Gesture Search: An Intuitive Tool for Fast Mobile Data Search

Anita Singh
MCA Research Scholar,
TIMSCDR
Mumbai, India

Ajeet Kumar Yadav
MCA Research Scholar,
TIMSCDR
Mumbai, India

ABSTRACT

It's very common to see that the latest generations of mobile phones are capable of gathering a wide range of data, right from GPS information to images to contacts. However, it is still a problem when it comes to accessing data in real time from mobile users. Therefore, we have taken to discover how Gesture Recognition can make wonders in gathering data from mobile phones.In this paper, we also demonstrate a fictitious scenario to emphasize this theory.This example has demonstrated that in the backdrop of a real-world example, people find it easier to access the mobile data. Given that mobile phones have also expanded in size, this helps the user to make gestures to their phones instead of actually touching the screens. Also, it seems that the current year is the beginning of the trend that will eventually popularize gesture control for smart phones.

Keywords

Gesture-based user mobile connectivity, search terms, shortcuts, Markov Models

1. INTRODUCTION

As the computing power and the storage capacities of new age phones expand, the base purpose of making calls no longer remains pivotal to the use. On the other hand, the potential to perform beyond measure is limited by user technology. Usually, widely used smart phones such as iPhones or Android phones have a large amount of data accessed and used by the user. However, it also becomes difficult to access the current data easily due to said technology limitations. The user usually needs to go through a lot of information in the mobile storage. On the other hand, the mobile screen and lack of a keyboard makes it difficult to access the data.

And in a similar manner to the desktop based keyword based queries, similar technique has been applied for mobile devices, e.g., theQuick Search Box used in Android phones [2]. Even though this techniques of searching for data items on smart phones makes it easier for the user, using a small virtual keyboard to type the query can still be an annoying.

On the other hand, search based query can be socially problematic as one cannot do so in public places. Also, the voice based query may not be able to understand the user's accent.

On the other hand, touch screen gestures can prove to be very convenient for frequently accessed data and commands (e.g., [12, 14]). Gestures on the other hand are a veryefficient way of finding data. The user makes certain hand gestures and associates them with a particular data item (e.g., [12]). However, there are several issues with these. First of all, the user will have to develop a certain gesture for a data item, which barely motivates the user. Secondly, with the growing number of shortcuts, it becomes difficult for the system to match gestures accurately, alongside making it challenging for the users to learn as well as recall. These issues were addressed by Kristensen and Zhao in their Command Strokes method. However, it still requires the phone's soft keypad and reduces the screen's "real estate". In addition, a user who is learningto use Command Strokes requires to scan and locate theright keys on the keypad. It takes a considerable time for the users to learnShape Writer shortcuts [14] that is the base of Command Shortcuts.

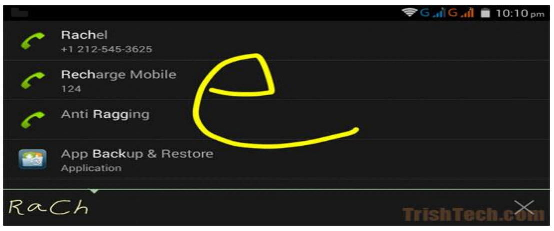


Figure.1 With Gesture Search a user can quickly access his contacts, phone apps and music just by making gestures.

Gesture Search was created for addressing these issues (Figure 1), that allows mobile phone users to access various types of data with gestures. It discovers a new approach in discovering newer ways to access mobile data.

A user is supposed to draw gestures on the screen for finding a data item. It matches the all possible gesture interpretations of the user and tries to find the data.

The search performance is optimized on the user's search history which allows even faster data access to the user on the phone.

1.1 Aim

The given user draws a sequence of gestures on the touch screen and Gesture Search makes a series of guesses on the patterns.

1.2 Objective

Distinguishing Gesturing from GUI touch actions.Gesture Search compares gesture inputs with the list in search results as well aslet'sinput gestures and GUI actions. Like scrolling and tapping for selecting. It requires users to explicitly signal the system for gesture drawing and list manipulation.In times of uncertainty that movements are making up gestures, the gesture system dispatches touch events to the widget for quick interface response (e.g., scrollingthe search result list as it normally does). Simultaneously, GestureSearch buffers existing touch events and looks forwardto more actions that may constitute a gesture. It also makes collected events as less obvious withyellow traces that indicate that the given strokes arestill being scrutinizedfor forming a potential gesture.Once it has been determined that the user is drawingAn actual gesture instead of just scrolling, the traces become bright yellow, which signifies use of a gesture (Figure 1).Simultaneously, Gesture Search stops applying touch actions towards the list of search results. The list would stop scrolling as a result, thus giving attributing the user a static background, thus finishingthe remaining gesture.Typical actions for the list widget include tap to select,scrolling as well flicking. For separating these events in a modeless manner and early when usersfinger slides the touch screen, the difference between the user's touch traces has to be discovered.

2. AN INTRODUCTORY EXAMPLE

Let's make an assumption that the user Bill, wants to make a phone call to Anne, his friend. First, he will draw letter 'A' on the screen. Once the gesture search is done, Gesture Search starts a process for search automatically. Since there is ambiguity in the gesture to Gesture Search — it maintains a kind of similarity with letters "H" and "A" — Gesture 1 Gestures are either letters or they are hand written. Therefore, there is an interchangeable use of gesturing and hand writing tracing simultaneously.

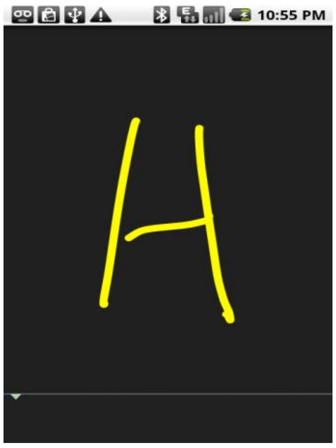
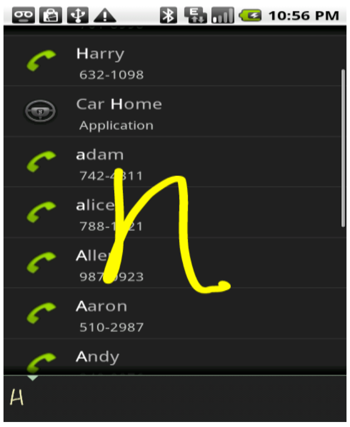
.


Figure 2. User gestures capital letter A on the screen with finger. Gesture Search employs timeout to delimit multi-stroke gestures


10:56 PM

Harry
632-1098

Car Home
Application

adam
742-4811

alice
788-1921

Allen
987-0923

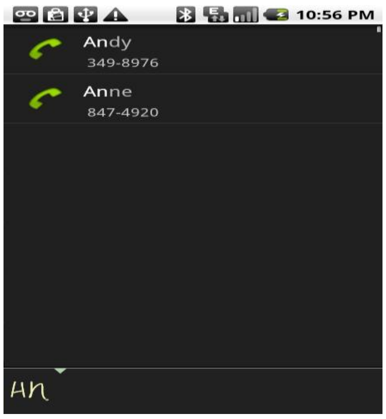
Aaron
510-2987

Andy

H

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.


10:56 PM

Andy
349-8976

Anne
847-4920

Hn

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

The search will display data that matches either of these (Figure 3). At the same time, a different version of the original data that has been scaled down is displayed at thescreen's bottom, meanwhile all the possible gesture recognitions are shown in search result contexts, e.g., The "Car Home" search result highlights "H" and "A" in the "Aaron" search. The highlights help explains the user why these data items get included in search results. Here, Bill can scroll down in the list or make additional gestures to the current ones. Gesture Search has the capability to distinguish automatically gestures by scrolling andtrajectory analyzing the user's touch patterns. The next section discusses this is detail. Now the user will draw a second letter "n" (see Figure 3). The search result will be analyzed by the application,simultaneously when the user is drawing a new gesture. When the two gestures are input in the system, the result shows Anne's contact in the second place in the list (Figure 4). Now bill only needs to click the contact item to view Anne's details. Or he can also tap on the green phone icon if he wants to call her directly. In this case, the user will need to redraw the gesture or the full query. All he needs to do is to make a horizontal wipe in the gesture query at the bottom of the screen. The right to left wipe will remove the last query gesture and the same action in the opposite direction will clear out the whole query. As the example shows, Gesture Search will use all the possible result interpretations of the gesture inputfor matching against every data item; the user does not need to draw the entire name of a target item before matching starts. In order to make the search query closer to a unique matching result, the user will have to draw a multiple prefix gesture query by delimiting the characters with spaces, e.g., waving "B S" in order to match an application "Barcode Scanner". He will draw a horizontal line on the search result's top.

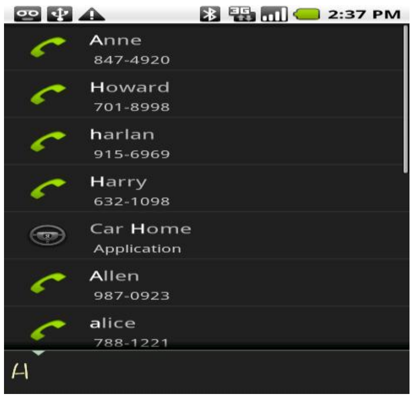


Figure 5. Data access is accelerated by Gesture Search by learning and assessing the search history of the user.

As a result, the user will be able to access the frequently visited data item with one search, gesture that is drawn casually or less perfectly.

3. SEARCH HISTORY LEARNING

By learning from the user search history, Gesture Search optimizes search performance. The optimization acceleratesdata access in several ways. Given that the application can tolerate the inaccuracies in gesture search patterns a user won't draw gestures with precision. Secondly, only a few gestures are requiredinput by the user as the systemas the system can predictitems that are derived from shorter queries.

The search history is also updated by the Gesture Search every time a user will click on search result list data item (Table 1).

Every row in the given search history will represent a unique mapping that is recognized query to that given target item. A row will also maintain mapping that was most recently occurred alongside the number of occurrences. Query field in the given row is a string that is recognized by the gesture query for the correspondingly selected item.

Last	Occurrence	Query	Selected Item	Frequency
8/6/09	5:00pm	BBilly		2
8/6/09	1:30pm	EuEugene		1
8/5/09	8:00pm	Ba	Barcode Scanner1	
8/5/09	7:00pm	LALosAngelesChronicle2		
8/4/09	10:00am	Bar	Barcode Scanner	1
8/3/09	2:00pm	TiTine		1
8/2/09	1:00pm	Br	Browser	1
8/2/09	9:00am	Sc	Barcode Scanner 1	

Table 1: Exemplified search history of the user.

IMPLEMENTING

This app was implemented using Java as well as SDK 2.0 and shows compatibility with Android 1.6. A series of tests have been carried out on various Android devices such as Motorola Moto X series and Google Nexus One.

4. MODUS OPERANDI

BASIC LOGIC& ALGORITHMS:

Here we discuss how does the basic logic and algorithms of this app work. For addressing the small form issue, Gesture Search expands the gesture input area by utilizing the whole screen for input and by superimposing an input layer on the search result list's top. This

creates an issue of how to effectively distinguish touch actions for a widget list like tapping 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.

5. CONCLUSION

We present Gesture Search, a tool for users to quickly access mobile phone data, such as applications and contacts, by drawing gestures. Gesture Search flawlessly integrates the user-mobile interaction for accessing mobile data faster. It shows a new way for gesture coupling with a standard GUI interaction. A study carried out on mobile users shows the Gesture Search was way more efficient in accessing mobile data. Short gesture queries let a user search mobile data items, usually two or lesser gestures. The study also demonstrated that the system got a positive response from an overwhelming majority of users. With a mean of user ratings above 5000, the first three months after its public release was given4.5, where 5 is the most positive.

6. REFERENCES

[1] Android. http://www.android.com/.

[2] QuickSearchBox. http://androiddevelopers.blogspot.com/2009/09/introducing-quicksearch-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. Combining Command, Text and Parameter Entry, in UIST'00. p. 213-216. 8. Kristensson, P.O. and Zhai, S., Using command strokes with as well as without preview: pen gesture usage on keyboard, command selection.