of two gesture samples per user per target. Since the targets were personalized, different participant could have different sets of targets.

Results & Analyses

We collected a total of 1,970 gestures spanning 407 different targets from 26 users. Our analysis of this data revealed three important findings.

Frequency of Handwriting versus Drawing Gestures

First we analyzed what types of gestures people tended to draw for different types of targets. We labeled all of the gestures we collected as either *handwriting* or arbitrary *drawings* and calculated the frequency for each (see Figure 2). As we can see, a majority (72%) of the gestures people drew consisted of alphanumeric characters, i.e., letters and numbers. The rest (28%) were arbitrary drawings, e.g., boxes, stars, and birds. There was a larger percentage of non-handwriting gestures for applications (40%) than for websites (14%). One possible reason for this is that application icons, which users are constantly exposed to, inspired them to use drawing-based gestures.

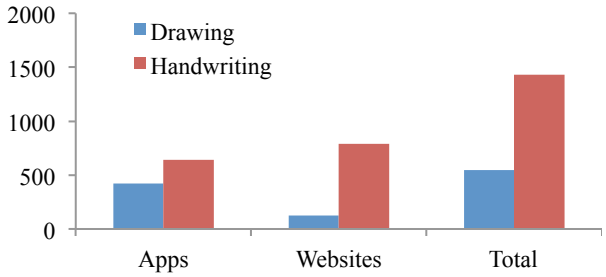

Figure 2: The frequency of arbitrary drawing gestures versus handwriting gestures.

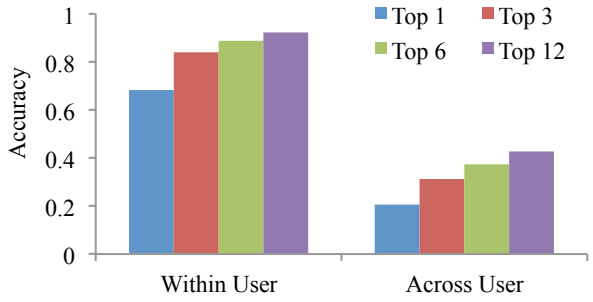

Figure 3: (left) Within-user similarity-based prediction accuracy and (right) across-user accuracy.

Consistency of Gesture Shortcuts Within Each User

Next we analyzed the consistency of gesture shortcuts created by *the same user*. The consistency here measures how well the visual similarity between two gestures predicts the likelihood that they both map to the same target.

To this end, we first split the gestures from each user into two halves, with each half containing one of the two examples for each target (assigned randomly). We then trained a template-based gesture recognizer on the first half and measured how well it can predict the intended targets for the gestures in the second half. The recognizer employs an appearance-based distance metric [17], so the result is an estimate for how often visually similar gestures map to the same target. We then repeated the experiment in reverse: training on the second half and testing on the first.

The accuracy results for this within-user cross validation analysis are shown in Figure 3 (left). We found that for a large majority of the shortcuts (68%), it is possible to predict the destination of a gesture by looking only at the most similar gesture (the nearest neighbor) drawn by the same user. One important reason that the accuracy was lower than that reported on previous gesture datasets [15,19,20] is the mapping ambiguity from gestures to targets. We frequently observed instances where the same user drew similar gestures for different targets (e.g., an "M" gesture for both Maps and Music), in which case it is impossible to determine the exact meaning of the gesture. In contrast, no such ambiguity needed to be addressed in previous experiments.

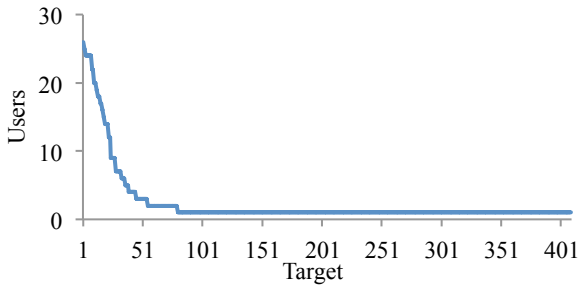

Figure 4: The number of users who defined gestures for each target obeys a power law

If we look at the top 12 nearest matches, the maximum number of targets that can be shown on the lower portion of our prototype interface (see Figure 1), the accuracy increases to 92%. With more gesture repetitions per target we may expect to see even higher accuracy.

Consistency of Gesture Shortcuts Across Users

Finally we analyzed whether there was any similarity in gesture shortcuts created by *different users*. We performed a similar analysis as in the within-user case before, except here there was no overlap between the users in the training and test sets. We are now measuring how well we can predict the target for a given gesture using only data from the other users.

We performed a total of 26 cross validation tests, one for each user, and the final accuracy results are shown in Figure 3 (right). The recognizer was able to predict the correct destination 21% of the time if we only look at the top match, and 42% of the time if we look at the top 12 matches. Notice that both the accuracy of the recognizer and the consistency of gesture shortcuts across users contributed to this result.

As expected, the accuracy was lower than in the within-user case, due in part to the following reasons. First, a large majority of the targets had gestures from only one user (see Figure 4). In such cases, it was impossible to predict what the user intended since there was no training data matching the target from any of the other users. Second, there was more mapping ambiguity because with multiple users, it was even more likely that some of them would assign different meanings to the same gesture.

However, the result is still encouraging given that our data was from only 26 users. As the system accumulates more and more gestures from the user population, this sparsity issue should become less of a problem. For example, if we repeat the same test but limit our dataset to only those shortcuts with gestures from 4 or more users (a total of 45 targets with 1146 gestures), the accuracy increases to 36% for the top 1 match and 77% for the top 12 matches. The mapping ambiguity issue can be addressed by taking into account of the popularity of gestures, e.g., the number of users who used the gesture, and the personal preferences and usage history of each user.

Discussions for Creating a Gesture Shortcut System

Based on these findings, we developed the following guidelines for creating a gesture shortcut system.

1. It should be able to predict the target for a gesture based on the targets of similar gestures from the same user.

The results in Figure 3 (left) show that visual similarity to the user's own gestures is a remarkably informative metric for predicting the target of a new gesture. This confirms previous work [20] that uses a template-based approach to recognize user-defined gesture shortcuts. A template-based gesture recognizer is also essential for non-handwriting input, where the number of gesture classes is unbounded and the number of training examples per class can be very low.

However, it will be impractical to expect users to define a gesture for every target they want to visit beforehand. In fact, the overhead for specifying user-defined gesture shortcuts has been the main challenge for its adoption in the real world. Thus, based on the findings from the study, we add two important bootstrapping and refinement mechanisms.

2. It should be able to take advantage of gestures drawn by other users.

The results in Figure 3 (right) show that there was significant predictive power from gestures drawn by other users, even in this study where the number of users is small and gestures per target were extremely sparse. In a real world setting where the number of users is continually expanding, we would expect the predictiveness of cross-user gestures to be even higher. This implies that even before a user defines any gesture shortcuts, there is the possibility to interpret the user gesture by looking at what other users drew.

3. It should be able to recognize handwriting gestures even when there are no matching templates.

Since handwriting gestures are used so frequently (see Figure 2), incorporating handwriting recognition into the system allows it to effectively recognize these gestures even when they have never been defined before by any user. This is especially useful for rare targets or targets that are specific to a given user.

GESTURE MARKS

Based on the findings and guidelines derived from the exploratory study, we designed and implemented Gesture Marks, a system that allows the user to use unconstrained gesture shortcuts to access applications and websites. Here we describe how users interact with the system.

Querying Based on Multiple Sources

Gesture Marks is implemented as an alternative Android home screen. When the user draws a gesture query on the screen, Gesture Marks interprets it based on three sources: 1) a library of learned gestures from the user; 2) gestures from other users; and 3) a built-in handwriting recognizer. The system then displays the best matches in a list view below (see Figure 1). The user chooses the correct target

Figure 5: Defining a new gesture shortcut drawn from the home screen.

Figure 6: Defining a new gesture shortcut from the gesture explorer view

from the list, which launches the website or application. Meanwhile, the system learns the new association between the user drawn gesture and the target for future queries.

Defining Gestures As a By-Product of Finding Targets

The intended target might not always be in the list of matches when the user draws a gesture query (see Figure 5 left). If this happens the user can manually define the gesture by clicking on the Manual Selection button. This opens a list of all apps and visited websites, as well as a search box for text queries (see Figure 5 right). The gesture just drawn by the user is shown in a miniaturized version on the upper left. When the user chooses the intended target, Gesture Marks asks her whether she would like the system to learn the definition for future queries before launching it.

Exploring Gestures from Others

The user can also perform a long finger press on any target to bring up a screen that shows all of the gestures assigned to that target (see Figure 6). Here she can explicitly define a new gesture shortcut by drawing it in the gesture panel on the bottom. She can also see what popular gestures other people have used for the target. Each crowd gesture is shown with its popularity metrics including the number of gesture examples and the number of unique users. If the user dislikes any of the gestures, she can remove it by

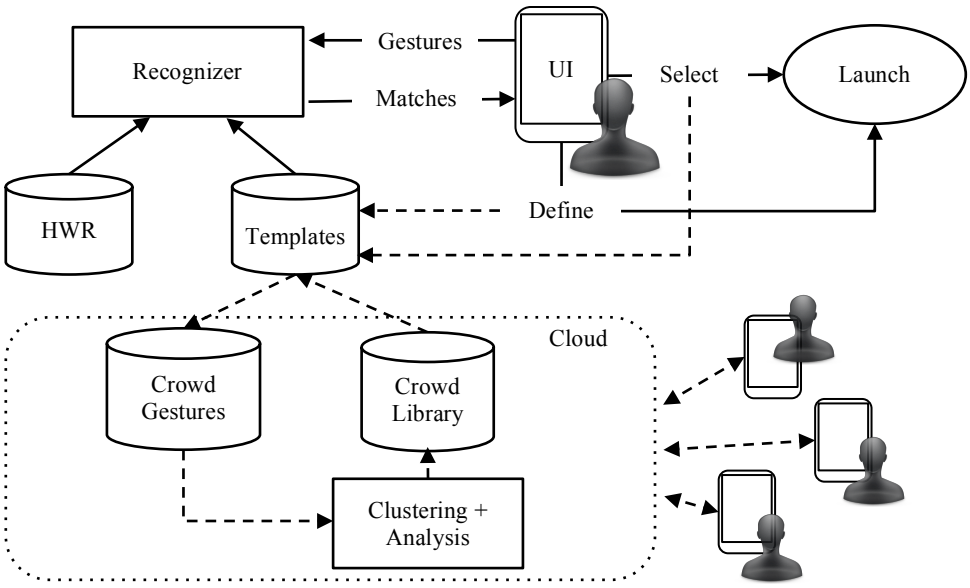

Figure 7: The Gesture Marks System Architecture.

clicking on the X next to it, which informs the system not to use this gesture for the target in the future.

THE SYSTEM ARCHITECTURE

This section presents the Gesture Marks architecture, shown in Figure 7. When the user draws a gesture, it is processed by both an extensible template-based gesture recognizer and a built-in handwriting recognizer. The resulting matches are presented to the user, who can either select one if the correct target is in the list or define the target manually. In either case, the new gesture is learned by the system. In addition, each gesture is recorded in the crowd-based gesture library, which clusters, analyzes, and shares the gesture shortcuts among the user population.

Inferring Shortcuts with Gesture Matching

Gesture Marks employs a template-based recognizer that infers the target of a user gesture by matching it against a library of collected gesture shortcuts. Every time the user draws a gesture and selects a match, the system stores the new gesture shortcut in the user's library that is both maintained on a server in the cloud and cached locally on the device. The recognition performance thus improves over time as the system accumulates more examples from the user for each target. Furthermore, since the gestures are cached, matches can be shown instantly for favorite shortcuts even if the connection to the Gesture Marks server is slow or inaccessible.

In addition, the system leverages the "wisdom of the crowd" by harvesting gesture shortcuts from the entire user population. It then recommends relevant gestures to individual users based on their installed applications and browsing history.

Gesture Matching

An important component of the recognizer is how it matches an input gesture against a set of learned gesture shortcuts. We adapted an appearance-based matching algorithm that had been applied successfully to the domain of sketch recognition on pen-based tablets [17] and tailored it to our domain - mobile touchscreen devices. Since this recognizer focuses on vision-based features instead of temporal stroke patterns, it naturally overcomes many common gesture recognition challenges such as stroke order differences and over-tracing (drawing over a previously drawn region). The performance of the recognizer is competitive with prior approaches [14,19,20] on recognizing single-stroke gestures. Furthermore, our recognizer also naturally handles multi-stroke gestures at no additional cost. A more detailed discussion on the implementation and evaluation of this recognition approach is available in [17].

Gesture Clustering

In order to facilitate fast retrieval and shortcut analysis, the server first clusters together similar gestures for each target. This is done using an agglomerative clustering algorithm [7] that produces a hierarchy of progressively larger clusters. Each cluster can then be described by 1) its center: the gesture that has the minimum average distance to each of the other members of the cluster; and 2) its diameter: the maximum distance between any two gestures in the cluster. We stop growing clusters when the diameter reaches a predetermined threshold, chosen empirically based on analysis of the similarity scores from the previous exploratory study.

Online Clustering

As new gestures are added to the server, they need to be incorporated into the cluster hierarchy to be used effectively. To accomplish this task Gesture Marks employs a scalable distributed agglomerative clustering algorithm that first splits the data into partitions, and then

clusters each partition independently. It then merges the clusters generated across multiple partitions by keeping only their respective cluster centers.

Gesture Cluster Analyses

The server records a set of statistics for each gesture cluster such as the number of gestures in the cluster and the number of unique users who provided gestures for the cluster. These statistics allow the system to more effectively evaluate the popularity of each cluster. Currently our simple model is to rank the clusters based on the number of users, and to use the number of gestures only when there is a tie. As the system collects more usage statistics, one area for future work is exploring more sophisticated models for predicting the popularity of a gesture cluster.

Combining Crowd and User's Own Gestures

Gesture Marks uses the same representation and matching algorithm for both gestures drawn by the user as well as gestures drawn by the crowd. However, since the user's own gestures are more likely to be relevant than those from others, we add a bias to the system so that it slightly favors these user gestures. This bias was determined empirically based on analysis of the similarity scores from the exploratory study.

Inferring Shortcuts with Handwriting Recognition

As we discussed earlier, a significant portion of gesture shortcuts were based on characters. We employ a handwriting recognizer in Gesture Marks so that these types of gestures can be recognized even if they have never been seen before by the system. While handwriting recognition is not our main contribution, it is an important component in our gesture bootstrapping approach and we discuss it in detail here for readers to better understand how our system can be implemented.

Touch-screen handwriting gestures have many special properties, such as finger-based input, limited drawing area, limited computational resources, and a language model consisting of quick mnemonic shortcuts instead of complete dictionary words. To address these challenges, we developed a specialized handwriting recognizer for Gesture Marks that has the following important properties:

- It is tailored to finger-based input and trained on letters and words collected on touch screen smartphones.
- It uses a language model that is specially designed for mobile shortcut navigation.
- It supports natural multi-stroke letters as well as mixed cursive handwriting.
- It does not require character level labels for training.
- It is fast enough to run in real time on a smartphone.

In order to support natural cursive handwriting, our recognizer needs to accomplish two tasks: character segmentation (i.e., determining where one character ends and the next begins) and recognition (i.e., inferring the identity of each segmented character). We will discuss both of these in turn.

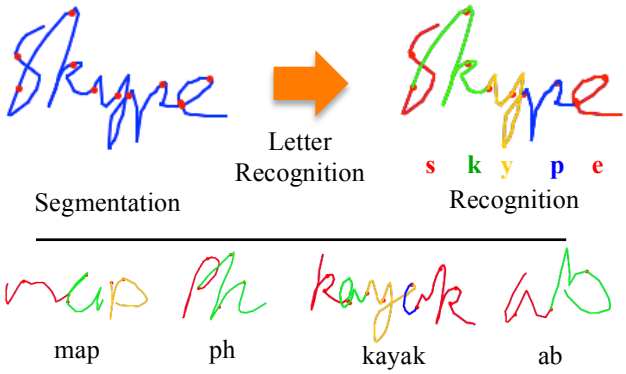

Figure 8: (top) An illustration of the segmentation and recognition approach in our handwriting recognizer. (bottom) Examples of handwriting gestures our system recognized correctly as well as their inferred character segmentations.

Character Segmentation

In the case of cursive handwriting a single character may be drawn with multiple strokes and a single stroke may contain multiple characters (we define each stroke as the sequence of points drawn by the user between a touch-down and touch-up event). Our approach to segmentation starts by splitting all of the strokes in the gesture at points that correspond to local minima and maxima in the y-axis. The result of this splitting process is a sequence of stroke fragments as shown in Figure 8.

The next task for the recognizer is to determine how the individual stroke fragments combine to form the characters in the word. We accomplish this task by applying our character recognizer (described in the next section) on all possible groups of up to k temporally contiguous stroke fragments (in our implementation we empirically set $k = 5$). Next we determine which of these groups most likely correspond to intended characters. Let $\boldsymbol{G}$ be a sequence of n groups $\{G_1, G_2, \ldots, G_n\}$ and $\boldsymbol{L}$ be a sequence of character labels $\{L_1, L_2, \ldots, L_n\}$ representing one possible labeling for the groups in $\boldsymbol{G}$. Then the word score for the sequence is defined as:

$$W(\boldsymbol{G}, \boldsymbol{L}) = \sum_{i=1}^{n} \log(f(G_i, L_i))\, n_f(G_i)$$

where $f(G_i, L_i)$ is the character score for assigning group G_i the label L_i and $n_f(G_i)$ is the number of fragments in group G_i (this is needed to prevent the system from biasing interpretations that contain fewer groups). We use beam search to find the n-best sequences that maximize W.

Recognition

We build a letter recognition model that employs the same visual feature images that we use for similarity based gesture matching. We also incorporate a set of additional features that capture the temporal pattern of the candidate group. These consist of the position, direction, and curvature of the gesture at n equally spaced touch points

sampled along the path of the gesture. Finally, we add a set of geometric features such as the number of fragments, the number of strokes, the aspect ratio, and relative size of the candidate group compared to the complete gesture.

We use a support vector machine (SVM) [4] with a linear kernel to classify each group, based on the set of features listed above. We chose the linear kernel because it allows for efficient constant time classification, and only requires that we store a weight vector for each class rather than the full set of support vectors.

The training data for the SVM consists of both positive examples (correctly segmented characters) as well as negative examples (mis-segmented fragments). Training is done via an efficient linear SVM algorithm based on gradient descent in the dual space [9]. In order to produce probability estimates we fit a sigmoid function over the SVM decision values.

Training using Forced Alignment

One important feature of our handwriting recognition approach is that it does not require character level labels for the words in the training set. This means that the developer does not need to manually label the beginnings and endings of all the characters in the word (e.g., that fragments 1-2 in Figure 8 form an "s", etc.). She only needs to provide the whole word label (e.g., "skype").

To extract and learn character models from these partially labeled words we employ an EM based approach [6] that is similar to the forced alignment process in natural language transcription. During training the recognizer first builds a letter model using only the isolated character examples in the training set. Next, it applies this character model to the whole words in the training set, inferring the correct segmentation as the one that maximizes the word sequence score W. This process relies on the fact that since we know the whole word label, we can force the system to only consider those character sequences where the labeling $\boldsymbol{L}$ matches the ground truth.

Once we have the approximate character boundaries, we can re-train our letter model with these inferred cursive characters in addition to the isolated characters from before. We can then use this updated character model to refine the approximate character boundaries even further, repeating the process across multiple iterations until performance on a validation set ceases to improve.

The Language Model

To improve accuracy and speed we limit the space of possible word interpretations to the set of partial words and initials found in the targets on the user's device. For example, these would include "a", "an", "b", "bi", "ab", etc., for the target Angry Birds. The result of the handwriting recognition process is a ranked list of word predictions, each of which is mapped to one or more targets (e.g., "ma" to both Maps and Market).

Combining Predictions

Our system generates predictions using a combination of two distinct methods: template-based gesture matching and handwriting recognition. We need to merge the two sets of predictions to build the final list of targets presented to the user. Unfortunately, this is not a trivial task because we cannot compare the results of the two recognizers directly. The former is a similarity distance and the latter is a word sequence score.

In this initial implementation of our system, we employed a simple but robust approach to combining these two sets of predictions. This method relies only on the rank of a given target in either prediction list. We assign a rank score to each target as $rs = 1/r_g + 1/r_h$, where r_g is the rank of the target given by the gesture similarity recognizer and r_h is the rank given by the handwriting recognizer. This ensures that the system presents the user with high-ranking matches from both gesture similarity matching and handwriting recognition, and favors most those that are ranked highly by both.

EVALUATIONS

We conducted a user study to understand how users interact with our system. The goals of the study were to learn:

- How often the system is able to correctly predict the destination for a gesture.
- The sources contributing to the correct prediction: crowd gestures, user's own gestures, or handwriting recognition.
- How the prediction accuracy and the role of each source evolve as users use the system.

Experimental Procedures

For this study we asked 22 participants (4 female) to use our system to launch a series of targets (websites and applications) using gestures. Their ages ranged from 19 to 35 (mean = 29) and their occupations included engineers, students, researchers, and business analysts. First, users were asked to pick out a set of 40 targets that they use most frequently from a list of 80 popular applications and websites. The list contained the most common targets from the previous exploratory study.

Next, the system prompted the users with each of the selected targets in random order, asking them to draw a gesture to launch the target in question. The system then tried to predict the intended target of the gesture using its combined template and handwriting recognizer, displaying a set of 8 best matches to the user. If the correct target was in this list, the participant was asked to click on it, and the system moved on to the next target.

The goal of the evaluation was not to test how well users could guess the most popular gesture for a given target, so we asked users to draw whatever gesture made sense to them. If the system did not predict the correct target among its list of top 8 matches, the participant was told to select the target manually from the full list of applications and

websites. The series of targets was repeated 4 times, to simulate users revisiting the same target.

To guarantee that the conditions across users remained constant, we fixed the crowd-based gesture library to contain only the gestures collected before the start of the study. We also fixed the list of applications and websites for this study to match the 80 most popular targets found in the first exploratory study (a total of 50 apps and 30 websites). The resulting crowd-based library contained a total of 414 gesture clusters covering the set of 80 targets in the study. Of course, in the real world users will have a much wider range of targets they may wish to visit, including less popular links for which there are fewer crowd-based gestures. However, in a real world deployment the crowd-based gesture library will also be much larger than the one used during the study, and will be continually expanding over time. Note that while users were prompted to draw gestures for their 40 selected targets, the system predictions spanned the whole 80 targets.

Experimental Results

In this section we present the results of our evaluation, looking at prediction accuracy, the contribution of each source to the correct prediction, and user impressions while using our interface.

Prediction Accuracy

We first wanted to analyze how often our system was able to predict the correct target for the user's gesture in the list of top 8 matches. In Figure 9, the "Manually Defined" line shows the number of gestures that the subject had to define manually because the system did not display it in the list, over each of the 4 repetitions. It shows that the users needed to manually define only 23% percent of the targets in the initial round. This means that the system was able to predict the correct target based on the user's gesture 77% of the time, even before it had seen any of the user's own gestures for the targets. By the final repetition, users manually defined less than 1% of their gestures, while the system correctly predicted the target over 99% of the time.

Contribution of Each Source to the Correct Prediction

For the cases where the system did predict the correct target to a gesture query, we also recorded the rank given by both the template-based recognizer and the handwriting recognizer. Furthermore, in the template-based case we recorded whether the correct match came from the user's own gesture library or the crowd's. Figure 9 plots how frequently each of the 3 sources (user gesture, crowd gesture, or handwriting) ranked the correct target the lowest (i.e., was the most informative). This was done for each of the 4 repetitions. In the event of a tie we applied partial counts, e.g., 1/2 to the handwriting recognizer and 1/2 to the user gestures.

At the beginning (during the initial round), because users had not drawn any gestures, the contribution of the user's own gestures (the red plot in Figure 9) was zero.

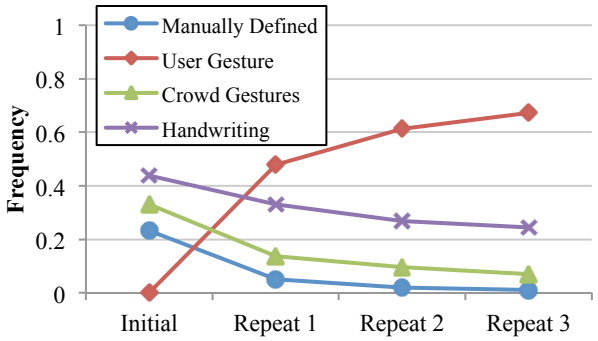

Figure 9: Frequency that each source was the most informative for the correct prediction.

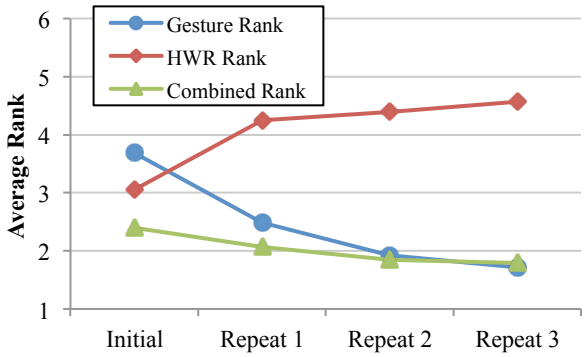

Figure 10: The average rank of the correct target using only gesture similarity, only handwriting, and the average final combined rank (lower means the correct target is closer to the top choice and more accurate).

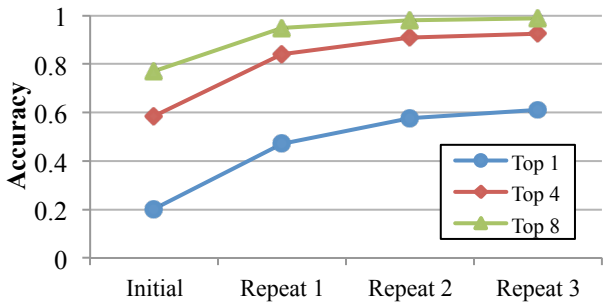

Figure 11: The prediction accuracy when considering the top 1, 4, and 8 matches.

Handwriting recognition and crowd gestures were the major contributors. Handwriting recognition gave the correct target with the lowest rank 44% of the time, while crowd gesture matching did so 33% of the time.

In the second repetition, the most useful source was already the user's own gestures, as all the gestures drawn in the initial round were recorded as templates. This source ranked the correct target the lowest 48% of the time. By the last repetition, this frequency increased to 67%. This suggests that users rapidly converged to a small set of possible gestures for a majority of the targets, and that our system was capable of quickly and effectively learning these gesture patterns.

Gesture Rank

In addition to looking at which source was the most informative, we also examined the average rank given to the correct target using only the template-based recognizer, only the handwriting recognizer, and using the combination of the two. Figure 10 plots these average ranks across the 4 rounds of the study. We can see that especially in the earlier rounds, the combined rank is noticeably lower than the rank from either gesture similarity or handwriting recognition alone. This demonstrates that there is significant benefit in combining the two sources. Furthermore, we found that the gesture rank and combined rank both converged to about 1.7 by the last repetition, meaning that the system was no longer relying on the handwriting recognizer at this stage and that the correct match was frequently ranked first or second in the list.

Interestingly, we also discovered that the average rank provided by the handwriting recognizer actually got slightly worse with each repetition. We suspect that one reason for this is due to the fact that users became faster and messier as they revisited the same targets, and became more confident as they noticed that the system could accurately recognize their particular handwriting or drawing style. As we can see from Figure 9, this accuracy was mostly due to the visual similarity to the user's own previously provided gesture examples.

Prediction Ranking

Finally, we also analyzed how often the system predicted the correct target in the top 1, 4, and 8 matches. This helps us understand how the system would perform in a different interface that provides a smaller list of predictions. The results in Figure 11 show that in the initial round our system was able to predict the correct target as the first choice 20% of the time and include it in the top-8 list 77% of the time. These numbers increase to 61% and 99% respectively by the last repetition. Note that one reason why the top-4 and top-8 accuracies are significantly higher than top-1 is likely due to mapping ambiguity, where a user draws the same gesture for multiple targets. By displaying multiple predictions, the system can retrieve all of the targets that map to a given gesture.

User Feedback

We asked all of the participants in the study to answer a survey afterwards about their experience using Gesture Marks. We asked users to rate whether they thought the system would be useful for them, whether it was accurate in predicting the intended target for a gesture, and whether it became more accurate with use. All of the ratings were on a 5-point Likert scale with 1 for "strongly disagree" and 5 for "strongly agree". A majority of users agreed that Gesture Marks would be useful (median=4) and was accurate (median=4), and strongly agreed that the accuracy improved with use (median=5).

Users mentioned in their comments that they liked the speed and accuracy of the system, and it's ability to handle arbitrary gestures: *"I liked that it learned quickly about what I wanted it to display, even if I picked something wacky", "liked -- got pretty accurate as time progressed. intuitive, simple to use."*, and *"I really like the simplicity in this program that allows you to access applications and websites that on a regular task may take you some time to find amongst all your apps."*

Several users also mentioned that they might have trouble remembering custom gesture for a large number of targets. One user came up with a solution to the recallibility problem by using the gestures to specify categories of targets: *"i tend to use the gesture for a category of things, like m (as in mail) for gmail and other mail apps."* Another user liked that fact that he could filter the list of targets quickly using a single letter.

One common criticism for Gesture Marks was that handwriting gestures did not work for some abbreviations such as "kyk" for Kayak or "m" for Gmail, since the language model employed by the handwriting recognizer only matched partial words or initials.

RELATED WORK

There is a long history of gesture and sketched based interfaces in the HCI community. Touch gestures have been used for searching mobile devices [14] and for creating custom shortcuts to invoke commands [2,8,12,13], bypassing the standard hierarchical menu structure with a quick and natural gesture.

Search with alphabet-based gesture shortcuts leverages the mnemonic relationship between gestures and targets (e.g., [14]), alleviating the effort for both defining and recalling gestures. However, they are limited because they are constrained to only character-based gestures, do not adequately adapt to a particular user's drawing style, and can be less efficient for accessing a large corpus such as the web. Other systems allow users to associate an alphabet gesture with a specific task (e.g., [12,21]), but the process of manually defining gesture mappings or learning larges sets of predefined ones can be tedious.

In contrast, Gesture Marks addresses these problems with gesture shortcuts using the following features. 1) It allows arbitrary drawing gestures in addition to handwritten characters. 2) It supports mixed cursive handwriting that is based on a language model tailored for shortcuts. 3) It contains a crowd-based gesture library that records, analyzes, and shares popular gestures among the user population. 4) It adapts to the user over time by learning his or her particular drawing or handwriting style.

Although substantial work [14,15,19,20] has been conducted on gesture recognition, little effort has been devoted to the situation when little or no data is available, or on how to continuously improve recognition over time. In our work we explored both of these challenges and contributed two bootstrapping approaches based on the "wisdom of the crowd" and handwriting recognition.

Morris et al. [16] elicited a set of gestures for interactive surfaces and revealed that participants preferred gestures authored by larger groups of people. Similarly, our crowdsourcing component is built on the assumption that popular gestures tend to be preferred by users. In contrast to their work, Gesture Marks automatically learns and dynamically evolves these gestures as users interact with the system instead of relying on researchers to manually distill a favorable gesture set from the users.

Handwriting recognition is a well-studied area of research that has been used in many mobile platforms such as the Palm Pilot, Apple Newton, and Microsoft Windows Mobile. Previous work on handwriting recognition has focused mainly on isolated characters [3,5] or complete lines of text consisting of dictionary words [10,18]. In contrast, Gesture Marks employs its own specialized handwriting recognizer that is tailored to the types of gestures people draw on touch screen devices. It also uses a dynamic language model that is customized for the set of targets available on each individual's device.

CONCLUSIONS AND FUTURE WORK

This paper presented Gesture Marks, a novel approach to touch-gesture interaction that allows a user to access applications and websites using natural gestures without having to define them beforehand. Our initial study contributed new knowledge on user-defined gesture shortcuts. It investigated the types of gestures people tend to draw and analyzed the visual and semantic consistency of gesture shortcuts, both from the same user and across users. We presented two approaches for bootstrapping user-defined gesture shortcuts, leveraging gestures drawn by the crowd and employing a specialized handwriting recognizer. An evaluation showed that our system was able to achieve a high level of accuracy even before a user had defined any of his or her own gestures, and was able to quickly learn a user's drawing patterns over time.

This paper focused on navigating applications and websites because these are two types of targets that are very likely to be shared across users. For more personal information like contacts, crowd-based gestures may become less useful. In that case the system will need to rely more on user-defined gestures and handwriting recognition. In the future we intend to support these and other types of targets such as phone settings and music tracks. We will also start to deploy Gesture Marks extensively to more participants for longitudinal studies, to find out how the system and user behaviors evolve over a longer period of time.

REFERENCES

1. Android. http://www.android.com/.
2. Appert, C. and Zhai, S., Using strokes as command shortcuts: cognitive benefits and toolkit support. In *CHI 2009*, 2289-2298.
3. Bahlmann, C., Haasdonk, B., Burkhardt, H., On-Line Handwriting Recognition with Support Vector Machines; A Kernel Approach. In *IWFHR 2002*, 49-54.
4. Boser, B., Guyon, I. M., and Vapnik, V.N. 1992. A training algorithm for optimal margin classifiers. In *COLT 1992*, 144-152.
5. Connell, S.D., and Jain, A.K. Template-based online character recognition. In *Pattern Recognition 2001*, 1-14.
6. Dempster, A, pp., Laird, N. M., Rubin, D. B. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society* 1977, 1-38.
7. Guha, S, Rastogim R, and Shim, K. CURE: an efficient clustering algorithm for large databases. In *SIGMOD 1998*, 73-84.
8. Guimbretiere, F. and Winogra, T., FlowMenu: Combining Command, Text and Parameter Entry. In *UIST 2000*, 213-216.
9. Hsieh, C., Hsieh, K., Chang, C., Lin, S., S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *ICML 2008*, 408-415.
10. Hu, J., Gek, Brown, M. Writer independent on-line handwriting recognition using an HMM approach. In *Pattern Recognition 2000*, 133-147.
11. iPhone. http://www.apple.com/iphone/.
12. Kristensson, P.O. and Zhai, S. Command strokes with and without preview: using pen gestures on keyboard for command selection. In *CHI 2007*, 1137-1146.
13. Kurtenbach, G. and Buxton, W. The limits of expert performance using hierarchical marking menus. In *CHI 1993*, 482-487.
14. Li, Y. Gesture Search: A Tool for Fast Mobile Data Access. In *UIST 2010*, 87-96.
15. Li, Y. Protractor: A Fast and Accurate Gesture Recognizer. In *CHI 2010*, 2169-2172.
16. Morris, M.R., Wobbrock, J.O. and Wilson, A.D. Understanding users' preferences for surface gestures. In *Graphics Interface 2010*, 261-268.
17. Ouyang, T and Davis, R. A Visual Approach to Sketched Symbol Recognition. In *IJCAI 2009*, 1463-1468.
18. Plamondon, R. and Srihari, S.N. Online and off-line handwriting recognition: a comprehensive survey. In *PAMI 2000*, 63-84.
19. Wobbrock, J.O., Wilson, A.D., and Li, Y. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *UIST 2007*, 159-168.
20. Anthony, L., Wobbrock, J.O. A Lightweight Multistroke Recognizer for User Interface Prototypes. In *Graphics Interface 2010*, 245-252.
21. Zhai, S. and Kristensson, P.O. Shorthand writing on stylus keyboard. In *CHI 2003*, 97-104.