Gesture Search: A Tool for Fast Mobile Data Access

Yang Li
Google Research
1600 Amphitheatre Parkway
Mountain View, CA 94043
yangli@acm.org

ABSTRACT
Modern mobile phones can store a large amount of data, such as contacts, applications and music. However, it is difficult to access specific data items via existing mobile user interfaces. In this paper, we present Gesture Search, a tool that allows a user to quickly access various data items on a mobile phone by drawing gestures on its touch screen. Gesture Search contributes a unique way of combining gesture-based interaction and search for fast mobile data access. It also demonstrates a novel approach for coupling gestures with standard GUI interaction. A real world deployment with mobile phone users showed that Gesture Search enabled fast, easy access to mobile data in their day-to-day lives. Gesture Search has been released to public and is currently in use by hundreds of thousands of mobile users. It was rated positively by users, with a mean of 4.5 out of 5 for over 5000 ratings.

Author Keywords
Gesture-based interaction, search, mobile computing, shortcuts, Hidden Markov Models.

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 increasing computing power and storage, the potential use of mobile phones goes far beyond their original purpose for making phone calls. However, this potential is hampered by existing mobile user interfaces. In particular, a mobile device such as an iPhone [4] or an Android-powered device [1] often has a large amount of data that a user frequently accesses, such as contacts, applications or music. However, it is difficult and time-consuming to locate a target item with current mobile interfaces. A user often needs to navigate through a deep interface hierarchy and manually scan a large collection of items. The small screen of mobile phones as well as the lack of a keyboard makes the data access even harder.

Similar to desktop search utilities for PCs (e.g., [5, 6]), keyword-based search tools have been developed for mobile phones, in which a user finds various types of data items by entering a textual query at a single place, e.g., the Quick Search Box on Android phones [2]. Although these search tools avoid the hassle of navigating in a mobile interface, entering a search query requires a user to scan and type on tiny keys in a keypad, which can be stressful and as hard as manually locating the target item on the device. Search with voice input is promising but not always socially acceptable, especially in public, quiet situations. In addition, voice-based interaction can be inefficient when the recognition or search results are wrong and revision is needed.

Alternatively, touch screen gestures have been employed as shortcuts for invoking commands or accessing frequently used data items (e.g., [12, 14]). Gestures are easy to input and rich in semantics. Users can associate a gesture with a target data item and then activate the item by drawing a similar gesture later (e.g., [12]). However, traditional gesture shortcuts suffer several problems. First, users need to explicitly create or learn the mapping from a gesture to a data item, which a mobile phone user is often less motivated to spend the effort. Second, as the number of shortcuts increases, it is difficult for the system to accurately match or recognize gestures, and it also becomes challeng-

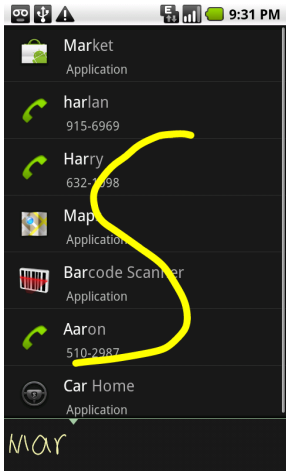

Figure 1. Gesture Search allows a user to quickly access contacts, bookmarks, applications and music tracks by drawing gestures.

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.
UIST'10, October 3–6, 2010, New York, New York, USA.
Copyright 2010 ACM 978-1-4503-0271-5/10/10....$10.00.

ing for users to learn and recall. Command Strokes by Kristensson and Zhai [8] made great progress in addressing these issues. However, it still requires a soft keypad to be visible on the screen, which consumes the mobile phone's limited screen real estate. In addition, a user first learning to use Command Strokes needs to scan and locate the correct keys on the keypad. It takes time for users to learn ShapeWriter shortcuts [14] that Command Strokes are based on before they can gain speed.

To address these issues, we created Gesture Search (see Figure 1), a tool that allows users to quickly access mobile phone data such as contacts, applications, bookmarks and music tracks by drawing gestures[1]. It demonstrates a novel approach in combining search and gesture-based interaction. To find a data item, a user draws gestures anywhere on the touch screen. Gesture Search finds relevant items by simultaneously matching multiple possible interpretations of a gesture sequence against the target dataset. Based on the user's search history, Gesture Search continuously optimizes search performance to allow even quicker data access with shorter gesture sequences.

Since a gesture query is a sequence of regular character gestures that is mnemonically related to the target data items, e.g., "Qu" for "Stock Quote", a user does not need to learn how to input these gestures nor recall the mappings from gestures to data items. Instead, Gesture Search actively learns from users' search behaviors and continuously optimizes search performance. As shown in our log analysis on the usage of Gesture Search, users were able to use single-gesture queries to complete 61% of their 5,497 data accesses.

In the remainder of the paper, we first describe how a user uses Gesture Search for accessing their mobile phone data. Next, we dive into the technical details of Gesture Search. We discuss how we implement the novel gesture-based interaction techniques that Gesture Search uses, and our search and optimization algorithms. We then report on a longitudinal user study that we conducted with mobile phone users over a period of three months. Finally, we compare Gesture Search with prior work before we conclude the paper.

THE GESTURE SEARCH SYSTEM

To use Gesture Search, a user draws a sequence of gestures on the touch screen. Gesture Search leverages the entire touch screen for gesture input (see Figure 2).

An Introductory Example

Let us assume a user, Bill, wants to call his friend Anne. Bill first draws letter "A" on the screen. Once the gesture is finished, Gesture Search automatically starts a search process. Since the gesture is ambiguous to Gesture Search — it looks similar to both the letters "H" and "A" — Gesture

[1] Gestures here are handwritten letters or numbers. Thus, we use gesture and handwriting recognition interchangeably.

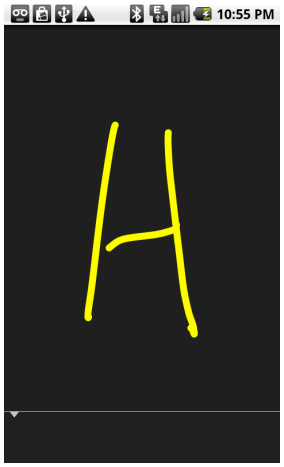

Figure 2. A user draws capital letter A on the screen using her finger. Gesture Search uses a timeout to delimit multi-stroke gestures.

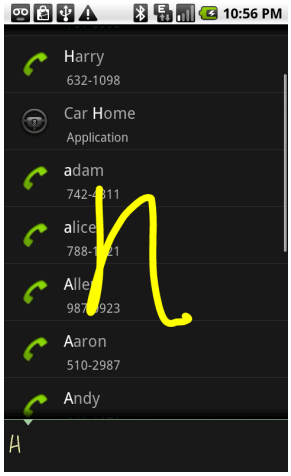

Figure 3. A scaled-down version of the gesture in Figure 2 is shown in the gesture query field at the bottom of the screen. The user then adds the second letter by drawing the letter directly on top of the research result list.

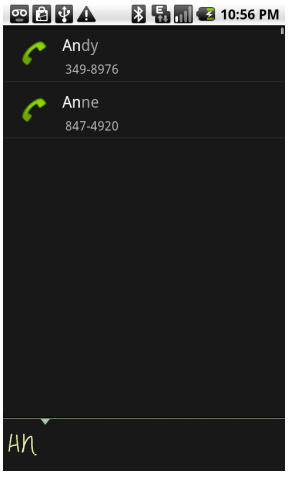

Figure 4. With the two gestures, the contact of Anne pops up at the second place of the list.

Search returns data items that match either of these letters (see Figure 3). Meanwhile, a scaled-down version of the original gesture is shown at the bottom of the screen, while possible recognitions of the gesture are displayed in the context of search results, e.g., "H" is highlighted in "Car Home" and "A" in "Aaron". These highlights explain to the user why these data items are included in the search results.

At this point, Bill can scroll down the list to find "Anne" or keep adding more gestures by drawing directly on top of the search result list. Gesture Search automatically differentiates gestures from scrolling by analyzing the trajectories of the user's touch traces. We will discuss this in more detail in the next section. Here the user draws a second letter, "n" (see Figure 3).

Gesture Search immediately revises the search result list as each gesture is drawn. With the two gestures inputted, Anne's contact is shown at the second place in the list (see Figure 4). Bill can click on the item to see Anne's contact details or tap on the green phone icon to call her directly.

In case the user needs to redraw a gesture or the entire query, he or she can wipe horizontally in the gesture query field at the bottom of the screen. Wiping from right to left removes the last gesture in the query and doing that in the opposite direction clears out the entire query.

As shown in this example, Gesture Search uses all of the possible interpretations of a gesture query to match against each data item; the user does not need to draw the entire name of a target item before matching starts. To make a search query more likely to form a unique match, a user can draw a multiple-prefix gesture query by delimiting characters with spaces, e.g., using "B S" to match a target application named "Barcode Scanner". A user draws a horizontal line on top of the search result list to add a space.

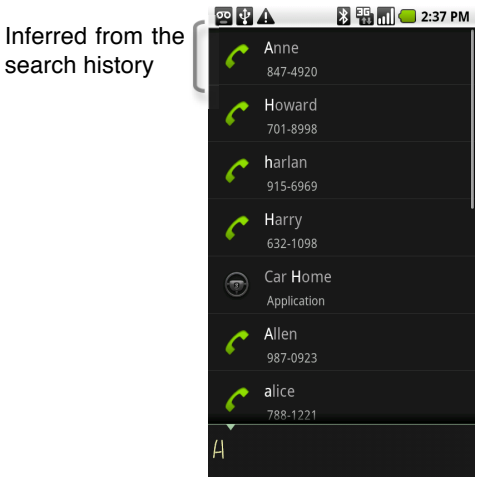

Figure 5. Gesture Search further accelerates data access by learning from the user's search history. Consequently, a user often can access a frequently visited item with one single gesture that is casually or less perfectly drawn.

Searching With Even Shorter Gesture Sequences

A user can quickly search through a large amount of data with short gesture sequences in Gesture Search. To enable even shorter gesture queries, we further optimize the search performance by learning from the user's search history.

For example, when Bill draws the "A"/"H" gesture again, Gesture Search automatically shows "Anne" at the top of the search result list after only receiving one single gesture (see Figure 5). Notice that Gesture Search still considers Bill's gesture to be closer to the letter "H" than "A", so items that match "H" are still ranked above those that match "A". However, Anne's contact is shown as the top result. This is because, in addition to the results of individual character recognition, Gesture Search takes into account the occurrence frequency of mappings from gesture queries to data items in the search history, e.g., Bill previously selected "Anne" after drawing the two-gesture query (see Figure 4).

As a user uses Gesture Search over a period of time, access to a set of frequently visited items becomes more reliable and further accelerated. Since Gesture Search continuously learns from the user's search history, it automatically adapts itself to users' evolving data access needs in their day-to-day lives.

Distinctions with Traditional Handwriting Technologies

Gesture Search employs a handwriting recognition engine to recognize individual gestures. However, it goes beyond simply using recognized results by closely coupling handwriting recognition with search and taking a holistic view to address handwriting and search query ambiguity. The novelty of our approach is reflected both in the UI and underlying algorithms of Gesture Search. We will discuss the differences in more detail in the related work section.

INTERACTION TECHNIQUES & ALGORITHMS

Here we discuss how Gesture Search works by elaborating on its interaction techniques and algorithms. To address the small form factor of mobile phones, Gesture Search maximizes the gesture input area by using the entire screen for gesture input and by overlaying a gesture input layer on top of the search result list. This brings up the challenge of how to effectively separate gestures from regular touch events for a list widget such as tapping, scrolling and flicking, which we discuss first. We then discuss how Gesture Search searches with a probabilistic distribution of a gesture query and how it optimizes searches by learning from a user's search history.

Separating Gesturing from GUI Touch Events

Gesture Search overlays gesture input over the search result list and allows modeless input of gestures and GUI events such as scrolling and tapping to select. It does not require a user to explicitly signal the system for drawing a gesture or manipulating the list widget.

When it is uncertain whether touch events are making up a gesture, the gesture overlay dispatches touch events to the

list widget for immediate interface feedback (e.g., scrolling the search result list as it normally does). Meanwhile, Gesture Search buffers existing touch events and looks ahead for more events that might form a gesture. Gesture Search also renders collected events as translucent, less obvious yellow traces on the list to indicate that these strokes are still being evaluated to potentially form a gesture.

Once the gesture overlay determines that the user is drawing a gesture instead of scrolling, it turns the traces bright yellow, which is used for showing a gesture (see Figure 1). At the same time, Gesture Search stops dispatching touch events to the search result list underneath. As a result, the list stops scrolling, giving the user a static background for drawing the rest of the gesture.

Typical events for the list widget include tapping to select, flicking and scrolling. To separate these events and gestures in a modeless way and as early as possible when users slide their finger on a touch screen, we need to explore the difference between their touch traces.

Hypotheses. Intuitively, GUI-oriented touch input should have less variation in its trajectory than gestures. This difference is rooted in the different interaction styles that GUI and gesture-based interactions support. GUI operations are mostly embodied by visual widgets that a user can manipulate, which often requires a simple motion, e.g., acquiring a target. However, the interaction semantics of gesture-based interactions are mostly represented by the shape or the motion of a gesture. As a result, gesture-based interactions require rich variations in touch input trajectories to represent different gestures for various interaction semantics.

To investigate this hypothesis, we collected GUI events on touch screen mobile devices and compared them against gesture data. For our analysis to be applied broadly, our data collection and analysis went beyond character gestures and list widgets that are used in Gesture Search.

Data Collection. We asked seven participants to perform a set of GUI interaction tasks on an Android phone that involved finding four contacts in a contact list that has hundreds of items and locating two locations on the Google Map (e.g., finding San Francisco). The participants were not informed of the purpose of the study and were asked to perform these tasks as they would normally do. All the participants were regular Android phone users. These tasks were designed in a way so that tapping, flicking, scrolling and panning could naturally occur. In total, over 560 touch trajectories for these GUI interactions were collected.

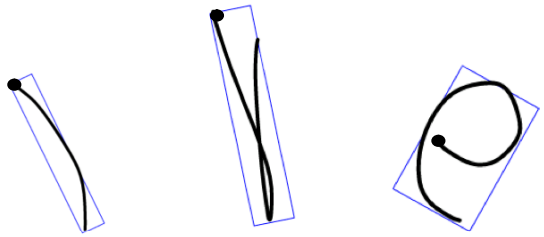

Figure 6. From the left to the right are three example traces for flicking down, scrolling down and then up, and alphabet gesture *e*. A minimum bounding box for each trace is shown. The dot attached to each trace represents the starting point of the trace.

Our analysis of gesture data included both user-defined and alphabet gestures. For user-defined gestures, we reused previously published data [13], which includes 4800 samples for 16 single-stroke gesture categories from 10 users. We also collected 14,269 alphabet gesture samples for Latin letters written by over 100 users. These users opted into contributing their data on their own mobile devices using a remote data collection application.

Data Analysis & Classification Methods. As expected, all of the taps had trajectories with much smaller bounding boxes than gestures. Consequently, when the bounding box for the trajectory of a touch input being drawn is smaller than a threshold, we do not consider it to be a gesture. However, this rule breaks when multi-stroke alphabet gestures are allowed. For example, a user might draw a dot first to complete a two-stroke gesture *j*, and the dot will conflict with a tap event. To address this ambiguity, when both tapping and dots are possible, we delay dispatching the tap event by a short amount of time to give the user an opportunity to add more strokes.

When the bounding box of a trajectory is larger than a threshold, we consider the touch input to be potentially either a gesture or a GUI manipulation such as scrolling, flicking and panning. A GUI input could have more variations than we expect. For example, our data shows that a simple flicking of a list widget often does not generate a straight path (see the leftmost screenshot in Figure 6). This is especially true when a user uses a single hand to operate a phone, e.g., flicking on the touch screen with the thumb while holding the device with the same hand. Scrolling and panning can also result in complex trajectories. For example, a user might scroll a list up and down while looking for an item (see the middle example in Figure 6).

However, by examining the touch input data that we collected, we found that although GUI touch input may also result in complex trajectories, most of them had a narrower bounding box than gestures did (see Figure 6). To formalize this observation, we use the *squareness* as the measure of how narrow a bounding box is. We define the squareness as the minimum ratio of two adjacent edges of the bounding box of a touch trajectory. This quantity is between 0 and 1. The larger the squareness of a touch trajectory is, the more its bounding box looks like a square. To precisely capture the squareness of a trajectory, we use its minimum bounding box instead of its orthogonal bounding box.

The squareness distributions of the collected data are illustrated in Figure 7. Note that taps are not included, and flicking, scrolling and panning are analyzed together as they had a similar distribution. User-defined gestures (4800 samples) were analyzed in the same probability space as

alphabet gestures (14,269 samples) but have a significantly smaller sample size. To avoid the effect of user-defined gestures being washed out, we give each sample of user-defined gestures more weight than alphabet gestures to balance their different sample sizes.

Based on these two empirical probability density functions, we can choose the squareness threshold, 0.275, to minimize the false positive (1%) and negative (4%) rates. This means when the squareness of a trajectory is larger than this threshold, we consider it to be a gesture. Based on this threshold in Figure 7, over half of the gesture samples were determined when one-tenth of the gesture was drawn (took roughly less than 250ms), and over 80% of gesture samples were detected when half of a gesture was drawn.

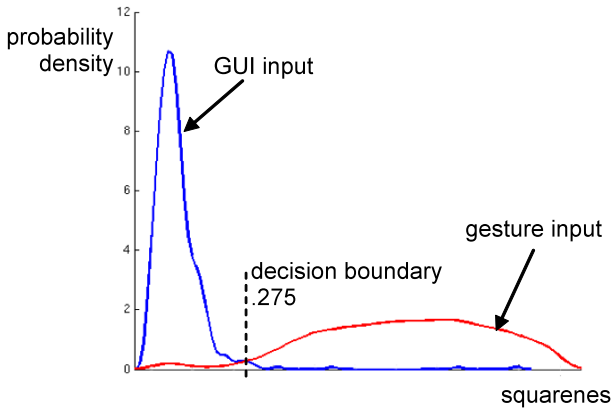

Figure 7. The red, flatter curve shows the empirical probability density function of gesture touch input. The blue, narrower curve shows that of GUI input, including scrolling, flicking and panning.

However, not all of the gestures were discernable with the squareness measure. For example, the lowercase letter *l* may have a small squareness measure. We had to use other heuristics (e.g., if the touch input triggers a valid scrolling action for the list given the direction of its trajectory); otherwise, a user would have to write the letter more cursively.

A longitudinal study on Gesture Search indicated that users had no trouble in inputting gestures and manipulating the list. Although the starting of a gesture may be temporarily treated as GUI interaction, the misinterpretation rarely bothered our users since the ambiguity was often resolved fairly quickly (e.g., within 250ms).

Searching with A Gesture Query

In Gesture Search, a query is a gesture sequence that consists of one or more character gestures or spaces. The search process involves several steps. First, a handwriting recognizer recognizes each character gesture in a query, which generates a probabilistic distribution of all of the possible character sequences of the gesture query. Then the sequence distribution is used to match against each of indexed data items. Multiple sequences in the distribution might match an item. A data item is scored based on the matched sequence that has the largest probability. Data items are then ranked based on the match scores and only top *N* results are shown in the search result list.

The search performance of Gesture Search is boosted by its ability to deal with misrecognition of the recognizer and potential misspelling of the user. As Gesture Search uses all of the possible interpretations of a gesture, an intended character does not have to be the best guess of the recognizer. However, Gesture Search constrains the number of low probability interpretations allowed in a match to exclude irrelevant data items from the search result and reduce the search space. The relationship between true and false positive rates was analyzed to determine an optimal threshold below which an interpretation is considered less probable for a gesture. By allowing a certain number of low probability interpretations, Gesture Search can tolerate misspellings in a query.

Our implementation of the search process is extremely optimized so that a search can be completed in real time on a mobile device, even when the sequence distribution is large (e.g., for a long query) and when there are hundreds of data items to match against. As a result, users can search as they draw.

Learning from Search History

Gesture Search optimizes search performance by learning from the user's search history. The optimization accelerates data access in several ways. First, the system can better tolerate the ambiguity of gesture shapes and the inaccuracy of handwriting recognition, so a user does not need to draw gestures as precisely. Second, fewer gestures are needed from the user since the system can better predict target items based on short queries.

Gesture Search updates the search history every time a user clicks on a data item in the search result list (see Table 1). Each row in the search history represents a unique mapping from a recognized query to a target item. A row also maintains when the mapping most recently occurred and the number of occurrences. The query field in a row is the string recognized from the gesture query for matching the selected item.

Last Occurrence	Query	Selected Item	Frequency
8/6/09 4:00pm	A	**A**ndy	2
8/6/09 2:30pm	Sc	**Sc**ott	1
8/5/09 9:00pm	Ba	**Ba**rcode Scanner	1
8/5/09 6:00pm	N Y	**N**ew **Y**ork Times	2
8/4/09 11:00am	Bar	**Bar**code Scanner	1
8/3/09 3:00pm	An	**An**ne	1
8/2/09 2:00pm	Br	**Br**owser	1
8/2/09 10:00am	Sc	Barcode **Sc**anner	1

Table 1: An example of the user's search history.

To address the ambiguity of both handwriting recognition and the mapping from a partially completely query to a data item, Gesture Search dynamically constructs a probabilistic

model from the search history table (see Figure 8), based on a high-order Hidden Markov Model (HMM) [11]. In the graphical model, C_i represents the intended character at position i in the gesture query while G_i represents the gesture input at the position.

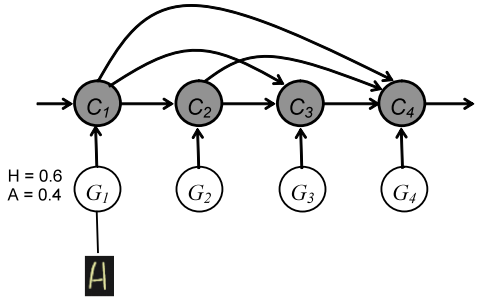

Figure 8. Gesture Search uses an HMM-based approach to predicate a data item given a partially completed gesture query. In this figure, one gesture, A, has been inputted.

The model effectively captures both the ambiguity of handwriting recognition, e.g., p (C_1=H | G_1= A) = 0.6 and p (C_1 =A | G_1= A) = 0.4, and the ambiguity of the intended query, e.g., the probability that "a" leads to "Andy" is 0.67 and that to "Anne" is 0.33 (see Figure 9).

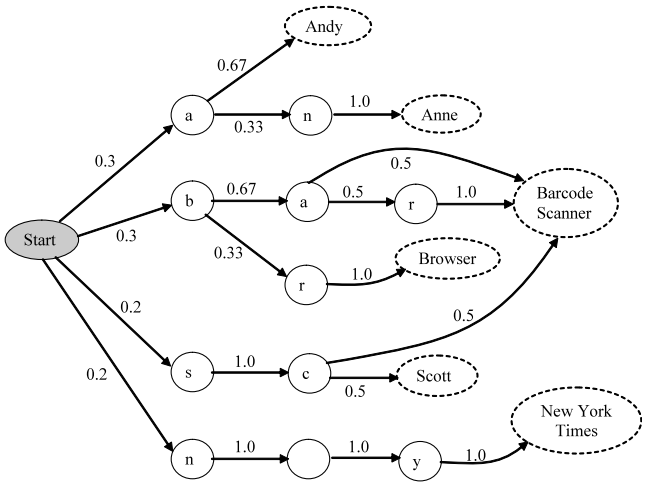

Figure 9. The state transition diagram represents the transition probability distribution manifested in the search history tablet shown in Table 1. Selected data items in the search history table are shown as dashed elliptical nodes.

Given a sequence of k gestures drawn by a user, $g_1,...,g_k$, Gesture Search suggests data item t' as the top result in the search result list if its probability given the gesture sequence is larger than a threshold. The inference process can be formalized in Equations 1 and 2. T denotes all of the selected (or visited) data items in the search history table.

$$t' = \arg\max_{t_i \in T} \underbrace{P(t_i | g_1, \cdots g_k)} \quad (1)$$

$$\sum_{c_1 \in C_1} P(c_1 | g_1) \cdots \sum_{c_k \in C_k} P(c_k | c_1, \cdots, c_{k-1} g_k) P(t_i | c_1, \cdots, c_k) \quad (2)$$

Essentially, the inference process looks at all of the paths from the start node to the target nodes in the transition diagram. Each path is scored by combining the recognition probabilities of the handwriting recognizer and the transition probabilities that are shown in Figure 9. $P(t_i | c_1, \cdots, c_k)$ represents the probability of data item t_i given the prefix sequence $c_1,...,c_k$. Graphically, this is calculated by combining the probabilities of all of the paths in the diagram that are prefixed by $c_1,...,c_k$ and lead to data item t_i.

To make the learning more tractable, we can discard old entries in the search history, e.g., if the last occurrence of an entry was three months ago. We can also let the frequency of an entry gradually decay based on its last occurrence time to give old occurrences less weight.

IMPLEMENTATION

Gesture Search was implemented in Java using Android SDK 2.0 and is compatible with Android 1.6. It has been tested on various Android-powered devices such as the T-Mobile G1, Motorola Droid and Google Nexus One.

LONGITUDINAL USER STUDIES

To understand how Gesture Search performs in users' day-to-day lives for accessing mobile data and to explore the characteristics of gesture-based search queries, we conducted a longitudinal study of Gesture Search with over a hundred of mobile phone users.

Hypotheses

We hypothesized that Gesture Search provides a quick and less stressful way for users to access mobile data. In particular, we intended to find out whether gesture queries with a short length could be used to retrieve a reasonable amount of data items. We also wanted to analyze how well Gesture Search optimized search performance by learning from the search history over time, and how the gesture queries that people use to access the same data items varied depending the complexity of their datasets.

These issues needed to be investigated based on users' search behaviors over an extended period of time. In addition, since Gesture Search helps users access their personal data such as contacts, it would have been unrealistic if we had asked users to search against the same set of data items in a laboratory setting. Thus, we chose to conduct a longitudinal, realistic user study instead of a controlled laboratory study.

Participants

We made Gesture Search downloadable through a company's internal website and asked Android phone users in the company, via recruiting emails, to test out Gesture Search. User participants who opted-in to the study had various backgrounds such as software engineering, human resource and product management. Participants could install and start to use Gesture Search or stop anytime they wanted. As a result, individual users used it for varying amount of time.

Study Design & Logging Infrastructures

In contrast to a controlled laboratory study, we did not instruct users to do a set of predefined tasks. Instead, participants used Gesture Search in their daily life for regular mobile data accesses. At the end, we asked participants to answer an online survey to solicit qualitative feedback.

To collect quantitative data for the usage of Gesture Search, we instrumented an interaction logger in Gesture Search and a web server for receiving and storing the log data. The logs included the size of the dataset that users searched in using Gesture Search (e.g., the number of contacts or applications), users' gesture input, the data items they clicked on, and actions such as deleting gestures or scrolling the list. These logs are automatically collected and periodically uploaded to the remote log server.

Log Analyses

From participants who used Gesture Search, we randomly selected 125 users based on the criteria that a user must have 1) used Gesture Search for at least one month and 2) used Gesture Search at least once per week. The first criterion excluded users who just started to use Gesture Search or dropped out early. Data from short usage users is not effective for investigating our hypotheses. The second criterion filtered out those who only occasionally used Gesture Search, but it included those who used Gesture Search regularly, whether they were mobile phone power users or not.

From 125 users, we collected 5,497 search sessions in which a user drew a gesture query and selected a search result. The percentages of the types of data that users accessed are shown in Figure 10. We found our participants primarily used Gesture Search to access contacts and applications[2].

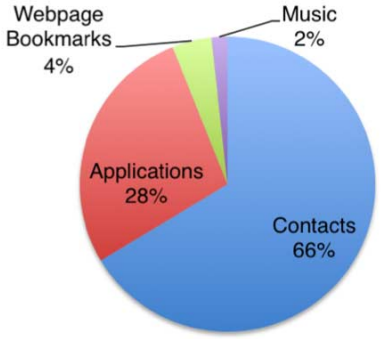

Figure 10. The types of data that participants accessed on their phones using Gesture Search.

We found that the usage of Gesture Search per day varies in users, ranging from less than once to five times per day. In 74% of these sessions, users did not do deletion at all, which indicated that most of the time while using Gesture Search, users accessed their target results only by drawing gestures.

From the log data, we found users were often able to access their data items using a short query (see Figure 11). In particular, 61% of 5,497 queries used a single gesture and 82% of them used two gestures or less.

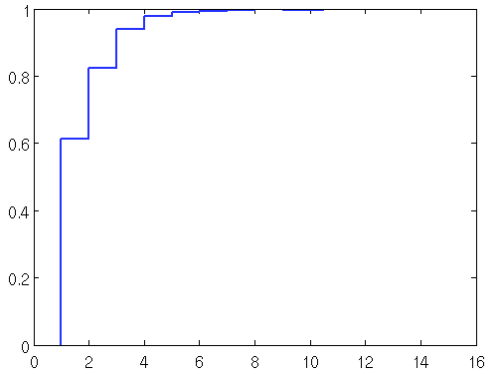

Figure 11. The empirical cumulative distribution function of the lengths of gesture queries based on 5,497 sessions. 82% of queries used two gestures or less.

In 46% of the 5,497 sessions, users picked the top results that were inferred by learning from users' search histories (as the example shown in Figure 5). Within these inferred top-result sessions, 84% of queries issued had a single gesture and 98% had two gestures or less. Without these inferred results, target items in 16% of these sessions would have not been shown in the search result list, given the short query. Target items in 15% of these sessions would have not been shown on the first screen of the search result list, and a user would have had to scroll the list to find them. These false negatives could have been due to misrecognition or the essential ambiguity of a short query — the queries of 90% of these sessions had only a single gesture. So our learning feature greatly improved the search quality of gesture queries and accelerated the access to mobile data.

To study the characteristics of gesture-based search behaviors, we hypothesized that the average length of gesture queries for individual users would vary depending on the complexity of the personal dataset that they search in. To investigate this hypothesis, we looked at two aspects of query length variation. First, we looked at how the query length of each user differed based on the size of the target dataset, i.e., the number of data items that are available on the user's device (see Figure 12a). Second, we looked at whether the lengths varied based on the number of items that a user actually accessed (see Figure 12b). In these two scatter graphs, each circle represents the mean query length of a user given the user's entire dataset size or the number of visited items. Since the query length distribution is heavily negatively skewed, i.e., most queries had a short length, we calculated the mean query length of each user by first calculating the mean of log-transformed query lengths

[2] The kinds of data accessed are not necessarily representative of typical mobile search traffic.

(which is a normal distribution) and then converted the log mean back to the original space.

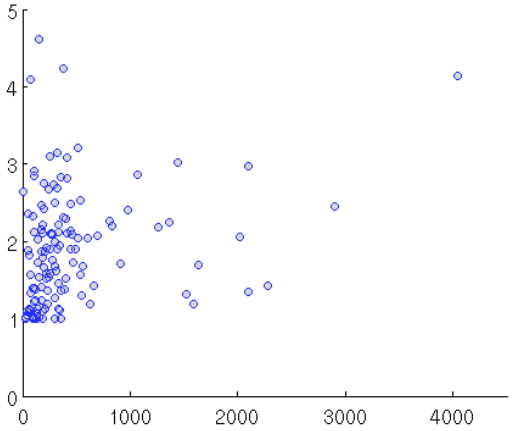

Figure 12a. The mean length of gesture queries of each individual user versus the dataset size of the user.

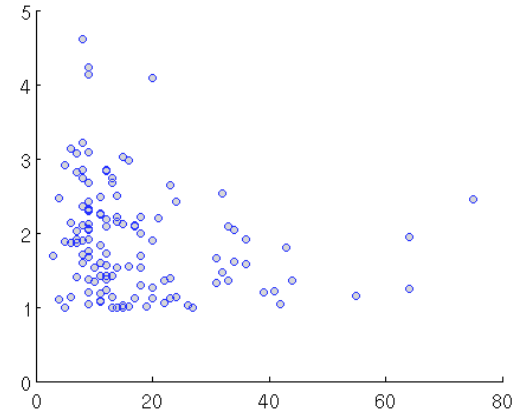

Figure 12b. The mean length of gesture queries of each individual user versus the number of data items that the user actually accessed.

From Figure 12a, we see most of the 125 users had less than 1,000 unique data items to access and the lengths of their queries primarily scattered between 1 and 3. From Figure 12b, we found the number of unique data items that users actually accessed was much smaller than the dataset that was available, although the number of accessed items might have increased if we continued collecting user logs. Pearson correlation tests conducted on both relationships indicated that neither of them was correlated with $R^2<0.08$. The results showed that the complexity of gesture queries did not vary given the users' datasets. This implies that gesture search queries were expressive enough to access a reasonable amount of data items and they have the capacity to deal with datasets more complex than what were logged in the study. In theory, two or single-gesture sequences (letters and/or numbers) can form 1,332 = 36 × 36 + 36 unique queries, although not every instance of them may apply to a user's data.

To better understand how gesture queries were mapped to data items, we further analyzed the relationship between the number of unique queries and the number of unique targets that were accessed by each user. The ratio of the number of unique queries over the number of unique targets reflects the diversity and ambiguity of gesture queries give a target data set size (see Figure 13). A ratio of 1.0 indicates that each unique query leads to a unique target. When the ratio is larger than 1.0, it reflects the diversity of using different queries to access the same target, and when the ratio is smaller than 1.0, it implies ambiguity that a query might be used to access different targets. From Figure 13, we see the ratios obey a normal distribution, falling in a range between 0.8 and 1.4 with an 80% confidence interval and a mean of 1.1. The result implies that users implicitly used a unique gesture query to access a specific data item, although the set of mappings might form gradually over the time. A Pearson correlation test indicated that there is a strong positive correlation between the number of unique queries and the number of unique targets accessed, $R^2=0.88$. The number of unique queries being slightly more than the number of unique targets indicates that users might happen to use different queries to access the same data item.

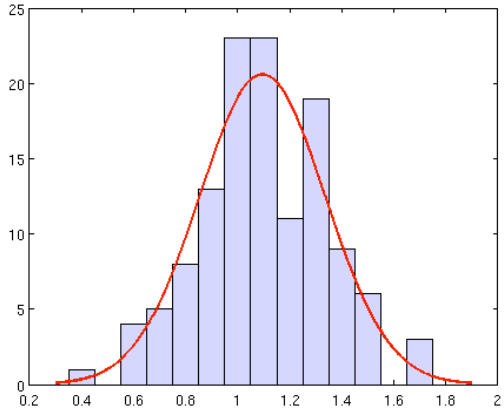

Figure 13. The distribution of the query/target ratio for all users. The graph indicates that with an 80% confidence interval, the ratio falls in the range between 0.8 and 1.4.

As mentioned earlier, the log analysis was to understand users' long-term search behaviors and did not include users who dropped out early. However, the feedback of some of these users is discussed in the next section. Two typical reasons that users dropped out were that they had few items to access and the current way to activate Gesture Search (using a home screen shortcut) is less optimal.

User Feedback

We asked participants to answer a survey after they used Gesture Search. 59 people answered the survey, including some of those who did not participate in the log analysis due to early dropout or insufficient use of Gesture Search. A 5-point Likert scale was used to solicit user feedback on the usefulness and usability of Gesture Search, with 1 for "strongly disagree" and 5 for "strongly agree". The majority of users agreed that Gesture Search is useful and usable (Median=4). Since the answers for these questions in our

Likert Scale are categorical, we calculate medians instead of means and standard deviations.

In terms of the usefulness in accessing each type of data, participants agreed that Gesture Search was useful in accessing contacts (Median=4). This is consistent with what we found in the log analysis, i.e., 66% of search sessions were contact access (see Figure 10). However, participants were less enthusiastic in accessing applications through Gesture Search (Median=3). By analyzing participants' detailed comments, we found 35% of survey participants needed to use only a small number of applications (e.g., less than 5), which they could just activate using home screen shortcuts with a single touch. However, there were still a significant number of participants (17 out of 59) who appreciated the value of Gesture Search in accessing applications, as they used more applications than could fit in the home screen. As more and more applications become available for mobile phones, we can imagine Gesture Search will be more useful for accessing them.

To understand how Gesture Search fit into people's everyday mobile usage, we asked participants to comment on situations where they did not want to use Gesture Search. 78% of users commented that they did not use Gesture Search when they saw the target items on the home screen.

For the top reasons to use Gesture Search, 44% of participants commented that Gesture Search saved them time to find an item. Based on the log analysis, 53% of the accesses took less than three seconds to complete. However, an even bigger advantage of using Gesture Search seems to be the low cognitive load required of users. 59% of participants commented that they liked Gesture Search because it does not require them to type or navigate in user interfaces. As users commented that they could find an item without even looking at the screen.

The major criticism from our users is that the current way of invoking Gesture Search, which is done using a home screen shortcut, can be inconvenient. Our users adopted various ways to make Gesture Search more accessible on Android phones, including adding Gesture Search permanently to the notification bar using a third-party tool so that it can be invoked more easily. Better integration with specific mobile phone platforms will be addressed in our future work.

RELATED WORK

The contribution of Gesture Search lies at the intersection of mobile search UIs interfaces and gesture-based interaction. Thus, we discuss the relationship of Gesture Search with prior work in these areas.

Mobile Search User Interfaces

As mobile devices can store a large amount of data, tools have been developed for searching mobile devices. For example, both Android phones [1] and the iPhone [4] have a phone search feature that allows a user to type in a search query to retrieve various kinds of data items, such as contacts and bookmarks. Similarly, Gesture Search also allows users to access various types of data in a single place and alleviates the effort of navigating the UI.

In contrast to existing search tools, Gesture Search uses hand-drawn gesture strokes as the input modality. From users' feedback, we found drawing strokes to search has a great advantage over typing in a search query using a keypad on the phone. Typing on a keypad requires a user to locate each key, and a user's finger often occludes the tiny keys on the keypad, which makes typing difficult and error-prone. A user also has to switch modes for inputting numbers or characters, since only a limited number of keys can be shown at a time.

Alternative modalities such as voice or handwriting have also been made available for existing search tools. Voice-based search has a great advantage in mobile situations. However, it has two shortcomings. First, speaking to search is not always appropriate, such as in a quiet, public room or in a noisy environment. Second, voice-based interaction can be slow and intractable when interactive correction or revision is needed, e.g., when voice recognition goes wrong.

Handwriting recognition technology has been integrated in mobile phones (e.g., iPhone [4]) as a generic technique for text entry. Obviously, handwriting recognition can also be used to input a search query in existing search tools. However, the existing integration of handwriting technology is limited in several ways. First, the user typically has to write in a dedicated area on the screen, which occupies precious screen real estate. Second, the handwriting must be correctly recognized before being used as a search query. Consequently, a user has to confirm and possibly correct the recognition results. If the recognition is not perfect, the search results might be incorrect.

In contrast, although Gesture Search relies on handwriting recognition technology, it employs a close coupling between handwriting recognition and search. It leverages the entire screen for drawing gestures and seamlessly handles any ambiguity due to misrecognition by the system, imperfect writing or misspelling by the user, or both. This innovation is reflected both in the gesture-based search user interfaces and in the underlying algorithms of Gesture Search. As discussed earlier, 74% of analyzed sessions did not involve rewriting, so there is little need for recognition correction in Gesture Search.

Gesture-Based Interaction

Gesture-based interaction has a long tradition in HCI research as it promises a new modality for interaction. Previously, gesture strokes have been employed as shortcuts for invoking commands, e.g., [3, 7-10, 14]. Gesture shortcuts allow a user to easily articulate a command by drawing strokes, without having to find the command within a menu hierarchy. Gesture shortcuts are particularly valuable in pen or touch-based user interfaces, where keyboards are either inconvenient or not an option at all.

As with other types of shortcuts, gesture shortcuts face several major challenges such as learnability, memorability and scalability. Samsung Sunburst [12] enables programmable alphabet gesture shortcuts, allowing the user to associate an alphabet gesture with a specific task, such as "U" for unlocking the phone or "b" for starting the browser. These shortcuts can be convenient once a user learns them. However, a mobile user is often less motivated to spend the effort to manually associate shortcuts with targets or to learn built-in shortcuts. Even if they are willing to learn, the number of shortcuts that can be grasped and successfully recalled can be very limited, without much practice. In contrast, Gesture Search requires a user to neither create nor learn the mappings from gestures to data items and can easily scale up for accessing a large dataset.

To address these challenges, prior work employed three kinds of strategies to design the mappings from gesture strokes to commands: spatial (e.g., [7, 9, 14]), iconic (e.g., [10]), and mnemonic [8], as well as a combination of these [8]. For example, marking menus [9] nicely support gradual learning of the mapping of angular pen strokes to commands, by arranging commands in a hierarchical pie menu. For iconic gesture shortcuts (e.g., some of the gestures in [10]), the shape of a gesture is often semantically related to the associated action, e.g., drawing a rectangle to create a page or drawing left for panning left.

The prior work most closely related to Gesture Search is Command Strokes [8]. Kristensson and Zhai investigated using gesture strokes to input the name or prefix of a command to invoke the command. In addition to mnemonic mappings, Command Strokes also uses spatial mappings because it employs ShapeWriter, a gesture-enhanced on-screen keypad [14]. ShapeWriter maps the trajectories of strokes to individual words by arranging keys on the keypad optimally.

Similar to Command Strokes, Gesture Search leverages the mnemonic association from gestures to data items. However, there are several key distinctions. 1) Gesture Search does not use a keypad. This design is important because mobile phones have a small form factor and the screen real estate is limited. 2) Command Strokes requires users to learn to use ShapeWriter [14] stroke shortcuts. Before users grasp these shortcuts, they have to look for each key on the keypad. ShapeWriter's approach has advantages for entering a large amount of full-word text, e.g., writing an email. But for search, a user rarely needs to input the entire name of an item in Gesture Search, as shown in our study. 3) Gesture Search allows more flexible matching of a gesture query with data items. A query can match each term of an item's name, rather than having to prefix the entire name. Gesture Search also allows multiple search terms and tolerates misspellings. Lastly, Gesture Search shortens required query lengths by continuously learning from a user's search history.

CONCLUSIONS

We present Gesture Search, a tool for users to quickly access mobile phone data, such as applications and contacts, by drawing gestures. Gesture Search seamlessly integrates gesture-based interaction and search for fast mobile data access. It demonstrates a novel way for coupling gestures with standard GUI interaction. A longitudinal study with mobile phone users showed that Gesture Search enabled a quick and easy way for accessing mobile data. A user could access a variety of data items using short gesture queries, usually two gestures or less. The study also showed the majority of users reacted positively to the usefulness and usability of Gesture Search. Gesture Search has been released to public and is in use by hundreds of thousands of mobile users. The mean of over 5000 user ratings within the first three months of its public release was 4.5 , where 5 is the most positive. As a user enthusiastically commented: *"I gesture it, I find it."*

REFERENCES

1. Android. http://www.android.com/.
2. Quick Search Box. http://android-developers.blogspot.com/2009/09/introducing-quick-search-box-for.html.
3. Appert, C. and Zhai, S., Using strokes as command shortcuts: cognitive benefits and toolkit support, in *CHI'09*. p. 2289-2298.
4. Apple iPhone. http://www.apple.com/iphone/.
5. Mac Spotlight. http://support.apple.com/kb/HT2531.
6. Desktop Search. http://desktop.google.com/.
7. Guimbretière, F. and Winogra, T., FlowMenu: Combining Command, Text and Parameter Entry, in *UIST'00*. p. 213-216.
8. Kristensson, P.O. and Zhai, S., Command strokes with and without preview: using pen gestures on keyboard for command selection, in *CHI'07*. p. 1137-1146.
9. Kurtenbach, G. and Buxton, W. The limits of expert performance using hierarchical marking menus. in *CHI'93*. p. 482-487.
10. Newman, M.W., et al., DENIM: An Informal Web Site Design Tool Inspired by Observations of Practice. Human-Computer Interaction, 2003. **18**(3): p. 259-324.
11. Russell, S. and Norvig, P., *Probabilistic Reasoning over Time. 2 ed. Artificial Intelligence: A Modern Approach*. 2003: Prentice Hall. 537-583.
12. Sunburst™.http://www.samsung.com/us/consumer/mobile/mobile-phones/at-t-phones/SGH-A697ZKAATT/index.idx?pagetype=prd_detail.
13. Wobbrock, J.O., Wilson, A.D., and Li, Y. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. in *UIST'07*. p. 159-168.
14. Zhai, S. and Kristensson, P.-O. Shorthand writing on stylus keyboard. in *CHI'03*. p. 97-104.