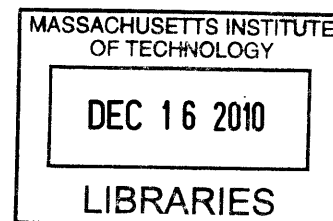


A Multimodal Speech Interface for Dynamic Creation and Retrieval of
Geographical Landmarks on a Mobile Device

by

Samuel S. Dyar

S.B. CS, M.I.T., 2009



Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

ARCHIVES

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2010

©2010 Massachusetts Institute of Technology. All rights reserved.

Author _____

Department of Electrical Engineering and Computer Science

July 27, 2010

Certified by _____

James R. Glass
Principal Research Scientist
Thesis Supervisor

Certified by _____

Scott Cyphers
Research Scientist
Thesis Supervisor

Accepted by _____

Dr. Christopher J. Terman
Chairman, Department Committee on Graduate Theses

A Multimodal Speech Interface for Dynamic Creation and Retrieval of
Geographical Landmarks on a Mobile Device

by

Samuel S. Dyar

Submitted to the
Department of Electrical Engineering and Computer Science

July 27, 2010

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

As mobile devices become more powerful, researchers look to develop innovative applications that use new and effective means of input. Furthermore, developers must exploit the device's many capabilities (GPS, camera, touch screen, etc) in order to make equally powerful applications. This thesis presents the development of a multimodal system that allows users to create and share informative geographical landmarks using Android-powered smart-phones. The content associated with each landmark is dynamically integrated into the system's vocabulary, which allows users to easily use speech to access landmarks by the information related to them. The initial results of releasing the application on the Android Market have been encouraging, but also suggest that improvements need to be made to the system.

Thesis Supervisor: James R. Glass
Title: Principal Research Scientist

Thesis Supervisor: Scott Cyphers
Title: Research Scientist

Acknowledgements

First of all, I would like to thank Jim Glass, my academic advisor and thesis co-advisor, for everything he has done for me. Jim has steered me through the stormy waters of MIT by planning out 8 wonderful semesters of classes for me. I would also like to thank him for offering me a MEng opportunity when no one else would. He has helped me, more than any other person, to graduate from MIT, and for that I will always be grateful.

I would also like to thank Scott Cyphers, my thesis co-advisor, for putting up with my many questions and for helping me with all the technical aspects of this project. Over the past year, Scott has taught me everything I know about databases and many other things about programming that I never learned in class.

I would like to include a personal thank you to my father, Stephen L. Dyar, for all the sacrifices he made for me, for the love he showed my family, and for providing me with the inspiration to always to do my best.

Most importantly, I would like to thank the Lord for the many blessings he has shown me and for making this all possible. “Trust in the Lord with all your heart, and lean not on your own understanding; in all your ways acknowledge Him, and He shall direct your paths.” (Proverbs 3:5)

This research was supported in part by DARPA under grant #HR0011-08-1-0079.

Table of Contents

1	Introduction.....	16
1.1	Motivation for Application	17
1.1.1	Real-World Problems Addressed by this Work	17
2	Background.....	19
2.1	Use of Mobile Devices	19
2.2	Use of Speech in Mobile Applications	19
2.2.1	Voice Command Systems	20
2.2.2	Speech-to-Text.....	21
2.2.3	Speech recording.....	21
2.3	Related Work	22
2.3.1	City Browser.....	22
2.3.2	Google Maps	24
2.4	Device Capabilities.....	26
2.4.1	Development Phone	26
2.4.2	Android API	27
2.5	Assumptions and Limitations.....	28
3	Architecture	30
3.1	Introduction.....	30
3.2	Content.....	30
3.2.1	Landmarks	30
3.2.2	Photographs	33
3.2.3	Comments.....	35
3.2.4	Albums	37
3.3	Sharing Content.....	39
3.3.1	Introduction	39
3.3.2	Server Database	39
3.3.3	Privacy Settings for Sharing Content.....	40
3.3.4	Downloading Shared Landmarks	41
3.3.5	Syncing the Phone with the Server	43
3.3.6	Pushing Data to the Server Database	43
3.3.7	Pulling Data into the Phone.....	44

3.3.8	Activating the Sync Process	45
3.4	Use of Speech.....	46
3.4.1	Introduction	46
3.4.2	System Commands.....	46
3.4.3	Speech to Text with the Google Speech Recognizer	47
3.4.4	Inconsistency between Hold-and-Speak and Click-to-Start.....	48
3.5	Selecting a Landmark	48
4	User Interface Design.....	50
4.1	Landmark Markers	50
4.2	Landmark Touch Interaction.....	52
4.3	Home Screen Buttons	54
4.4	Screen Interaction.....	56
4.5	Displaying Speech Results.....	57
4.6	Speech and Touch Options	58
5	Capabilities	60
5.1	Introduction.....	60
5.2	Terminology.....	60
5.3	System Commands	62
Zoom In or Out.....		62
Change Map Modes.....		62
Display Street View Overlay		63
Display Traffic Overlay		63
Turn My Location On or Off		64
Find My Location.....		64
Show My Location Options		65
Show All Visible Landmarks.....		66
Show Street View for a Point.....		67
Zoom In On Point.....		68
Add a Landmark.....		69
Add a Photograph to a Landmark		72
Add a Comment to a Landmark		73
Display Information for a Landmark.....		75
Display Photographs for a Landmark.....		76
Export Landmark Photographs		77

Display Comments for a Landmark	78
Display Edit Options and Commands for a Landmark	79
Edit a Landmark's Name or Description	80
Edit a Landmark's Position.....	81
Enter an Address	82
Edit a Landmark's Address.....	83
Move or Copy a Landmark to an Album.....	84
Add an Album	85
Manage Albums	86
Display Album Info	88
Display Album Options	89
Delete Content.....	90
Get Distance between Two Points	92
Get Directions to Places	93
Find Landmarks	98
Find Landmarks along a Street	99
Search Activity.....	100
Download Landmarks	106
Clear Search or Download Results.....	108
Sync Phone.....	109
Undo or Redo the Last Command.....	110
Help for Voice Commands	112
Help for Main Menu	113
Close the Application	114
6 Using Speech in MapDroid.....	115
6.1 Introduction.....	115
6.2 Building the Grammar	115
6.3 Handling the Results.....	117
6.3.1 Disambiguating landmarks.....	118
6.4 Displaying Results.....	119
6.5 Restriction of Grammar	119
6.5.1 Defense of Context-Free Grammars	120
6.6 Teaching MapDroid Syntax	121
7 Using the Mechanical Turk Service to Help Build a More Robust Grammar.....	123

7.1	Problems with Context-Free Grammars	123
7.2	Using Mechanical Turk to Gather Possible Commands.....	123
7.3	Setting up the Turk Tasks	124
7.3.1	Turk Task 1.0.....	124
7.3.2	Turk Task 2.0.....	124
7.3.3	Turk Task 3.0.....	125
7.4	Data Collected.....	126
7.5	Modifying MapDroid’s Existing Grammar	126
7.6	Collecting Test Data.....	126
7.7	Testing the Grammar Modifications	127
	Results	128
8	Testing MapDroid	130
8.1	Releasing MapDroid on the Android Market	130
8.2	Voice Command Results	131
9	Summary and Future Work.....	133
9.1	Summary.....	133
9.2	Future Work.....	133
9.2.1	Automatic Comment Transcription	133
9.2.2	Export Geo-Tagged Photos	134
9.2.3	More Flexible Sharing Policy.....	134
9.2.4	Improving Speech Recognition	135
9.2.5	Recognition for Locations	136
9.2.6	How to Correct Users.....	137
9.2.7	Offline Speech Capabilities.....	138
10	References.....	140
11	Appendix A.....	141
	Turk Task 2.0 Results.....	141
12	Appendix B	151
	Grammar	151

List of Figures

Figure 2-1: Screenshots of City Browser components running on the web. Main page (top), N-best list correction feature (bottom left) and the suggestions menu for what users can say (bottom right).....	24
Figure 2-2: T-Mobile’s G1 development phone with screen flipped.....	26
Figure 2-3: Popup menu for MapDroid's homescreen.	27
Figure 2-4: Example of Toast being displayed on MapDroid home screen.	28
Figure 3-1: The process of providing Text-to-Speech on MapDroid using the Google Recognizer activity. User clicks microphone button (left), waits for Google Recognizer Activity to start, speaks (center), and the best result is inserted into the text field. Clicking the text field displays an N-best list of the latest speech results (right).....	47
Figure 4-1: MapDroid home screen with landmarks present.	50
Figure 4-2: The home screen of MapDroid displaying the name of a landmark that has just been single tapped.	53
Figure 4-3: Home screen of MapDroid with all shortcut icons visible.	54
Figure 4-4: The microphone overlay that is displayed on the home screen while a user is talking.	57
Figure 4-5: Display of most likely speech result for “zoom in on my apartment” (left), and N-best results (right).....	58
Figure 5-1: My Location Options Activity used to display options for the My Location marker.....	65
Figure 5-2: Result of calling "show all the landmarks" command.	66
Figure 5-3: Street view display on home screen. Blue lines indicate streets that have street view capability. The “Street View Man” icon represents where the user would like to start the Street View Activity.....	67
Figure 5-4: Process for selecting a point to zoom in on. The user must invoke the recognition process by pressing down on the point of interest on the map (left). The map zooms into the point once the command is recognized (right).	68
Figure 5-5: Add Landmark Activity panel used to add a landmark’s name, description, and album.	69
Figure 5-6: The second Activity panel for adding a landmark, which is used to select the landmark's position.	70

Figure 5-7: Process for adding a comment to a landmark. Users can record a spoken comment and enter a transcription (left). They must then specify which landmark to add the comment to (right).....74

Figure 5-8: Info tab's panel that is used to display a landmark's information.75

Figure 5-9: Process for viewing a landmark's photographs. Thumbnails of photographs are displayed in a gallery (left). Selecting an image causes the photograph to be shown at full-view (right).....76

Figure 5-10: Process for viewing a comment. Comments are listed in a panel (left). Selecting a comment will open it, and allow the user to read the full transcription and listen to the audio (right).....78

Figure 5-11: Display of all the options a user can perform on a landmark.79

Figure 5-12: Activity panel used to edit a landmark's name or description.80

Figure 5-13: Graphical controls for directly moving a landmark.81

Figure 5-14: Activity panel used for editing a landmark's coordinates.81

Figure 5-15: Activity panel for entering an address.82

Figure 5-16: Activity panel used for editing a landmark's address. It is identical to the Enter an Address Activity panel (see Figure 5-15).....83

Figure 5-17: Activity panel used to create an album. It allows the user to input a name and description for the album.85

Figure 5-18: Activity panel used to create an album. It allows the user to specify the privacy level for the album.85

Figure 5-19: Tabbed activity for managing albums. Tabs show opened albums (left), closed albums (center), and information for each album (right).86

Figure 5-20: Tabbed activity that allows a user to view information about an album.88

Figure 5-21: Tabbed activity that allows users to view the available options for editing or adding content to an album.89

Figure 5-22: Example of screen display when calculating the distance between two points. A line connects the two points, and the distance is displayed on the screen as a Toast.92

Figure 5-23: Activity panel used to select two points to get directions between.93

Figure 5-24: Dialog used to help disambiguate which landmark to get directions for.94

Figure 5-25: Activity panel that allows user to specify a location.95

Figure 5-26: List of previous points used to get directions for.....96

Figure 5-27: Typical map display for directions on google Maps.....	96
Figure 5-28: Home screen display for landmark search. Landmarks matching the search results are identified by green "S" icons.	98
Figure 5-29: Process for searching along a street: A user must hold down on the street seen on the map, using the touch screen, and say the voice command (left). The map adjusts to display all the landmarks located on the specified street (right).....	99
Figure 5-30: Tabbed activity panel used to search for landmarks by name and/or description. .	100
Figure 5-31: Tabbed activity panel used to search for landmarks by proximity.	101
Figure 5-32: Tabbed activity panel used to search for landmarks by address.....	102
Figure 5-33: Tabbed activity panel used to search for landmarks by username.	102
Figure 5-34: Process for displaying search results. After the Search button is hit, all valid results are listed and the user can select one landmark or all of them (left). All selected landmarks are then displayed on home screen as green icons (right).....	104
Figure 5-35: Textual representation of search criteria display.	105
Figure 5-36: List of download results users can select from.	106
Figure 5-37: Tabbed activity used to download landmarks. The activity uses the same interface as the Search Activity.....	106
Figure 5-38: Help panel used to teach users what actions they can invoke using voice commands.	112
Figure 5-39: Help panel used to teach users what actions they can perform using the application's Main Menu.	113
Figure 5-40: Confirmations dialog displayed to users before they exit the application.	114
Figure 6-1: Dialog displayed to users that forces them to choose between landmarks of the same name.	118
Figure 6-2: N-best list of recognition results for a command.....	119
Figure 6-3: Help dialog for teaching users what they can say to the application.	121

List of Tables

Table 7-1: Error Rates for the Complete Test Dataset.....	128
Table 7-2: Error Rates for the In-Domain Dataset.....	128
Table 8-1: Results of running the recognizer on sample user commands, using both the normal MapDroid grammar, and an Oracle grammar comprised of only the test phrases.	131

1 Introduction

As mobile handheld devices become more powerful yet compact, it becomes more important to develop the use of speech as a means of input and instruction. While the functionality of mobile devices has increased, many applications depend on traditional modes of input. Most applications force the user to perform point-and-click operations on the device's touch screen or enter text by typing on a miniature keyboard. To improve performance, more progressive applications are starting to include speech-to-text and voice instructions that can be used to operate an application.

As multiple modes of input are explored, it is also important for developers to harness the multiple functionalities that modern mobile devices afford them. While most mobile handheld devices are now equipped with high resolution displays, GPS, camera, audio, and pen input, many applications only utilize a few of these capabilities.

This thesis presents an application that addresses these mobile device issues. MapDroid is a fully speech enabled application that allows users to bookmark important geographical points of interest. Additionally, users can take photographs or record comments and tag them to landmarks that they have created. Users can retrieve information and perform operations related to their saved landmarks by issuing voice commands. The landmarks can also be shared with other users, or be privately kept on the phone's database. In general, this research paper outlines how to build a powerful application that uses all the hardware functionalities of the device and the functionalities that are provided by the Android API that the system was built on.

1.1 Motivation for Application

The primary goal of this research is to create an application that can add and retrieve geographical landmarks using speech. The fact that speaking is both a natural means of communication and easy to do make it an attractive method of inputting information. Geographical landmarks were chosen because they are used to store information that is associated with a geographical position, which offers a natural organizational structure and contextual significance when displayed on a map. Landmarks can easily be referenced by name, and they allow the application to utilize the mobile device's many capabilities, since users can attach photographs and audio comments to them.

1.1.1 Real-World Problems Addressed by this Work

The application that is presented in this paper was designed so that it could be used in a variety of ways. Some possible uses of the application are described below:

Easy References to Locations

Instead of having to input the address of a destination each time you want to find directions to it, simply place a landmark at that destination and give it a unique spoken name. The labeled name is automatically added to the system's working vocabulary and can then be referred to at any later time (e.g. "Show me directions to *Rob's house.*")

Recording Spoken Information and Their Transcriptions to a Database

A cable technician must make rounds each day to different houses and apartments. At each stop, the technician surveys the problem and possibly uses his mobile device to retrieve information about the proper setup of the cables. After fixing the problem, or even being stumped by it, the technician can then speak to his/her mobile device and record information about the problem and its status (fixed, fixed but check back in, unfixed, etc.). The recorded speech and transcription can be stored in a company database. These comments can be retrieved later on by other technicians who may wish to see the past history for a particular customer.

The Modern Postcard

A daughter takes a trip to Italy. As she explores certain sites in different cities, she takes pictures and records spoken comments. The pictures and speech are anchored to her current global position on a map. She can then upload these landmarks and share them with family and friends. Each landmark is much more convenient to create and more informative than a postcard.

Chapter 2

2 Background

2.1 Use of Mobile Devices

The driving motivation behind mobile devices is that a user's actions should not be restricted just because they aren't at home or the office. Consequently, each new generation of mobile devices includes more functionality so that the user can conveniently travel without forfeiting the ability to perform certain tasks. Current mobile devices typically have high resolution touch-sensitive screens, GPS receivers, sizeable but limited RAM and storage, cameras, microphones, internet access, and many other capabilities. Because there are so many tools available on a single device, these devices are an excellent platform for applications that desire access to a rich variety of information.

2.2 Use of Speech in Mobile Applications

The use of speech in mobile applications is typically segregated into three different categories: voice command systems, speech-to-text transcription, and speech recording. The strength and novelty of the system proposed in this paper is that it uses all three speech technologies.

2.2.1 Voice Command Systems

Very few applications that have been developed for mobile devices use voice commands to operate them. The two major areas that we have seen voice command systems gain a foothold in are navigational applications (in cars) and voice dialing systems for mobile phones. Gruenstein et al. [2] have already demonstrated that City Browser can be used as a navigational aid in car systems. However, while cars are technically highly mobile devices, they do not fit into the category of handheld devices that we are exploring in this project.

The reason that few mobile devices currently rely on speech input is that conversational systems that run on these platforms face a variety of difficult problems. Some of the many issues that such systems must overcome are:

- Noisy background environments
- Varying user dialects and accents
- Limited vocabularies
- Low computational power means information must be sent to a server to do processing, which slows the system down.

Voice dialing systems have become popular because they don't face as many of these problems. They have a very limited vocabulary and a very strict structure. More complex systems, such as ours, must deal with these issues in a robust and reliable fashion.

2.2.2 Speech-to-Text

Many advances have been made in simple transcribing applications. These programs are mostly used for dictation, which only requires a straightforward conversion of speech-to-text. These capabilities are used in some mobile applications such as Promptu System's *ShoutOUT*¹, which allows users to dictate messages. *ShoutOUT* converts audio input into text which can then be saved or sent to friends. Recently, Android included Speech Recognizer intent in its SDK² that allows applications to perform speech-to-text transcription using the Google Recognizer. While speech-to-text can be very useful, it is notoriously unreliable since it faces most of the problems a voice command system faces (see Section 2.2.1).

2.2.3 Speech recording

Perhaps the easiest application for spoken input is just recording it and storing it as a verbal memo. One such application, e-Mobile *Voice Memo*³, allows users to take voice memos and replay them at a specified time (as a reminder to do something). These files are stored on the device for later viewing. They can also be sent to colleagues or sent to a remote desktop computer. The greatest challenge that voice memo applications face is figuring out a clever and intuitive way to store the voice memos so that they can be replayed in the future.

¹ <http://shout-out.mobi/>

² <http://developer.android.com/sdk/android-1.5.html>

³ http://www.palm.com/us/support/handbooks/tungstenc/tungstenc_voicememo_ENG.pdf

2.3 Related Work

When designing MapDroid, a few other closely related systems were examined in order to gain an understanding of their use of speech.

2.3.1 City Browser

City Browser is a multimodal interface that allows users to use speech and touch interaction as input to retrieve urban information. City Browser started out as a web-based platform that allowed users to retrieve information about restaurants and museums (Gruenstein et al. [3]). More recently, it has been adapted to be an automotive navigation aid (Gruenstein et al. [2]) which uses spoken language as input and outputs driving directions and other useful information for drivers. Most recently, City Browser was modified so that it could be run on a mobile device (Liu [4]). The current capabilities of the system are:

Speech driven search: Users push a click-to-talk button and speak a command into a microphone. The system converts this speech to text, parses out the information, and performs the task it thinks the user asked it to do. For example, a user could say “zoom-in” and the system would zoom-in on the map. A user could also say, “Show me all Italian restaurants in Cambridge, Massachusetts” (see Figure 2-1) and the system would display all the Italian restaurants located in Cambridge, MA.

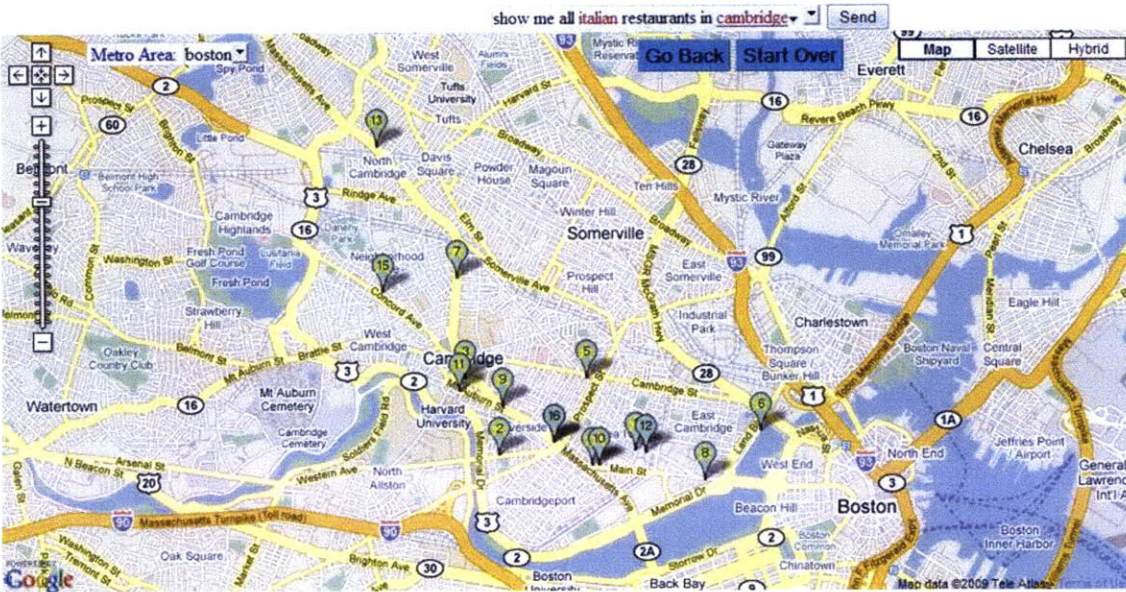
N-best list correction capability: Because speech-to-text isn’t perfect, the system offers an N-best hypothesis correction method. After each spoken command has been transcribed, words

that the system has low confidence about are highlighted in red. Beside each word is a drop down menu (see Figure 2-1) that contains the N-best hypotheses (usually restricted to about six words) the system generated for the word. If the system got one of the words wrong, the user can simply check to see if the correct word is in the dropdown menu. If the correct word is there, the user can select it and resend the command.

Suggested input template: Because novice users might not know what commands they can use, City Browser uses an unobtrusive suggestions template on the right side of the screen. The template provides a range of phrases such as “I’m looking for Chinese restaurants on Massachusetts Avenue in Cambridge,” or “Show me the cheap ones.” Both of these commands are valid and help users structure their requests. The template is dynamically updated so that the suggestions contextually relevant to the material currently being viewed are always displayed (see Figure 2-1).

MapDroid adopted many of these capabilities when it was designed. The most basic concept MapDroid used was the ability to say complex commands that included a number of parameters (locations, location types, etc...). Additionally, providing an N-best list of results became an important feature to help users correct misrecognition errors. MapDroid also considered the need for a template to help users learn what the system was capable of recognizing. This resulted in the *What can I say?* help dialog, described in *Teaching MapDroid Syntax*, pg. 121.

The primary difference between MapDroid and City Browser was that MapDroid was specifically designed to allow users to generate new content and to store this information. City Browser was designed to retrieve information but does not allow users to customize locations or add persistent content.



show me thirty two vassar street cambridge

show me thirty two vassar street cambridge

show me thirty two vassar street in cambridge

show me thirty two mansur street cambridge

and show me thirty two vassar street cambridge

show me thirty two mansur street in cambridge

What can I say?

Do you have the phone number for Game On!?

Tell me the address of Brown Sugar Cafe.

Do you have hours for Good And Healthy?

Show me the web page for Falafen Place.

Figure 2-1: Screenshots of City Browser components running on the web. Main page (top), N-best list correction feature (bottom left) and the suggestions menu for what users can say (bottom right).

2.3.2 Google Maps

There was a lot of excitement surrounding a recent release of Google Maps⁴ because it was “speech enabled”. Since speech is supposed to be the novel element of MapDroid, it is important to examine the differences between the uses of speech in each application. Google Maps uses a very basic approach. It only allows users to input an address using speech. The process is simple: the Google Recognizer produces a transcription for a spoken location and passes it to a geocoder, which finds the closest match for the location, and then Google Maps displays the

⁴ <http://googlemobile.blogspot.com/2009/06/search-by-voice-and-transit-directions.html>

point on the map. It is important to note that Google Maps cannot handle any commands. It simply uses speech as an alternative to typing in a destination to search for.

MapDroid uses a more complex set of commands that allows users to reference content that they have created, and perform a number of actions on these items. It cannot recognize free form speech, which means it can't find locations like Google Maps does. In this sense, the applications use speech in two very separate ways.

2.4 Device Capabilities

2.4.1 Development Phone

We chose to use T-Mobile's G1 development phone to build our application on. The G1 comes with many of the typical features we've come to expect from mobile devices. Some of the most relevant features are:

- High-resolution 3.2" touch screen display
- Memory card for storage
- Built-in GPS receiver
- High quality stereo speakers and sensitive microphone
- 3.2 mega pixel color camera
- High-speed 3.5G connection



Figure 2-2: T-Mobile's G1 development phone with screen flipped.

2.4.2 Android API

The application presented in this paper was designed to run on phones that use the Android operating system. The Android API offers a wide range of powerful components that have been incorporated into MapDroid. Some of the most important classes that were used are:

- **Activities:** Activities are to Android as windows are to a web browser. Activities primarily focus on a single thing a user can do. The main activity for MapDroid is responsible for displaying landmarks. However, there are many sub-activities a user can start, such as Get Directions, Search for a Landmark, Add a Landmark, and other tasks. Each Activity has its own window panel and design that directs user interaction. There are over 20 different Activities in MapDroid.
- **Menus:** Each Activity can generate a popup menu. The menu for each Activity appears at the bottom of the device's screen when the phone's MENU button is pressed. It is removed from the screen when the phone's MENU button or back button is pressed. A menu consists of 1-6 menu items. Each menu item can open a sub-menu list.
- **Google Map overlay:** One of the most important features for MapDroid is the ability to use a Google Maps overlay. The map is displayed on top of MapDroid's main Activity panel and allows landmark icons and other items to be overlaid on top of it.
- **My Location:** My Location is an overlay that uses the phone's GPS device. When the overlay is enabled it turns the phone's GPS on. As soon as the GPS

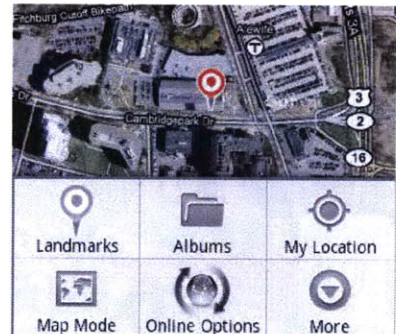


Figure 2-3: Popup menu for MapDroid's homescreen.

has a lock, the phone's location can be shown on a map as a blue dot. This allows the application to display the user's position, and to reference it in spoken commands.

- **Speech Recognition:** Google recently released a Speech Recognition activity. Applications can start this activity and the user can dictate speech. The speech is streamed to a Google Recognizer and an N-best list of the transcribed speech is returned to the Activity that started the Recognizer activity. Since a language model can't be passed to the Google recognizer, MapDroid only uses it for speech-to-text transcription purposes.

- **Toasts:** In order to display simple messages to the user, Android provides Toasts. Toasts are textual messages that are flashed on the screen for 2-3 seconds. They are handy because they disappear after a set period of time, unlike a Dialog message. If a position to place the Toast is not provided, then it will appear at the bottom of the device screen.



Figure 2-4: Example of Toast being displayed on MapDroid home screen.

- **Geocoder:** The geocoder allows an application to provide it an address or position, and returns the geographical location and address for the specified point. Geocoders are very helpful for filling in missing pieces of data about a location that aren't provided by the user.

2.5 Assumptions and Limitations

For the application to run, it is necessary for the user to have internet access in order to retrieve map views. Internet connectivity is also required in order to store and retrieve database information. Speech recognition also requires internet service, since all the recorded audio is

sent to an external recognizer. Requiring internet access is not an unreasonable assumption, since this application focuses on urban areas which should have the best available internet service.

The device's small screen is a restriction that must be designed for. While it is important to add shortcuts and menu options when possible, it is equally important to keep the screen free of these items so that users can see the full contents of the screen. One approach to working around this limitation, which is used in MapDroid, is to provide timed shortcuts icons that appear for a short period of time and then disappear.

There is also a restriction on the memory an application can use. Each application on the Android is allocated approximately 14 Mb to run. If the application exceeds this limit, it will crash. For most processes, this limitation is not a problem. However, this turned out to be significant when MapDroid tried to download content from its external database. When the application attempted to download photographs for a landmark, they were being sent as a single object, which filled up the application's memory and caused it to crash. This problem was fixable, but it is important for future developers to be aware of this limitation.

One of the features that Android does not provide is the use of Google Map directions to assist in real time navigation. Consequently, if MapDroid needs to get directions, it must outsource this task to the Google Maps activity. This can be done by starting the phone's Navigation activity and providing it with two points. This causes Google Maps to open, which displays navigational directions between the two provided locations.

3 Architecture

3.1 Introduction

As mentioned in the introduction to the problem (*Motivation for Application*, pg. 17), MapDroid's primary goal is to allow users to add photographs and comments to geo-specific locations, and to access this information through multimodal speech interactions with the application. Users also need to be able to share this information with other users. This chapter describes the architecture used in MapDroid that makes these features possible.

3.2 Content

3.2.1 Landmarks

Description: A landmark, in its most basic form, is a named location that is defined by its geographical coordinates. A user can add a description to a landmark, which offers a more detailed explanation of what real life feature the landmark represents. Landmarks are the most important items in MapDroid since they allow users to bookmark locations of interest. Users can view the landmarks they have created on the home screen of the application. To read about the process of creating landmarks, please see *Add a Landmark*, pg. 69.

Storage: Landmarks that are created by the user are stored in a single table in the phone's internal database. While name, description, and position are the most relevant pieces of

information for a user who is viewing the landmark, there are a lot of other fields that are relevant to the application. The following list describes all the fields that are stored in the database's Landmark table:

- *Name*: A landmark's name is used to easily identify landmarks when they are displayed in the application.
- *Description*: A landmark's description offers a more detailed explanation of what location the landmark is representing.
- *Latitude and Longitude*: The landmark's latitude and longitude are used to define the landmark's global position.
- *Longitude*: The landmark's longitude is used, along with its latitude, to define the landmark's global position.
- *Address*:
 - *Street*: The street address holds the value of the street address line for the landmark's location. This value is stored so users can find landmarks that are located at, or near, a specific address.
 - *City*: The city a landmark is located in is saved so that users can search for landmarks in a specific city.
 - *State*: The state a landmark is located in is saved so that users can search for landmarks in a specific state.
 - *Zip code*: The zip code for a landmark is saved so that users can search for landmarks by their zip codes.

- Album information:
 - *Parent album ID*: The ID of the album that the landmark is part of (see *Albums*, pg. 37, for an explanation about the use of albums), is stored so that the application can track which album a particular landmark is in.
 - *Parent album's privacy*: The parent album's privacy is also stored so that landmarks can automatically be synchronized without having to access the database to check the parent album's privacy setting.
- Ownership:
 - *Author username*: The username of the user who created the landmark is stored, so that users can reference this when searching for landmarks. It is also used to identify who created the landmark when users view the landmark information.
 - *Author ID*: The user's ID is stored so that the application can distinguish who "owns" the landmark. The creator of a landmark has extra privileges related to the landmarks content. The value used as the ID is currently the phone's unique ID.
- Synchronization:
 - *Sync status*: The sync status for a landmark allows the application to quickly check if a landmark needs to be synchronized. New or updated landmarks are marked as unsynchronized.
 - *Online ID*: The online ID for the landmark refers to the row ID for the landmark in the server database. This value is used to reference the online database's copy of the landmark when syncing changes.
- *Creation date*: The creation date is used to sort landmarks by creation time.

It should be noted that a landmark's photographs and comments are not stored with it.

MapDroid uses a model where items that belong to a parent item keep a reference to their parent item. The parent item, however, is never aware that items are attached to it. If a landmark is deleted, the application checks the photograph and comment tables and deletes all items that belonged to the deleted landmark.

3.2.2 Photographs

Description: Users can associate any number of photographs with landmarks. Photographs can add vital information to a landmark, since a landmark may be used to mark the location of a specific feature, but without a photograph of the feature, another user might not understand its significance. For example, if a police officer wanted to use MapDroid to mark the location of a vehicle that needed to be towed, a picture of the vehicle would be very helpful for whoever came to tow it. The process for adding a photograph is described in *Add a Photograph to a Landmark*, pg. 72.

Storage: Photographs that were taken by a specific phone are stored in a separate table in the phone's database. A photograph contains several important pieces of information, other than the photograph itself. The important pieces of information that are preserved in the photograph table are:

- *Photograph:* The actual image that has been taken is stored as a blob in the database.
- *Parent landmark ID:* The row ID of the parent landmark is saved so the application knows which landmark to attach this photograph to.

- Synchronization:
 - *Sync status*: The sync status of a photograph is stored so that the application can quickly grab all unsynchronized photographs.
 - *Online ID*: The online ID is used to update the copy of the photograph that's located in the external database (primarily to delete it if the original photograph is deleted).
- Ownership:
 - *Author username*: The username of the user who took the photograph is kept so that other users know who took the picture.
 - *Author ID*: The user ID for the user who took the photograph is kept for permission reasons (mainly so only the author can delete a photograph).
- *Creation date*: The date and time the photograph was taken is preserved so that photographs can be sorted by creation time.

It should be noted that all photographs are saved as 512 x 384 pixel images. This is the default “attachment” size for photographs on Android phones, and this size fits nicely with the MapDroid framework. Since photographs must be uploaded during the sync process (see *Sharing Content*, pg. 39), and potentially downloaded by other users, it is important to keep photograph sizes to a reasonably small level. The average size of a photograph of this type is less than 200 Kb. Additionally, the size of the photographs is still larger than the screen display for most Android phones, so there is no significant loss in resolution when the photo is viewed on the phone.

3.2.3 Comments

Description: Comments are short notes or observations that users may attach to specific landmarks. Comments are comprised of both an audio recording and a textual transcription of a spoken comment. Comments are structured like this because MapDroid was intended to offer textual transcriptions for spoken comments. MapDroid is lacking the automatic transcription capability, so users must manually type a transcription if they wish to have a textual representation of their comment. For a description of the comment recording process, please see *Add a Comment to a Landmark*, pg. 73.

Storage: Comments that were recorded on a specific phone are stored in a separate table in the phone's database. Like photographs, comments contain information other than the audio recording and textual representation of the comment. The important pieces of information that are preserved in the comment table are:

- *Comment recording:* The audio recording of the comment is stored in the table as a blob. It can be played back to users who wish to hear the original comment.
- *Comment transcription:* The transcription of the comment, if one has been entered, is used to offer a textual representation of the comment. This information is especially useful when listing all the comments that have been added to a landmark, since users can easily identify the comments they want to listen to.
- Parent:
 - *Parent landmark ID:* The row ID of the parent landmark is saved so the application knows which landmark to attach this comment to.

- Synchronization:
 - *Sync status*: The sync status of a comment is stored so that the application can quickly grab all unsynchronized comments.
 - *Online ID*: The online ID is used to update the copy of the comment that's located in the external database (primarily to delete it if the original comment is deleted).
- Owner:
 - *Author username*: The username of the user who recorded the comment is kept so that other users know who took the picture.
 - *Author ID*: The user ID for the user who took the comment is kept for permission reasons (mainly so only the author can delete a comment).
- *Creation date*: The date and time the comment was recorded is preserved so that comments can be sorted by creation time.

3.2.4 Albums

Description: Albums are used to organize landmarks. Like folders or directories on any operating system, they allow users to collect and group content. Albums are useful because they allow users to change settings for all the landmarks in an album by changing a single album setting. There are essentially three things that can be controlled by an album:

- *Privacy:* The primary attribute of an album is its privacy setting. The privacy setting controls whether the contents of the album are uploaded to the server database used by MapDroid. Changing an album's privacy setting is much easier for a user to do than trying to manage each individual landmark. Sharing content is described in *Sharing Content*, pg. 39.
- *Visibility:* An important feature for albums is that they can be “opened” or “closed”. This feature allows users to control whether the landmarks in each album are displayed on the map screen. This option is described in *Manage Albums*, pg. 86.
- *Deletion:* While this is not the intended purpose of albums, they can be used as a quick way to delete a group of landmarks. Deleting an album will delete all the landmarks in the album, and consequently, all the photographs and comments that belonged to those landmarks.

Storage: An album is only required to have two pieces of information: a unique name to identify it, and a privacy setting. Albums are stored in a separate table on the phone's database. Unlike the other three content items (landmarks, photographs, and comments), they are only used on the phone, and do not have a complementary table on the server database. Albums are very simple items to store, since they don't actually keep track of the landmarks that are grouped in

them; it is the responsibility of the landmark to keep track of the parent album's ID. For simplicity, albums cannot contain other albums, so there is no tree-like structure to manage.

Like all other content items, there are a number of fields that are stored in the phone's Album table:

- *Name*: The album name is used to identify each album on the phone and must be unique.
- *Description*: An album's description offers a detailed explanation of what types of landmarks are stored in the album.
- *Privacy*: The privacy setting for an album is represented as integer value, and is used to control whether its contents are publicly shared (uploaded to server database).
- *Author ID*: The user ID for the user who created an album is kept for permission reasons (mainly so only the author can delete an album).
- *Creation date*: The date and time an album was created is preserved so that albums can be sorted by creation time.

3.3 Sharing Content

3.3.1 Introduction

If MapDroid were to confine its content so that it could only be viewed on the device it was created on, it would severely restrict the usefulness of the information. For this reason, MapDroid was built to be a social application that allows users to share the landmarks they create and the photographs and comments they make.

3.3.2 Server Database

To achieve this functionality, MapDroid uses an external database. This database stores tables for:

- Landmarks
- Photographs
- Comments

These tables store information that is very similar to the information stored in their complementary tables on the phone (see *Content*, pg. 30). Consequently, landmarks, photographs and comments can be published to the external server. Each item on the server has a unique Sync Version number, which is used to associate downloaded landmarks with versions on the server. Once a landmark has been added to the server's database, other users can download the landmark.

3.3.3 Privacy Settings for Sharing Content

Like any social site, such as Facebook⁵, we expect users to generate both private and public content. For example, users may want to create landmarks that mark their home, or where friends live, and this content should probably not be publicly shared. Consequently, it becomes crucial that users have the ability to monitor the privacy settings for the landmarks they create.

Currently there are only two privacy settings; public and private. Every album that is created must choose one of these settings. The landmarks, and consequently the photographs and comments associated with these landmarks, use the privacy setting of the album they are in. The effect of each setting is listed below:

- *Private*: Landmarks located in private albums are not uploaded to the external database. Consequently, private landmarks can only be viewed by the author, since they only exist on the phone they were created on.
- *Public*: Landmarks that reside in public albums are uploaded to the server database when a user syncs his/her phone. From here, any other user can search for and download another user's landmarks.

Albums were chosen to control privacy, since it seems natural that landmarks of a similar privacy setting will be grouped together, and it would be a tedious process for the user to monitor the privacy for every landmark created.

These two settings do not offer enough flexibility. It is quite possible that a user will generate content that they would like to share with coworkers, friends or family, but not with the general public. Unfortunately, MapDroid does not currently offer this capability. This problem is discussed further in *More Flexible Sharing Policy*, pg. 134.

⁵ <http://www.facebook.com/>

3.3.4 Downloading Shared Landmarks

Users can download other users' landmarks by searching for them with the Download activity described in *Download Landmarks*, pg. 106. A user can specify which landmarks to download by:

- Name and/or Description
- Proximity
- Address (street, city, state, zip)
- Author username

When results are returned from a server query, a user is given the opportunity to complete the download by clicking the "Download All" button, or selecting a single landmark from the list of results.

All downloaded landmarks are stored in a separate table in the phone's database. The table contains similar fields to the regular Landmark table described in *Landmarks*, pg. 30, with the addition of a Sync Version number. This field is used to sync downloaded landmarks with the server, and serves a similar function as the Synchronized field in the regular landmark table.

Separating regular landmarks and downloaded landmarks into two tables makes some operations simple (such as clearing all the downloaded landmarks), but often makes other processes confusing because landmarks can't simply be referred to by their row ID, since the application also needs to know which table the ID refers to. Consequently, when the application accesses landmarks, it assigns them a Database ID, which is simply the row ID of the landmark with either an "R" (for Regular) or "O" (for Open or Downloaded) attached to the front of the row ID to indicate which table the landmark belongs to. For example, a downloaded landmark

that's located at row 17 in the Download table would be referred to as O17 when used in MapDroid. This allows the application to keep track of which table a landmark belongs to, when landmark IDs are passed between activities.

One important design decision was not to include the photographs and comments that belong to a landmark when it is downloaded. MapDroid purposely avoids this because these operations are extremely slow and photographs can take up a lot of space. If a user were to download 10 landmarks, there might be 20 photographs related to these landmarks. It could easily take 5 minutes to download all these photographs. Since the user might not even want to view the photographs of these landmarks, MapDroid simply ignores photographs and comments at this point. The trade off, of course, is that any time the user wishes to view a downloaded landmark's comments or photos, this content must be downloaded.

When users choose to view a downloaded landmark's photos or comments, they must open the content panel for the landmark and go to the Photos tab as described in *Display Photographs for a Landmark*, pg. 76. At this point, MapDroid finally requests that all the photographs for the selected landmark be downloaded. Once the photographs are downloaded, they are stored in a separate table of the database. This way, the photographs don't have to be downloaded again while the landmark's content panel is open. It should be noted that if a user exits the content panel and then opens it back up, the landmark's photos must be re-downloaded. This is acceptable because even though the user must wait to re-download the photographs, they are assured of having the most up-to-date content. This process works the same for comments as well.

3.3.5 Syncing the Phone with the Server

Since users can alter content on their phone, there will frequently be inconsistencies between the phone's internal database and the server database. To reconcile these differences, MapDroid contains a syncing process. The process can be broken down into two steps:

1. Push data from the phone's database to the server database
2. Pull data from the server database into the phone's database

3.3.6 Pushing Data to the Server Database

The initial step in the syncing process is to collect all the altered landmarks, photographs, and comments on the phone that are public, and upload the new content to the server database. The upload process is completed in four steps:

1. The application gathers all the landmarks that are currently not synchronized. This can easily be done by selecting all the rows from the Landmark table that have their Sync status value set to NOT_SYNCHRONIZED. Each unsynchronized landmark is then examined. For each landmark, if the landmark does not have an online ID (meaning it hasn't been synchronized with the database yet), it is inserted into the server database. The online row ID of the landmark created in the server is inserted into the online ID field for the landmark that is on the phone. If the landmark does have an online ID, then the application simply updates the row in the server database that corresponds to the online ID, with the phone's values. After being added or updated, each landmark has its Sync status changed to SYNCHRONIZED.
2. The application then gathers all the public comments that have not been synchronized. It applies a similar procedure to each comment as it does to the landmarks described in Step

1. After the second step, all the unsynchronized public comments have been uploaded or updated.
3. The application then gathers all the unsynchronized public photographs. It applies a similar procedure to each photograph as it does to the landmarks described in Step 1. After the third step, all the unsynchronized photographs have been uploaded or updated.
4. Finally, all the items that have been deleted on the phone also need to be deleted from the server database. Before a landmark, comment, or photograph is deleted, MapDroid checks to see if the item has an online ID. If it does, this ID is preserved, along with the item's corresponding server table, in a table on the phone's database. In Step 4 of the sync process, all the rows in the phone's Deleted table are collected. For each row, the application requests that the row in the provided table be deleted from the server. This ensures that there are no floating landmarks that have been deleted by the user but still exist on the server.

3.3.7 Pulling Data into the Phone

The landmarks a user has downloaded can become out of sync with the server database if a landmark's author uploads new content. To make sure that downloaded landmarks are synchronized, the application gathers all the downloaded landmarks from the phone's Download table. For each landmark, it uses the landmarks online ID to grab all the columns for that row from the server database.

If a landmark requested from the server database does not exist, the application assumes the landmark has been deleted and deletes the copy of the landmark that is in the phone's Download table. Otherwise, the application compares the Sync version number of the server

database and the phone's version. If the sync version is not the same, this indicates changes have been made to the server's copy, and the application replaces the values in the phone's database with the newly downloaded content.

It should be noted that even though the sync version is the only value that initially needs to be checked, MapDroid simply downloads all the content of a landmark because it is relatively small. This is done because it guarantees that only one request to the server has to be made for each landmark. Making one request and handling a slightly larger object is faster than making multiple requests that return small objects.

There are a couple other important points to make about the syncing process. First of all, since the phone does not keep copies of a downloaded landmark's photographs or comments, these items never have to be synchronized with the server database. Also, since only the author can edit the content for a landmark, there is no need to "merge" changes. Downloaded landmarks can simply be replaced by the newest version found on the server's database.

3.3.8 Activating the Sync Process

For a phone to be synchronized, a user must manually initiate this process (meaning the user has to push a sync button). While this places responsibility on the user to remember to sync their phone when they want to update their shared or downloaded landmarks, this is preferable to automatically syncing the phones since this process is often slow. If a user downloaded 100 landmarks, it would be a very time consuming process to check for updates for all these landmarks every time the user started the application.

3.4 Use of Speech

3.4.1 Introduction

MapDroid uses speech as a valuable source of input and control. The application uses speech in two distinct ways:


- Entering system commands from the home screen
- Speech to text

3.4.2 System Commands

From the home screen of MapDroid users can say a variety of commands that control MapDroid operations. To record speech, MapDroid provides users with a large talk button. When the user presses the button, the application automatically begins to record the audio input from the phone's primary microphone. When the button is released, the application stops recording. Due to streaming issues, MapDroid waits until a user has stopped recording before its sends the audio file to a server that is running the SLS WAMI[1] recognizer. The recognizer uses a JSGF grammar, provided by MapDroid (see *Using Speech in MapDroid*, pg. 115), to transcribe the input audio. The N-best results are then returned to the application. The operation associated with the best result is then performed. This is a brief description of how speech is processed when entering commands to the system. For a higher level description of the use of speech, see *Using Speech in MapDroid*, pg. 115.

3.4.3 Speech to Text with the Google Speech Recognizer

To be consistent with MapDroid's use of speech as a means of input, virtually every text field that can be edited for input is also speech enabled with the Google Recognizer. MapDroid uses the Google Recognizer primarily because the group that is responsible for creating MapDroid (CSAIL's Spoken Language System group) does not currently have a generic recognizer that can be used for transcriptions. It should also be noted that newer versions of the Android system have a speech button embedded in the soft keyboard that appears when the phone is in portrait mode.

 Except in the *Search Activity*, pg. 100, all textual input fields follow the same format. A small button with a blue microphone is located to the left of the text field. Clicking the button will start the Google speech recognition activity. A user can then dictate his/her speech. When the user is finished, the activity will automatically finish after it detects a long enough break in speech. An N-best list of the most likely sentences is then passed to the activity that started the speech command. The most likely result is displayed in the text field. The other results are stored as an array in the auto-complete list for the text field. Clicking the text field will bring up this N-best list. A user can then select a closer match to what they said if one exists. Otherwise, the user can manually enter the text using the keyboard.

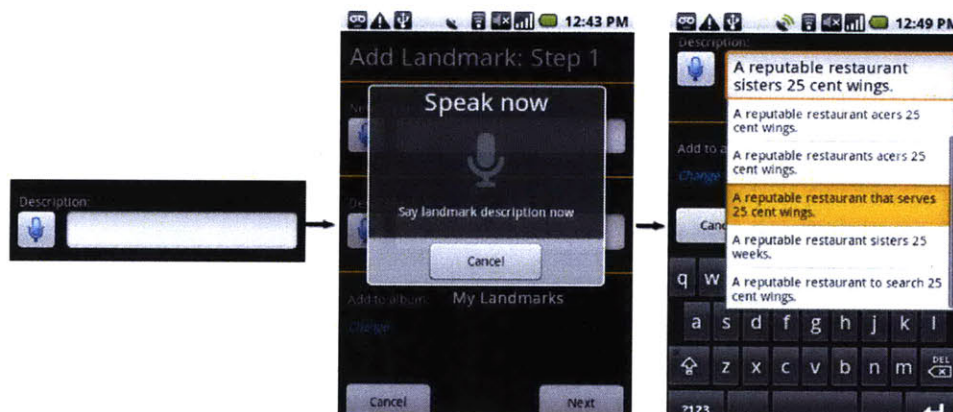


Figure 3-1: The process of providing Text-to-Speech on MapDroid using the Google Recognizer activity. User clicks microphone button (left), waits for Google Recognizer Activity to start, speaks (center), and the best result is inserted into the text field. Clicking the text field displays an N-best list of the latest speech results (right).

3.4.4 Inconsistency between Hold-and-Speak and Click-to-Start

During preliminary user tests, almost a unanimous complaint was that the Talk button on the home screen uses a walkie-talkie approach, where the user must be pressing the button in order to record speech, but the Google recognizer button uses a single tap to start the Recognizer activity. This is a significant inconsistency between two buttons that perform similar functions. The problem stems from the fact that users quickly become used to the walkie-talkie approach and they port this behavior to the Google Recognizer buttons. This results in users holding down on the Google Recognizer button, speaking, releasing the button, and the Google Recognizer activity finally being started. The user then has to repeat everything they just said. This is very frustrating, and a frequent mistake.

Unfortunately, there is no easy fix. Since the Google Recognizer activity can't be altered, the only option would be to modify the home screen's Talk Button so that it behaved consistently with the Google Recognizer activity. Since MapDroid does not have access to End-of-Speech technology, this means there is currently no fix.

To help combat this inconsistency, every Google Recognizer button has a long press listener added to it that Toasts a message to the screen that prompts the user to release the button and allow the Google Speech Recognition activity to start.

3.5 Selecting a Landmark

MapDroid uses the concept of "selection" to assist in user interaction with landmarks. The selection of a landmark is indicated by the landmark turning red. Only a single landmark can hold the status of being "selected." A user can select a landmark either by touching it or by

making a reference to it in a speech command. In the case where two landmarks are referenced in the same command (eg. “What is the distance between the *Prudential Center* and *Fenway Park*?”) then the last referenced landmark is used as the current selection (“Fenway Park”).

The Database ID of the currently selected landmark is stored in a separate table on the phone’s database. It is loaded on every restart of the application. There are a few instances when there are no landmarks selected (such as when a downloaded landmark is selected and then the user clears all the downloads). This is a perfectly valid state, since the application doesn’t require that a landmark be selected. If a user refers to “selected landmark” during a voice command and no landmark is selected, then the command simply won’t execute.

4 User Interface Design

4.1 Landmark Markers

MapDroid’s home screen is used to display landmarks on a Google Maps overlay. Landmarks are represented by balloon shaped icons that are similar to the markers used in Google Maps. There are six different landmark icons that represent all the different landmarks states.



Unselected regular landmark: The blue marker, with a solid circle in its center, represents a regular landmark that was created by the user and is currently not selected.



Unselected downloaded landmark: The blue marker, with an “O” in its center, represents a downloaded or “open” landmark that is currently not selected.



Unselected search landmark: The green marker, with an “S” in its center, represents a landmark that meets the most recent search criteria, but is not selected.



Selected regular landmark: The red marker, with a solid circle in its center, represents a regular landmark that was created by the user and is selected.



Selected downloaded landmark: The red marker, with an “O” in its center, represents a downloaded or “open” landmark that is currently selected.

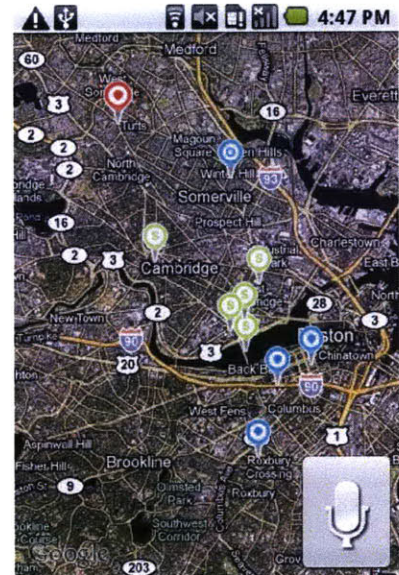


Figure 4-1: MapDroid home screen with landmarks present.



Selected search landmark: The red marker, with an “S” in its center, represents a landmark that meets the most recent search criteria, and is selected.

Red is used for all three types of landmarks (Regular, Downloaded, or Search) to represent a selected landmark. This was done in order maintain consistency; if a user sees a red landmark, no matter the symbol, then they know the landmark is selected. It should also be noted that the selected icons are slightly larger than their unselected counterparts. This makes selecting a landmark slightly more dramatic, since the landmark grows in importance, but not to the point where there is an obvious difference in size. Additionally, the selected landmark is drawn last, so that it stands out at the forefront of the screen and isn’t obscured by other landmarks.

Regular (created by phone’s owner) and Downloaded landmarks share a similar color scheme (blue), since they represent similar content. The only significant difference is that the user has created one, and has downloaded the other. In an early stage of development, a separate color was used to denote downloaded landmarks, but this resulted in four differently colored landmarks on the screen. A wide range of color becomes very confusing and unattractive. Consequently, both landmarks share the same color and only the center symbol indicates whether a landmark is Regular or Downloaded.

Search landmarks were given a separate color (green) to distinguish them from the other landmarks. This is done because users need to easily distinguish which landmarks meet their search criteria. Changing landmarks to green “S” icons results in a very noticeable difference in appearance. These landmarks will keep this appearance until a new search request is made, or the user clears the search results (see *Clear Search or Download Results*, pg. 108).



Another marker that is sometimes visible on the screen is the My Location icon. My Location is displayed as a blinking blue dot on the screen. A larger transparent blue circle is shown around the marker to indicate the accuracy of the phone's estimate of its position. The larger the circle, the more uncertainty there is about phone's current location. It should be noted that this marker only appears when My Location is turned on and the phone's GPS is usable.

One key design decision related to My Location was whether to automatically start the process when the application was started. Since many of the activities in MapDroid use My Location, it is important to have it enabled. The trade off is that searching for a GPS lock drains the battery life of the phone. When searching for a precedent on this matter, we observed that Google Maps always starts the My Location process when it is opened. Since this design didn't seem to offer users much control, MapDroid allows users to turn My Location on or off. When MapDroid is closed, it saves the users current preference (on or off). When MapDroid is started again, it can then refer to the last known state for My Location. We believe this is a nice solution, since it allows users to customize the use of My Location and automatically save their preference.

4.2 Landmark Touch Interaction

MapDroid makes use of the phone's touch screen in order to make the landmark markers interactive. Touch interactions with an object can be classified as three different types: single tap, double tap, and long press. MapDroid fully utilizes these interactions to make landmarks as interactive as possible:

Single tap: When a user taps a landmark, the name of the landmark appears directly underneath the selected landmark as a Toast message. If the map is moved, so that the Toast is no longer positioned next to the landmark, it is removed from the screen.

Double tap: If a user double taps a landmark, which is the equivalent of double clicking an item, the map automatically centers itself so that the selected landmark is located in the center of the screen. This is useful for when a user wants to zoom in on a landmark, since the marker remains at the center of the map and is never lost from view.

Long press: If a user long presses on a landmark, the content panel for the selected landmark is displayed. From here, the users can view all the landmark's information, photographs, comments and options available for the landmark. A long press, as specified in the MapDroid application, is considered to be a press down on the screen that lasts longer than a second.

It should also be noted that in order to be consistent with Google Maps use of the My Location marker, interacting with the My Location marker will produce a result similar to the Landmark touch interaction.

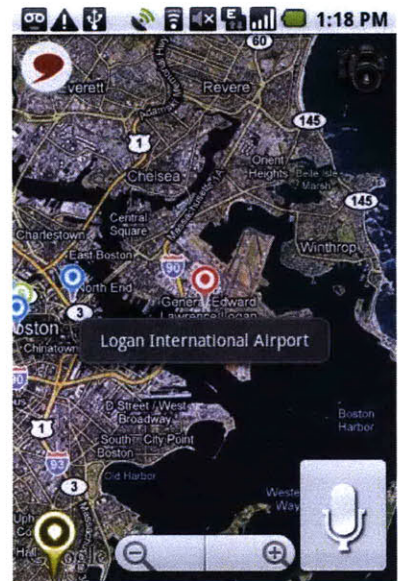


Figure 4-2: The home screen of MapDroid displaying the name of a landmark that has just been single tapped.

4.3 Home Screen Buttons

The MapDroid home screen contains more than just a map with landmarks. Additionally, there are several buttons that are located in the corners of the application home screen.



A large Talk button is located in the lower right-hand corner of the screen at all times. This button is used for speech commands. To start a verbal command, a user must hold down on this button and keep it held down until they are finished speaking. The audio that was recorded while the button was held down is then sent to a speech recognizer. The talk button is the only object that is a permanent fixture on the home screen. It is always present and prominent in order to promote users to use voice interaction.

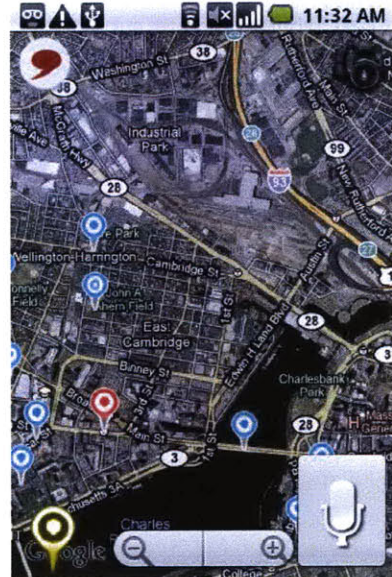


Figure 4-3: Home screen of MapDroid with all shortcut icons visible.

Because screen real estate is so important on mobile devices, Google has designed zoom controls, for map overlays, that fade out after approximately 3.5 seconds of inactivity. MapDroid adopts this design by providing the user with three shortcut icons that appear on the screen when a user touches the map. After about 3.5 seconds of inactivity, these icons disappear so that a user has a complete view of the map.



The large golden landmark icon that is located in the bottom left hand corner of the screen is a shortcut for starting the *Add a Landmark* Activity, pg. 69.



The speech bubble icon that is located in the upper left corner of the screen can be used as a shortcut for starting the *Record a Comment* activity (see *Add a Comment to a Landmark*, pg. 73).



The camera icon located in the upper right corner of the screen is a shortcut that starts the phone's Camera activity (see *Add a Photograph to a Landmark*, pg. 72).

These icons are provided as shortcuts for creating the three most important pieces of content for the MapDroid application. They have been included to not only reduce the time it takes to add a new item, but to promote the creation of new content. It is expected that new users will experiment with these icons to learn what they do.

The positions of the icons are relevant. The landmark icon, which is deemed to be the most important operation, is located at the bottom of the screen near highly trafficked operations (zoom controls and Talk button). Additionally, all the icons are located in the three unoccupied corners of the screen to reduce their impact on the maps visibility.

4.4 Screen Interaction

The main portion of the MapDroid home screen is filled by the map overlay. MapDroid provides a couple basic features to interact with the map. If a user would like to change the center of the map, they can simply touch the screen and drag their finger to pan around the map. MapDroid also has a pair of zoom controls that are located at the bottom of the screen. Pressing the - / + buttons will cause the map to zoom out or in (respectively) on an area.

Users can also start the speech activity that the Talk button is used to start by pressing down on the home screen's map overlay. If a user presses down on a landmark, this will open the landmark's content panel. Therefore, users must find a space unoccupied by landmark icons in order to start the speech activity. The process behaves similarly to the Talk button, in the sense that the user must press down on the screen in order to record speech and release when they are finished talking. This feature was included to support multimodal interaction. By pressing down on a specific point, users can reference the current point as "here" or "this place" in a voice command. For example if a user is holding down on the map they can say, "Zoom in on *this place*," and the application will zoom in on the point the user has selected.

As a preventive measure, the system will stop recording and throw out the recognition results if the user moves the map (by panning across the screen with their finger). This is enforced to prevent false recognition in cases where a user is long-pressing on the map, but is merely moving it around and doesn't mean to start the speech activity.

When the home screen speech recognition activity is started, an animated microphone overlay is shown on the center of the map that indicates that the application is listening to the user speak. This overlay serves a dual purpose, since it prompts the user to speak and also indicates the system is listening. The marker is immediately shown when the Talk button is

pressed. When the map is used to activate speech recognition, however, the microphone isn't shown on the map for a second, in order to make sure the user isn't simply performing other interactions with the screen.

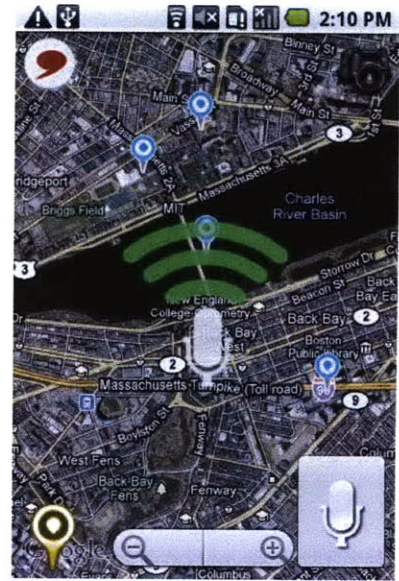


Figure 4-4: The microphone overlay that is displayed on the home screen while a user is talking.

4.5 Displaying Speech Results

Once a voice command has been successfully recognized, MapDroid executes it. As a visual confirmation that the recognizer did process their speech, MapDroid displays the results produced by the speech recognizer. MapDroid uses a dropdown list item, which is positioned at the top of the screen, to display the results. Initially, the dropdown list only shows the highest weighted speech result, and by extension, what action MapDroid just performed. The N-best results provided by the recognizer are also stored in the dropdown list. When the dropdown item is clicked, it shows all the N-best results produced by the recognizer. A user can then select a more appropriate command, if one is available. This process is described more in *Displaying Results*, pg. 119.

Because there is such limited screen real estate, the shortcut icons used for adding landmarks, photographs and comments are automatically hidden when speech results are displayed. Consequently, deciding when to hide the speech results became a very important decision. The logical conclusion was to remove the speech results from the screen as soon as any part of the map was touched. This conclusion is based on the assumption that once the user has touched the screen they have finished looking at the speech results and now wish to interact

with the map. Note that as long as the user touches the dropdown list, Talk button, or zoom controls, the speech results that were most recently returned will remain visible. As soon as the map is touched, the screen results are cleared, and the short cut icons regain visibility.

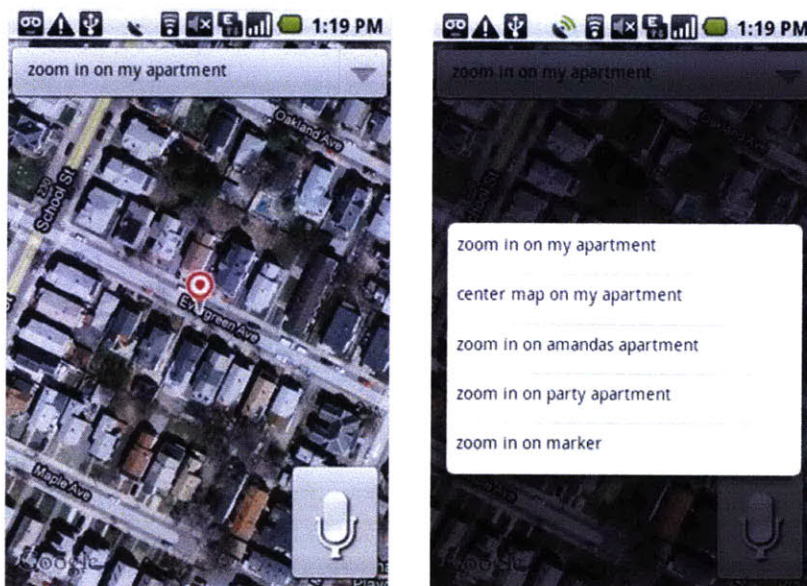


Figure 4-5: Display of most likely speech result for “zoom in on my apartment” (left), and N-best results (right).

4.6 Speech and Touch Options

When possible, MapDroid attempts to use speech and touch together to make an effective multimodal application. However, there are many operations that can’t fuse the two modes of input, such as turning on My Location. In such cases, MapDroid makes sure to allow users the ability to perform an operation using either touch or speech. Continuing with the previous example, users can either manually enable My Location by using the application’s main menu, or by saying, “Turn on My Location.” While this isn’t a true example of multimodality, it is better than a system that only allows a single mean of input.

There are some minor exceptions to this design specification. Some shortcut operations that are available with speech, such as “Zoom in on this landmark,” aren’t available in any menu options. The user can still use the application’s zoom controls to zoom in a landmark, but a manual shortcut option isn’t provided. Operations such as these were purposely ignored in order to keep option menus and lists small and easy to use. Similarly, some options that are available through touch interaction aren’t speech enabled. Some processes were deemed to be used so infrequently, such as “Export all photographs,” that they were not included in application’s grammar. This was done to reduce the possibility of misrecognition for more frequently used commands.

5 Capabilities

5.1 Introduction

When developing the functionality for the MapDroid application, there were three goals. The first was for MapDroid to possess almost all the capabilities currently available in Google Maps. Only incorporating a few of the basic features would be inconsistent with user expectations and would likely result in confusion and frustration. The second objective was for MapDroid to provide new and useful functionalities that used the application's content (landmarks, comments, photographs, etc...) that make MapDroid worth using. Lastly, and most importantly with respect to this project, each process should effectively be speech enabled in a natural manner that improves user performance.

5.2 Terminology

Before the current capabilities of MapDroid are listed, a few of the schemes for describing the available voice commands will be explained.

When describing how to perform an operation manually, the symbol ">" denotes a step in the process. For example, "Menu > Landmarks" indicates that the user should open the main menu on the phone (by clicking the MENU button), and then select the "Landmarks" menu item that appears in the resulting popup menu.

Two or more words surrounded by parentheses and separated by a line indicate that either word can be substituted. A good example of this is "Zoom (in | out)". This indicates that both

“Zoom in” and “Zoom out” can be used, but does not mean that both commands result in the same action.

When “here” is used, it refers to a selected point on the map that is neither a landmark nor the My Location marker. If the user is pressing the map in order to start the speech command, then the pressed point is considered to be “here”. In the case where the user is using the talk button to start the speech command, “here” is considered to be the center of the map.

When “this landmark” is used, it means the landmark that is currently selected will be used as the object of interest when the command it is part of is executed. For more information about how a landmark is selected, please see *Selecting a Landmark*, pg. 48.

In the case where “<...>” symbols are used to bracket a group name, it means a name from that group can be used directly in the voice command. These symbols are primarily used for “<landmark name>”, which means that the actual name of a landmark (eg. “Stata Center”) can be substituted. When searching for landmarks, however, you can use <tags>, <city>, and <state> names as well.

Lastly, when square brackets “[]” are used in a voice command, it indicates optional content.

5.3 System Commands

Zoom In or Out

Task description: This basic operation simply controls the zoom level of the map (where a higher zoom level results in a more detailed view for an area of the map).

Manual operation: Press the zoom buttons located at the bottom of the screen.

Voice command: “Zoom (in | out).”

Change Map Modes

Task description: Change the map type to the street map or satellite map / aerial view.

Manual operation: Menu > Map Mode > View (Street | Satellite) map.

Voice command: “Switch to (street| satellite) map.”

Display Street View Overlay

Task description: The street view overlay indicates which locations have Street View capability. Roads that do have Street Views are indicated by blue lines overlaid on top of them. Because both the street view and traffic overlays are drawn along streets, only one can be displayed at a time.

Manual operation: Menu > Map Mode > Show Street View.

Voice command: “Show street view.”

Display Traffic Overlay

Task description: The traffic overlay indicates the current flow of traffic along a street (red being congested and green being easy riding). Because the traffic overlay and street view overlay are drawn along streets, only one can be displayed at a time.

Manual operation: Menu > Map Mode > Show Traffic.

Voice command: “Show traffic.”

Turn My Location On or Off

Task description: This action enables or disables the phone's GPS unit. When the GPS is enabled and is able to get an accurate lock, the user's current location is displayed as a blue dot on the screen. Additionally a larger transparent blue circle is located around the location dot to display the accuracy of the position estimate. This is consistent with Google Maps use of My Location.

Manual operation: Menu > My Location > Turn (on | off) My Location.

Voice command: "Turn (on | off) My Location."

Find My Location

Task description: This command centers the map on the My Location marker. If My Location is not turned on, this command automatically enables My Location and centers the map on the phone's current position as soon as the GPS gets a lock.

Manual operation: If My Location marker is visible, double tap on the marker.

Voice command:

- "Where am I?"
- "Find My Location."

Show My Location Options

Task description: To be consistent with the landmark model and Google's use of My Location, MapDroid contains a page of operations that can be performed using My Location. The page also provides the street address of the phone's current position.

Manual operation: Long press on the My Location marker.

Voice command: "Show options for My Location."

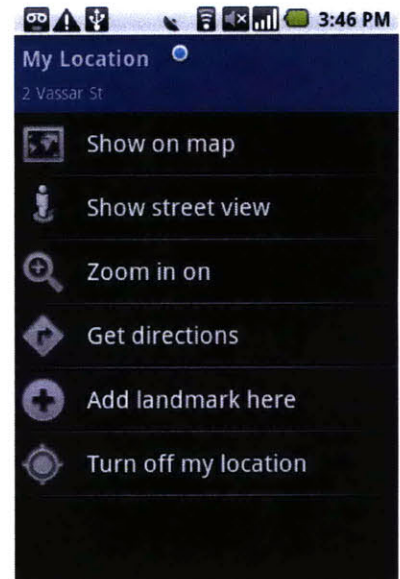


Figure 5-1: My Location Options Activity used to display options for the My Location marker.

Show All Visible Landmarks

Task description: To help users keep track of all their visible landmarks, MapDroid introduces a simple function that fits the map display to the closest zoom level that still shows all the landmarks on the screen.

Manual operation: Menu > Landmarks > Show All My Landmarks

Voice command: “Show all my landmarks.”


Note: There is a slight inconsistency in the name of the method, since the function fits the map to all visible landmarks – which includes downloaded landmarks that were not created by the user. Additionally, if a user has closed an album, the landmarks in this album will remain hidden and the function will ignore them.



Figure 5-2: Result of calling "show all the landmarks" command.

Show Street View for a Point

Task description: Google Maps includes the Street View activity, which offers a 360 degree first person view of a location. These views are only available along streets (which are indicated by blue lines when the street view overlay is on). MapDroid allows users to easily start this activity for a specified location.

Process: While Street View is on, a “Street View Man” icon is  present on the screen. Tapping anywhere on the screen will relocate the icon to that point. To actually start the Street View activity for a given point, the user must long press on the icon located at that point. It should be noted that if a user taps close to a landmark icon, the street view icon will snap to that location.

Manual operation: Menu > Map Mode > Show Street View > Long press icon

Voice command: “Show street view for (here | My Location | this landmark | <landmark name>).”

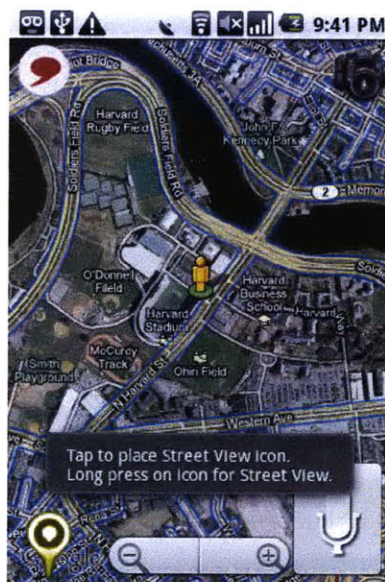


Figure 5-3: Street view display on home screen. Blue lines indicate streets that have street view capability. The “Street View Man” icon represents where the user would like to start the Street View Activity.

Zoom In On Point

Task description: To quickly get a detailed view for a location of interest, MapDroid provides the “zoom in on” feature. The method automatically focuses the map on the location of interest and zooms to the maximum level (~19). This is much faster than trying to keep the map centered on a point while zooming in 4-5 times in order to get a closer view of an area.

Manual operation: Long press on the My Location marker > Go to the options page > Zoom in on.

Note: This function can only be manually performed on My Location. There is no manual shortcut for zooming in on landmarks or points on the map – users must use the provided zoom controls.

Voice command: “Zoom in on (here | My Location | this landmark | <landmark name>)”

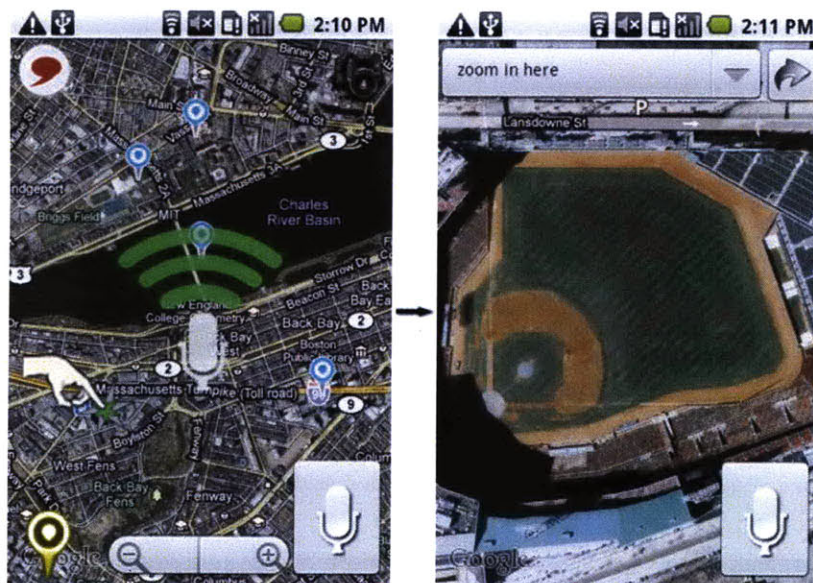


Figure 5-4: Process for selecting a point to zoom in on. The user must invoke the recognition process by pressing down on the point of interest on the map (left). The map zooms into the point once the command is recognized (right).

Add a Landmark

Task description: Users may add new places to their list of landmarks. A landmark has two critical pieces of information that the user must specify: name and location.

Process: The process of adding a landmark is broken up into two steps.

Step 1:

- *Name:* Naming a landmark is essential, since MapDroid expects to be able to reference each landmark by name. A landmark's name is also what is displayed when a user taps on a landmark, so it is important to have relevant information to help distinguish between landmarks. If a user does not enter a name, and attempts to proceed to the second step, a message is flashed to the screen that prompts the user to enter a name for the landmark.

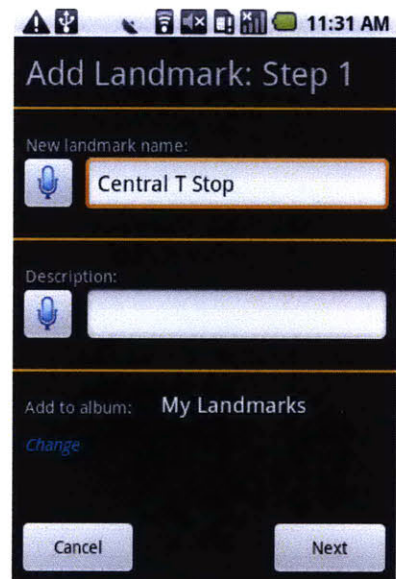


Figure 5-5: Add Landmark Activity panel used to add a landmark's name, description, and album.

Users also have the option of inputting two pieces of additional information:

- *Description:* A user can input more detailed information about a landmark in the description field (which is on the same panel as the name). The description content can be used to find and filter landmarks by type.
- *Album:* The user can also choose which album to store the landmark in. The name of the current album (the default being "My Landmarks"), is displayed below the description field. A user can choose a different album by selecting the "Change" label.

Step 2:

- **Location:** The location is necessary for knowing where to display landmarks. Users can specify the location for the landmark using three different places.
 - **My Location:** The landmark is added to the current position of the phone. This option is only enabled if the user has My Location turned on and the GPS has a lock.
 - **Click on Map:** Upon selection, the “Add landmark” panel is removed and the application returns to the home screen. At this point, the user can touch the map and a landmark is automatically added at this point.
 - **Address:** Users are taken to a page which allows them to add an address’s street, city, state, and zip code. This information is then passed into a Google geocoder to generate a specific GeoPoint (latitude, longitude) for the landmark. This option is only enabled if the phone has internet connectivity, since the geocoder requires this.

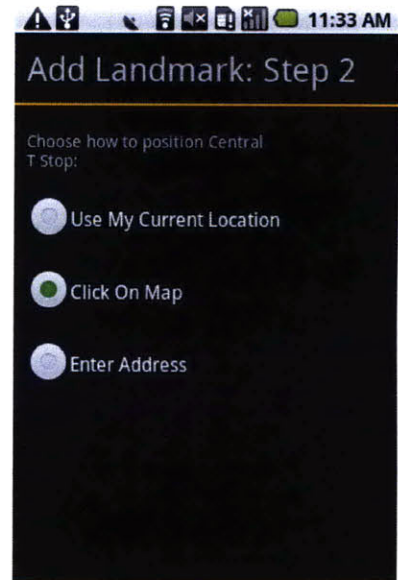


Figure 5-6: The second Activity panel for adding a landmark, which is used to select the landmark's position.

One of the advantages of using a voice command is that a user can specify where to add the new landmark (either My Location or a point on the map). In this instance, Step 2 is bypassed, and the landmark is directly added to the specified point.

Manual operation:

- Press the gold landmark icon in the bottom left corner of the home screen
- Menu > Landmarks > Add landmark



Voice command: “Add a landmark (here | at My Location).”

Add a Photograph to a Landmark

Task description: Users can add photographs to their landmarks, and to downloaded landmarks.

Process: When a user chooses to take a photograph, the application opens the Camera activity.

In the case that the user has simply taken a picture without specifying which landmark to add it to, the application switches to an “Add Media” activity, where the user explicitly chooses which landmark to add the photograph to. If a landmark is currently selected, the page starts with this landmark as the default.

In the case where a landmark has already been chosen, the “Add Media” step is bypassed, and the photograph is immediately added to the specified landmark.

Manual operation:

- Pressing the camera icon present on the MapDroid home screen (indirect)
- Long press landmark to open content > Edit tab > Add photograph (direct)

Voice command:

- “Add a photograph.” (indirect)
- “Add a photograph to (this landmark | <landmark name>).” (direct)

Add a Comment to a Landmark

Task description: Comments are an essential part of MapDroid since they allow users to attach thoughts and reminders to places, and then share this content with other users. While most systems that deal with comments only use text, we allow users to enter both textual and verbal comments. Both the textual transcription and waveform for a comment are saved, since it was our initial goal and expectation to allow users to say comments and automatically generate textual transcriptions. Because we currently lack this capability (to both save the waveform and generate a transcription), users must manually type out a transcription if they wish to have one.

Process: To record a comment, the user can tap a record button, which automatically disables the other media playback buttons. A “Recording...” prompt message is also displayed to indicate the system is recording. To end the recording, the user can simply tap the record button a second time. The length of the audio comment (in seconds) is immediately displayed on the screen as an indication that the user has successfully recorded something. The user can listen to the comment by pressing the play button. When the user is satisfied, he/she can click the “Attach” button.

If a landmark hasn't been specified, the user is shown the “Add Media” page as described in “Add a Photograph to a Landmark”. If a landmark has already been specified, the comment is directly added to the landmark.

Manual operation:

- Pressing the speech bubble icon present on the MapDroid home screen. (indirect)
- Long press landmark to open content > Edit tab > Add Comment (direct)

Voice command:

- “Add a comment.” (indirect)
- “Add a comment to (this landmark | <landmark name>).” (direct)



Figure 5-7: Process for adding a comment to a landmark. Users can record a spoken comment and enter a transcription (left). They must then specify which landmark to add the comment to (right).

Display Information for a Landmark

Task description: All the textual information related to a landmark is displayed on one page of a 4-tabbed content panel.

This information includes the landmark's:

- Name
- Description
- Address
- Position (GPS coordinates)
- Album name
- Username of author
- Creation data and time

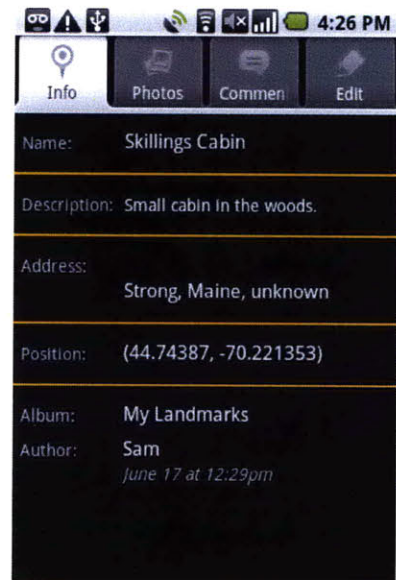


Figure 5-8: Info tab's panel that is used to display a landmark's information.

When a user opens a landmark by long pressing on it, this is the default tab that is displayed.

Manual operation: Long press on a landmark

Voice command: "Show information for (this landmark | <landmark name>)."

Display Photographs for a Landmark

Task description: All the photographs for a landmark are displayed in a thumbnail gallery on one page of a 4-tabbed content panel. The photographs are displayed from most-recent to oldest creation date. To view a specific image, a user can click on it. This opens a View Image activity that displays the image at the largest size possible that still fits on the screen. Users can scroll through the images in this view by selecting buttons pointing to the left and right. The buttons fade away after ~3.5 seconds, but touching the screen will make them reappear. To return to image gallery on the Photos tab, users can click the phone's back button.

Manual operation: Long press on landmark > Photos tab

Voice command: "Show photographs for (this landmark| <landmark name>)."



Figure 5-9: Process for viewing a landmark's photographs. Thumbnails of photographs are displayed in a gallery (left). Selecting an image causes the photograph to be shown at full-view (right).

Export Landmark Photographs

Task description: Exporting photos allows users to access photographs they have taken using MapDroid. Because photos for landmarks are stored on the phone's database, these photos aren't initially available for use outside of the MapDroid application. If a user wants to gain access to these files, they must choose the "Export photos" option in the More menu. Users can also export photos for a single landmark by choosing the "Export photos" option in the Edit tab of the landmark (see Export Landmark Photographs, 77).

Process: When a user chooses to export photos, a folder is created in the Camera folder on the phone. Then, for each landmark, a separate folder is made using the landmark's name. In case of conflicts, a number is attached to the landmark name to make the filename unique. The photographs for each landmark are then written as .PNG files into the corresponding folder. Since this can be a long process (10 seconds or more), a dialog message is displayed that alerts the user that files are being written, and where they are being written to. It should be noted that only the user's photographs are exported, since exporting photographs for downloaded landmarks would require the application to download every photograph, which would be very time consuming.

Manual operation:

- Menu > More > Export photos (all photographs)
- Long press landmark > Edit tab > Export photos (export photographs for individual landmark)

Voice command: NONE. This process is not used enough to warrant a voice command.

Display Comments for a Landmark

Task description: All the comments that have been recorded for a landmark are listed on one page that is part of a 4-tabbed content panel. The comments are listed in order of most-recent to oldest creation date. The transcription, author, creation date and time of the comment are displayed as a single list item. Only the first two lines of the comment are actually shown, so that all listed comments are of equal size.

To listen to a specific comment, or to read the full transcription, a user can click on it. This opens up the same panel that is used in the *Add a Comment* Activity (see *Add a Comment to a Landmark*, pg. 73). The main difference is that the “Record” button is removed from the panel, so users can only play the audio and read the transcription. The panels for recording and displaying comments are similar in order to reduce the number of interfaces users have to familiarize themselves with. To return to the “Comments” tab, a user can click the OK or Cancel button.

Manual operation: Long press landmark > Comments tab

Voice command: “Show comments for (this landmark | <landmark name>).”

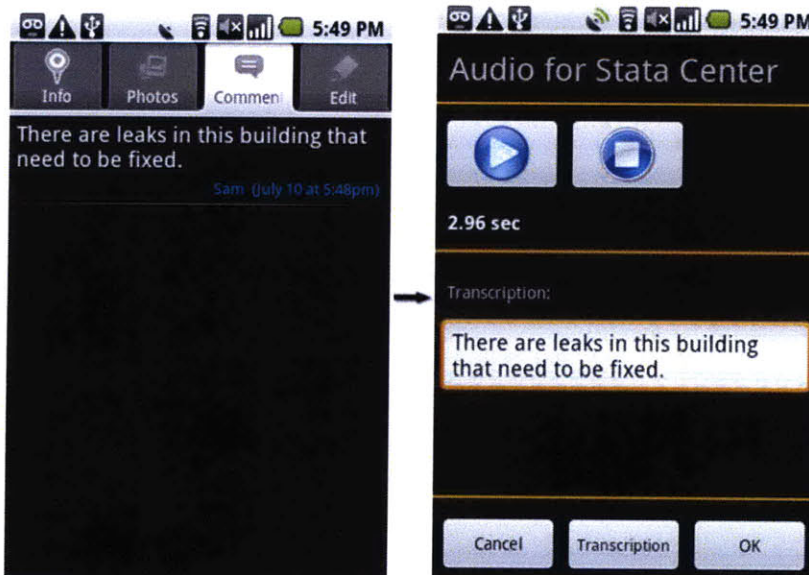


Figure 5-10: Process for viewing a comment. Comments are listed in a panel (left). Selecting a comment will open it, and allow the user to read the full transcription and listen to the audio (right).

Display Edit Options and Commands for a Landmark

Task description: To edit a landmark, or to view the different commands that can be performed on a landmark, a user can open a landmark's Edit tab. The "Edit" tab's name is a misleading title since other operations are also displayed (such as "Sync this landmark"). This grouping was chosen primarily because a tabbed activity can only have a maximum of 4 tabs, and the other three tabs were already being used. To choose an action, the user can click on the corresponding list item.

Manual operation: Long press on landmark > Edit tab

Voice command: "Show options for (this landmark | <landmark name>)."

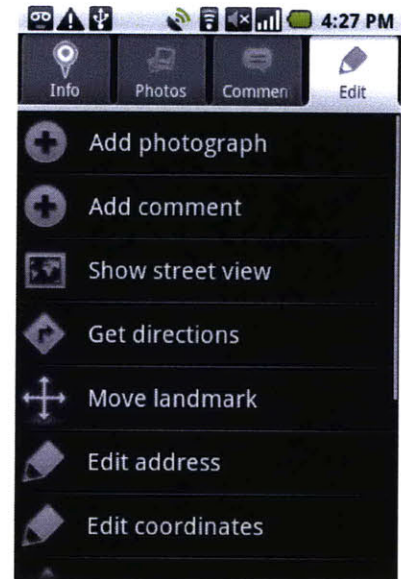


Figure 5-11: Display of all the options a user can perform on a landmark.

Edit a Landmark's Name or Description

Task description: In order to update the name or description for a landmark, a user must open the edit panel for this content.

Name and description are included on the same panel, so a user may edit either field even if they start the activity by requesting to edit only one of the fields. Name and description are grouped together to be consistent with Step 1 of the *Add a Landmark* Activity, pg. 69.

Manual operation: Long press on landmark > Edit tab > Edit name or description.

Voice command: "Edit (this landmarks | <landmark name>) (name | description)."

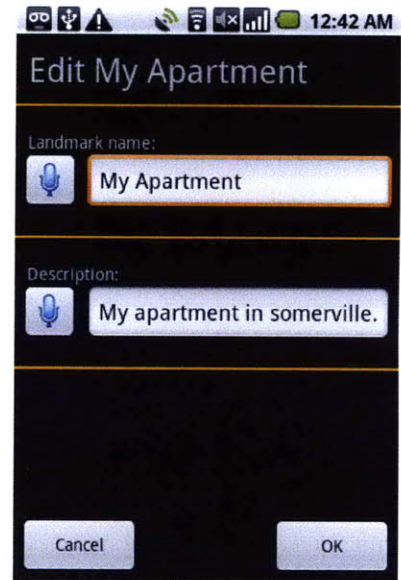


Figure 5-12: Activity panel used to edit a landmark's name or description.

Edit a Landmark's Position

Task description: It is very possible that a user will add a landmark to the wrong location, or wish to adjust a landmark's position for some alternative reason. In order to do this, they must be able to alter a landmark's coordinates. There are two methods for altering a landmark's position. The easiest way to reposition a landmark is to directly manipulate it by dragging it around on the map.

A user may also edit the GPS coordinates of the landmark if they know the precise coordinates they wish to move the landmark to. It should be noted that moving a landmark will also cause the landmark's address to automatically be updated to the best address determined by the Google geocoder, based on the landmark's new location.

Manual operation:

- Long press on landmark > Edit tab > Move landmark
- Long press on landmark > Edit tab > Edit coordinates

Voice command:

- "Edit (this landmark's | <landmark name>) position." (indirect)
- "Move (this landmark | <landmark name>) here." (direct)

Note: There is no voice command for editing a landmark's coordinates.

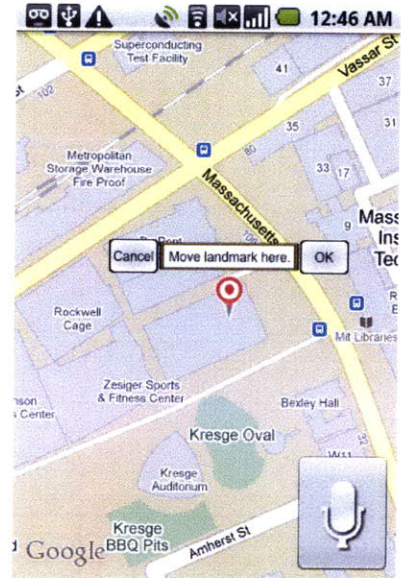


Figure 5-13: Graphical controls for directly moving a landmark.

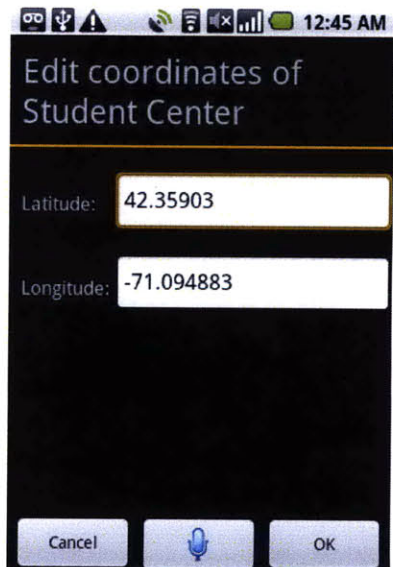


Figure 5-14: Activity panel used for editing a landmark's coordinates.

Enter an Address

Generating addresses for locations is a very simple and powerful process, due to the combination of the Google Recognizer and geocoder. There are four natural fields for a United States postal address: Street, city, state and zip code. Consequently, the “Enter an Address” panel has four text boxes for each of these fields.

Assigned to each text box is a button that starts a Google Recognizer activity. The results from these Recognizer activities are directly pasted into the text fields that are assigned

to the buttons that started the activity. Alternatively, users may type in each of the text fields.

The most efficient means of entering an address is to use the speech button at the bottom of the screen. This starts the Google Recognizer activity, which produces a transcription of the spoken address. This transcription is then fed into a geocoder, which returns the physical address for the input location. Each of the four elements (street, city, state and zip) are stripped from the address, and inserted into their corresponding text fields. Consequently, a user often only has to say the street and city, and the activity will fill in the state and zip code fields for the user.

Figure 5-15 illustrates the effectiveness of this process. The speech input to the Google Recognizer was “32 Vassar Street”. The recognizer was able to correctly recognize this name, and the geocoder was able to come up with an entire address for the location, which can be inserted into the address fields that weren’t specified.

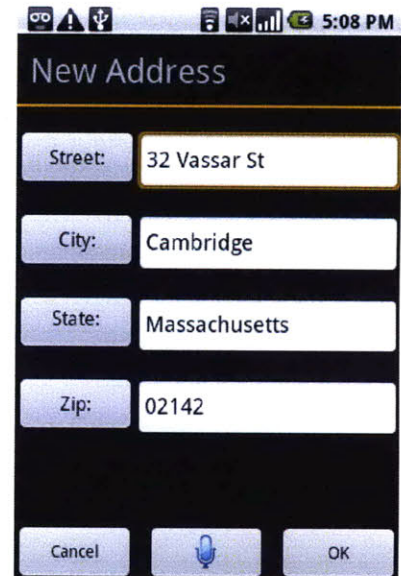


Figure 5-15: Activity panel for entering an address.

Edit a Landmark's Address

Task description: In many situations, the address automatically generated by adding a landmark using My Location or by using a point on the map, is inaccurate. The Google geocoder will often assign the address for a specific point to a range of addresses. In the case where a user adds a landmark and their phone does not have internet connectivity, then there will not even be an address associated with the landmark.

When a user chooses to edit an address, the landmarks address is shown in panel similar to the *Enter an Address*

Activity, pg. 82. The address is split up by street, city, state and zip code. Users can either type in the name or number for each field, or use the Google Recognizer buttons assigned to each field. It should be noted that changing the address for a landmark does not change position of the landmark. This restriction is enforced in order to prevent a deadlock between incorrectly altering position and address values.

Manual operation: Long press landmark > Edit tab > Edit address

Voice command: "Edit (this landmarks | <landmark name>) address."

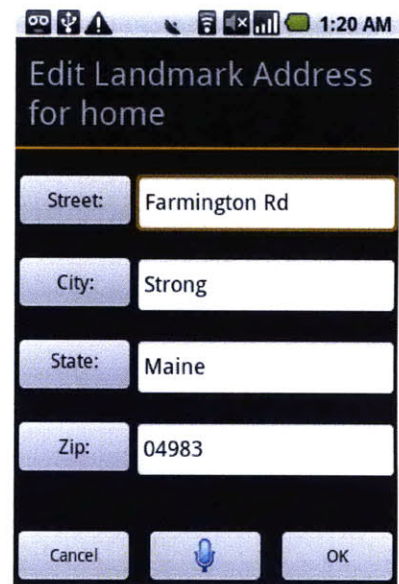


Figure 5-16: Activity panel used for editing a landmark's address. It is identical to the Enter an Address Activity panel (see Figure 5-15).

Move or Copy a Landmark to an Album

Task description: As users become more familiar with MapDroid that they may want to group landmarks in a more organized manner by using albums. If a user is not really aware of the albums feature, all their landmarks will be grouped in the default “My Landmarks” album. MapDroid allows users to organize these landmarks by providing an option to move or copy a landmark into an existing album.

Copying an album means that a twin landmark will be created in the folder that has been selected. It should be noted that none of the content associated with the landmark (photographs or comments) are actually copied to the new landmark. This limitation is enforced to preserve the unique content of the original landmark. In the case of downloaded landmarks, allowing users to duplicate another user’s content (by copying the photographs and comments) was not considered desirable.

Moving an album means that the landmark will actually be reassigned to the newly selected album. Consequently, this option isn’t available for downloaded landmarks, since users don’t have access or permission to alter the landmark’s album.

Manual operation:

- Long press landmark > Edit tab > Copy to other album. (copy landmark)
- Long press landmark > Edit tab > Move to other album. (move landmark)

Voice command: There is no voice command for either of these options.

Add an Album

Task description: As described in the *Albums* section of this paper, pg. 37, Albums are used to group landmarks. Each album has two critical pieces of information: name and privacy setting. Based on these two fields, the process for adding albums has been split up into two steps.

Name:

- The first step in creating an album is to specify a name for the album. Since names are the primary mean of differentiating between albums, each album is required to have a name, and cannot duplicate the name of an existing album.

During the first step of adding an album, a user can also add a description that further describes the contents of this album.

Privacy:

- The user must also specify a privacy setting for the album. The privacy determines who can see the landmark. Currently there are only two options: private and public. The privacy policies for albums and landmarks are further discussed in the *Albums* section.

Manual operation: Menu > Albums > Add Album

Voice command: “Add an album.”

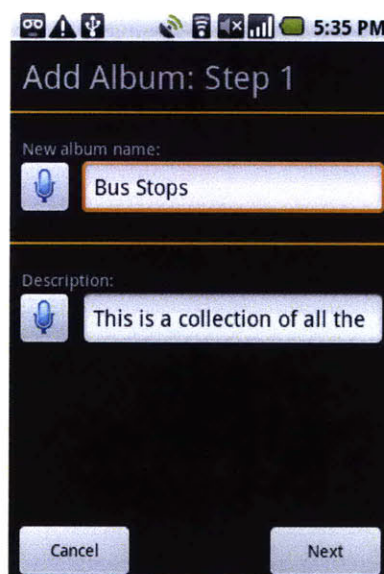


Figure 5-17: Activity panel used to create an album. It allows the user to input a name and description for the album.

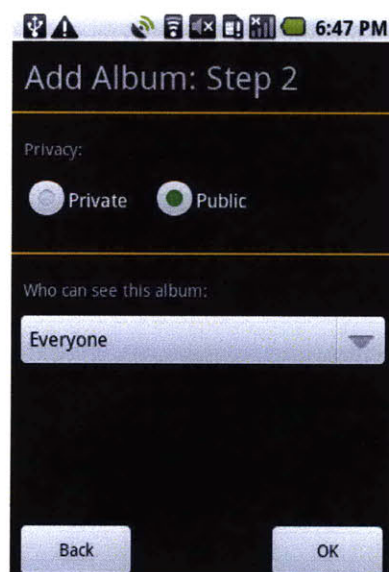


Figure 5-18: Activity panel used to create an album. It allows the user to specify the privacy level for the album.

Manage Albums

Task description: MapDroid offers users the ability to monitor the open/closed status of albums, and to alter album settings by providing a “Manage Albums” activity. Starting this activity brings up a 3-tabbed panel.

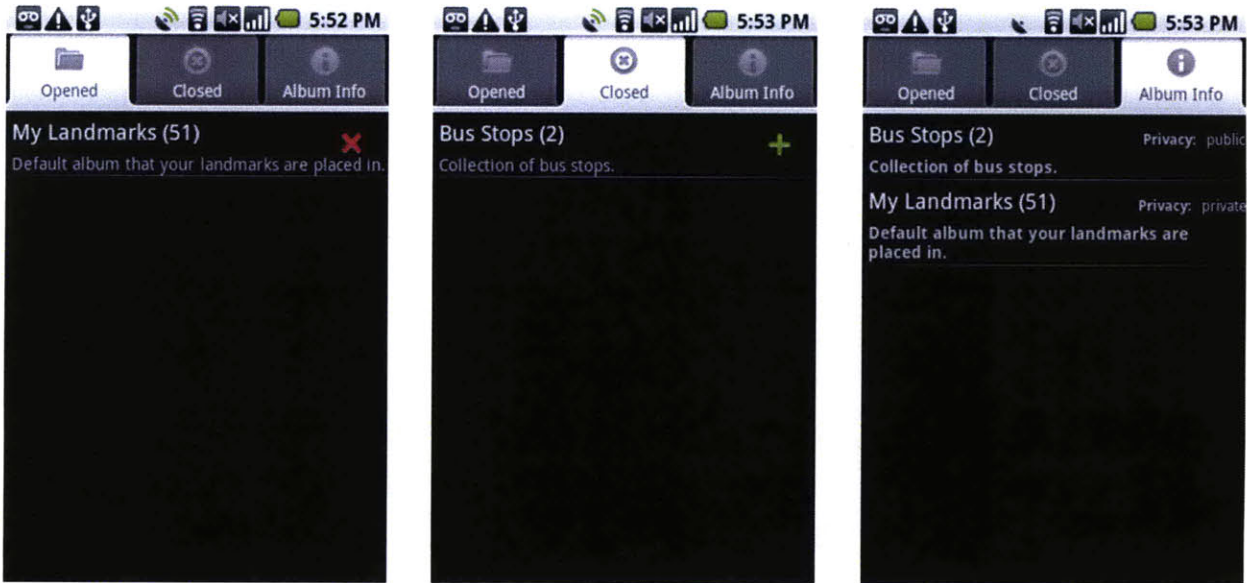


Figure 5-19: Tabbed activity for managing albums. Tabs show opened albums (left), closed albums (center), and information for each album (right).

The first tab lists all the albums that are currently opened. Clicking any of the items on this list will cause them to automatically be closed. Closing an album is a helpful way to filter landmarks, since only landmarks in open landmarks are displayed on the MapDroid main screen. For example, if a user were to go on a vacation, they may want to create an album with landmarks related to their trip. After the trip, they may still want to keep these landmarks, but not view them on a daily base. Closing the album hides these landmarks, and reduces clutter.

The second tab lists all the albums that are currently closed. Clicking an item in this list will cause the corresponding album to be opened.

The last tab, “Album Info”, lists all the user’s albums. Clicking on any of the items opens the information panel for the corresponding album. From this panel, users can access the information related to the album, or alter its name and privacy settings.

Manual operation: Menu > Albums > Manage Albums

Voice command: “Manage albums.”

Display Album Info

Task description: For completeness, MapDroid allows users to view all the information pertaining to a particular album. This information includes the album's:

- Name
- Description
- Number of landmark's contained in the album
- Current privacy setting
- Author username
- Creation date

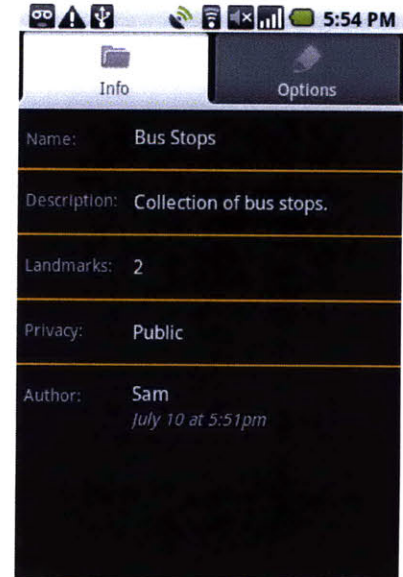


Figure 5-20: Tabbed activity that allows a user to view information about an album.

Manual operation: Menu > Albums > Manage albums > Album Info > Select album from list of albums

Voice command: NONE. This activity is not used enough to warrant its own command.

Display Album Options

Task description: For completeness, MapDroid offers the user the ability to edit an album's content. The two notable fields involve the album's name and privacy. This is a necessary functionality, since users should be able to alter whether users can see their landmarks (public privacy setting) at any point in time. Additionally, the option to directly add a landmark to the album is available.

Manual operation: Menu > Albums > Manage albums > Album Info > Select album > Options tab

Voice command: NONE. This activity is not used enough to warrant its own command.

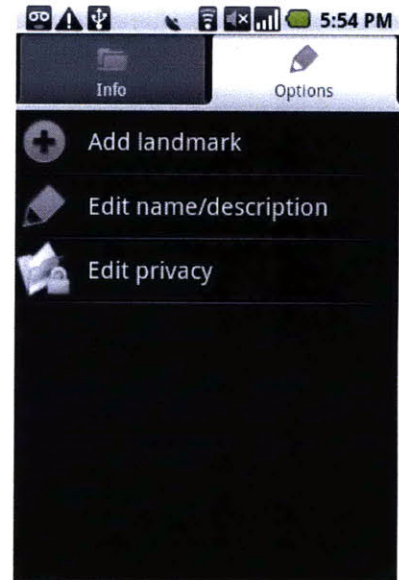


Figure 5-21: Tabbed activity that allows users to view the available options for editing or adding content to an album.

Delete Content

Task description: Besides creating content (albums, landmarks, photographs, and comments), users can also delete each item they make. The actual steps to get to the panel where you can delete an item are a little different, but the process is the same.

1. The user must be viewing the item they want to delete. For example, if they want to delete a landmark, they have to open its content panel.
2. Once a user is viewing an item, they can access the menu for the current view (hit the MENU button on the phone), and choose to delete the item by selecting this option from the menu that appears.
3. A confirmation dialog appears that warns the user that they will be deleting the current item, and all the information that pertains to it. For landmarks, this means the photographs and comments related to the landmark will also be deleted. For albums, this means all the landmarks in the album will be deleted.

It is important to mention that there is no way to recover deleted content. All deletions are final.

Manual operation:

- Deleting album: Menu > Albums > Manage Albums > Album Info tab > *select album* > Menu > Delete Album > “OK” on Delete Dialog
- Deleting landmark: Long press on landmark > Menu > Delete Landmark > “OK” on Delete Dialog
- Deleting photograph: Long press on parent landmark > Photos tab > *select image* > Menu > Delete Image > “OK” on Delete Dialog

- Deleting comment: Long press on parent landmark > Comments tab > *select comment*
> Menu > Delete Comment > “OK” on Delete Dialog

Voice command: For error prevention, there are no voice commands for this operation.

Get Distance between Two Points

Description: MapDroid offers users the ability to quickly find the distance between two points. When a user calls this function, they must first specify two different locations. MapDroid then calculates the straight-line distance (in miles) between these two points and displays it on the screen as a Toast message. Additionally, the map adjusts itself so that both points are displayed on the screen, and a line is drawn between the two points, to indicate which points the distance being displayed is related to.

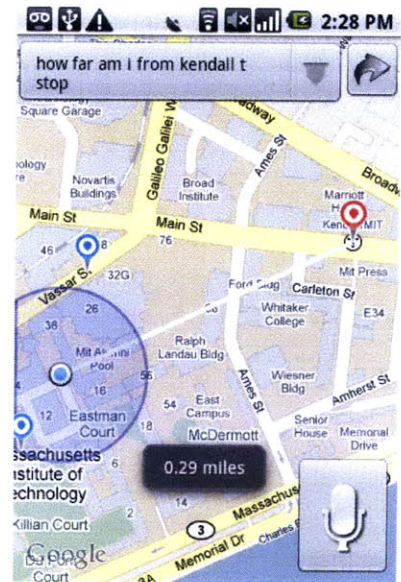


Figure 5-22: Example of screen display when calculating the distance between two points. A line connects the two points, and the distance is displayed on the screen as a Toast.

Manual operation: NONE. This is not supported since it would be tricky to do and is not a necessary functionality. The drawback, of course, is that this function is unavailable without using voice commands, and by extension, internet connectivity.

Voice command: "How far is (here | My Location | this landmark | <landmark name>) from (here | My Location | this landmark | <landmark name>)"

Get Directions to Places

Task description: One of the most useful features of Google maps is the ability to get directions between two locations. In order to make MapDroid a useful application, this feature was also included. There are two ways to get directions using MapDroid. The easiest is to specify both the starting and ending locations using a voice command. This directly starts the Google Maps process and retrieves directions between the two locations. When using a voice command, users can also directly specify what type of directions they want (driving, transit, walking or bicycle). If no direction type is specified, the default directions are for driving.

Besides being a more direct approach (single step), voice commands are useful since they are the only way a user can directly use a map point for directions. If the user activates the voice command while holding down on the map, they can refer to “here”.

A more deliberate way to do get directions is to start the “Get Directions” activity, and enter both the starting and ending locations.

Process (Directions Activity): To open the “Get Directions” activity, a user can either specify a single location using a voice command, such as “Get directions from *My Apartment*,” or manually open the process using the application’s main menu. The activity requires both “from” and “to” points. To enter these locations, a user can either use speech with the Google recognizer, or select the bookmarks button to the right of the field to start the “Location Picker” activity.

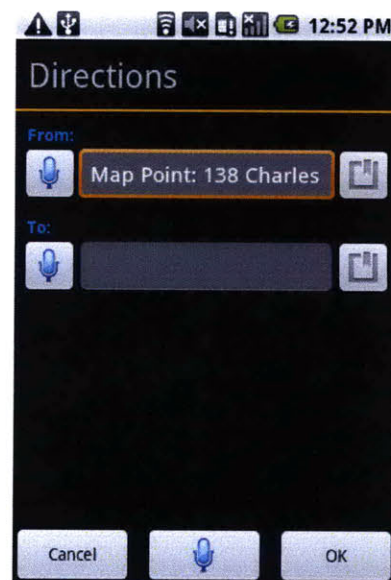


Figure 5-23: Activity panel used to select two points to get directions between.

When a user uses the Google recognizer to input speech for either location, the transcribed speech is returned to the application. The resulting best match then goes through a series of processes to determine the point the user wants to use.

1. The application initially starts by stripping “to” or “from” from the start of the string, just in case the user has inadvertently specified this information.
2. The new string is then compared against “my location” to see if the user would like to use their current position. If the strings match, then the application checks to see if My Location is enabled and the GPS has a lock. If this fails, then a message is toasted to the screen informing the user that My Location is not enabled. Otherwise, “My Location” is displayed in the field, and used as the point of reference.

3. If My Location was not the location specified by the user, the string is sent to the phone’s database to see if it matches any of the names of landmarks. All the landmarks that contain the string in their name are returned. In the case where there is only one match, this landmark is automatically chosen as the point of reference and its name appears in the field. In the case where there are multiple matches, a Dialog window appears that forces the user to select a single landmark to use.

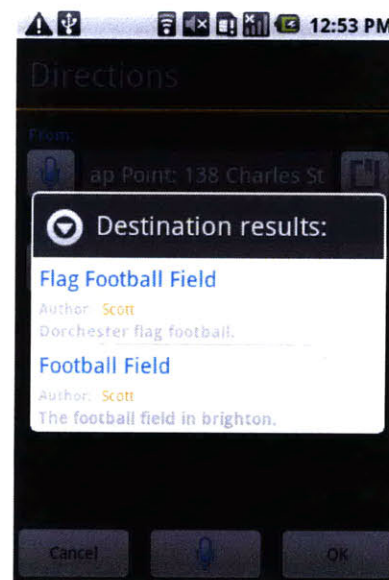


Figure 5-24: Dialog used to help disambiguate which landmark to get directions for.

4. In the case where a string doesn’t match “My Location” or any landmark names, the application assumes that the user has entered a specific name or address that defines the location. The string is passed into a Google geocoder, and if the geocoder can produce a

geographical point for the input string, then the string is accepted. At this point, the string is pasted into the input field. If the string does not result in a location (which is very rare), then a message is toasted to the screen asking the user to retry their entry.

A user can also use the speech button at the bottom of the panel. When this speech activity is used, the application assumes the user is trying to enter both locations in a single step. The user is prompted to use a “From ... to ...” format when requesting directions. The resulting string is split up into two strings if the word “to” is present. Both strings are then processed in the same way a single “from” or “to” request is processed. It should be noted that this can be a tricky process since a user entering an address, such as “From to 32 Vasser Street, Cambridge” can be recognized as “From 232 Vasser Street, Cambridge”. From preliminary evaluations however, this process works well.

If a user clicks the bookmarked button to the right of either field, the “Location Picker” is activated. The location picker offers four different ways a user can enter a location. The different entries are:

- **My Location:** A user can simply use the current position of the phone if it is available. This option is only enabled when the phone currently has a GPS lock.
- **Enter address:** When this option is selected, the *Enter an Address* Activity, pg. 82, is started and a user can enter an address for a location.

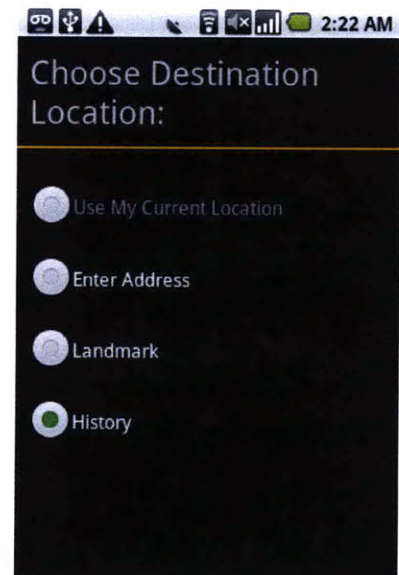


Figure 5-25: Activity panel that allows user to specify a location.

- **Landmark:** The user can use the Search Activity to select a landmark to get directions to/from it.

- **History:** The user can also use the history shortcut and select any of the most recently used locations.

When the history option is selected, it opens a list of the most recent locations that have been used for getting directions. The history feature is available as shortcut for users who frequently get directions to/from the same locations. Each location in the list is represented by an icon and its name. It should be noted that addresses are represented by a globe, and map points are represented by a thumb tack.

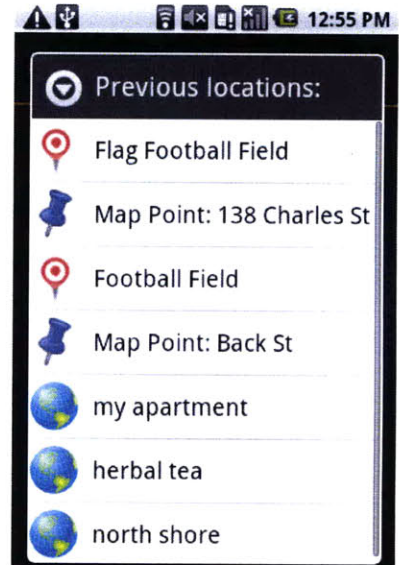


Figure 5-26: List of previous points used to get directions for.

As in Google maps, users can also choose to reverse their start and end points. By click the Menu button, they can choose the “Reverse” option, which swaps the two locations. Users can also clear their directions history, by choosing “Clear History” from the menu bar.

If start and destination points have both been specified, the activity opens the Google Maps Directions activity. This is a bit of an annoyance, since the user has to leave MapDroid to view the directions (which are specified by their geographical points when they are passed into Google maps). MapDroid is forced to do this, due to the restrictions Google has placed on developers using overlays on Google Maps to assist with real time navigation. The benefit, of

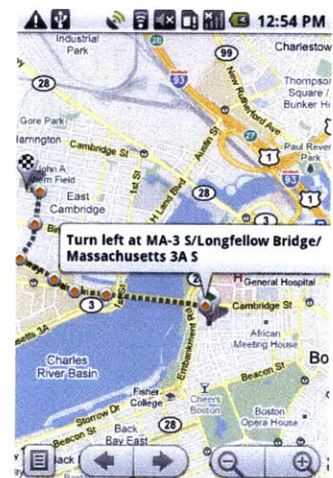


Figure 5-27: Typical map display for directions on google Maps.

course, is that the Google navigation activity is much more robust than any navigational activity that MapDroid could implement.

In order to get back to MapDroid after Google Maps has been started the user must click the phone's back button multiple times. Once the user has returned to the MapDroid application, a message that says, "Welcome back to MapDroid!" is toasted on the screen. This helps users know when they have finally returned, since the home screen of MapDroid looks very similar to the Google Map screen.

Manual operation: Menu > Internet Options > Get Directions

Voice command:

- "Get [driving | transit | walking | bicycle] directions (from | to) (My Location | here | this landmark | <landmark name>)." (indirect)
- "Get [driving | transit | walking | bicycle] directions from (My Location | here | this landmark | <landmark name>) to (My Location | here | this landmark | <landmark name>)." (direct)

Find Landmarks

Task description: Finding a landmark is an essential task, and also one of the functions that benefits the most from speech.

From the home screen, users can use voice commands to directly find landmarks using:

- Name
- Tags (see *Building the Grammar*, pg. 115)
- Proximity
- City / State

MapDroid uses the input criteria, and filters the landmarks

accordingly. If no landmarks match the request, then a message

is displayed that states that there were no results found. If a single landmark is found, it

automatically is chosen as the selected landmark. In all cases where landmarks are found, the

map adjusts so that it is at the highest resolution that displays all the search landmarks. As

mentioned in *Landmark Markers*, pg. 50, all search landmarks are represented as balloon shaped icons with an “S” symbol.

Manual operation: Menu > Landmarks > Find Landmark (This will start the Activity described under *Search Activity*, pg. 100).

Voice command: “Find [<tag name>] landmarks [named <landmark name>] [in <city name> | <state name>] [within <number> miles of (My Location | here | this landmark | <landmark name>)]

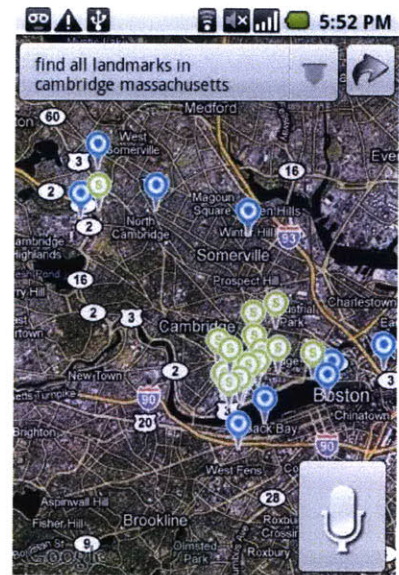


Figure 5-28: Home screen display for landmark search. Landmarks matching the search results are identified by green "S" icons.

Find Landmarks along a Street

Task description: Since streets can't be used in voice commands, MapDroid allows users to search for landmarks along a visible street. If the user activates the main screen voice command by pressing down on the map, they can search for landmarks along the street they are touching. In order to do this, MapDroid takes the geographical point on the map, and uses the Google geocoder to see if the point has a known address. If the location has an address, then the street name is extracted from it and passed into the phone's database. All the landmarks that contain the provided street name are then displayed as search markers on the screen.

Manual operation: Menu > Landmarks > Find Landmark > Address tab > Enter street name > Search

Voice command: "Show <tag> landmarks on this street" (while pressing down on street)

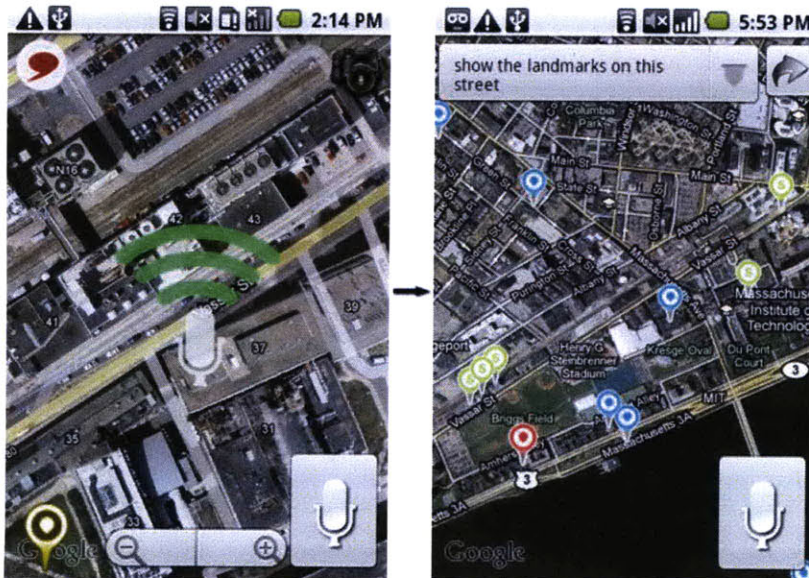


Figure 5-29: Process for searching along a street: A user must hold down on the street seen on the map, using the touch screen, and say the voice command (left). The map adjusts to display all the landmarks located on the specified street (right).

Search Activity

Task description: One of the important features for MapDroid is the ability to find landmarks. Users can search for landmarks using voice commands, but there is also a separate search activity which allows users to manually find a landmark. The Search activity allows users to search for landmarks using:

- Name and/or description
- Proximity
- Address (street, city, state, zip)
- Author username

Each of these four search options is assigned to its own page on a 4-tabbed search panel.

Name and/or description: The Name tab of the search activity allows users to search for landmarks using name and/or description. The page includes two text fields for both fields, and each field contains an internal list of all the open landmark names and descriptions. A landmark can easily be selected by typing its name or description because both fields use an autocomplete method to filter out landmarks that don't match the input text.

MapDroid attempts to be clever in its search approach. If a user selects a landmark from the dropdown list that appears

below either field, the Search activity assumes the user is attempting to select a single landmark.

In this case, the activity returns the selected landmark's ID and closes. Otherwise, the

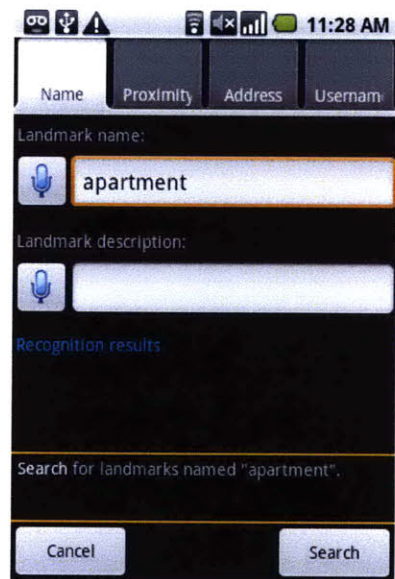


Figure 5-30: Tabbed activity panel used to search for landmarks by name and/or description.

application waits for the “Search” button to be pressed before it uses the input to complete the search.

Proximity: The proximity tab allows users to select landmarks which are located within a specific distance of a chosen point. The tab has an “Add location” button that allows users to add multiple points to the proximity search. Clicking this button causes the “Location picker” activity (see *Get Directions to Places*, pg. 93) to be started. A user can then choose a point, using a variety of location types, to center their search around.

Once a location has been selected, the Search activity regains focus. A user can then specify the proximity search radius by either typing a distance into a text field or by using a

scroll bar that ranges from 1-100 miles. If a user wants to focus their search around multiple points, they can simply add a new location by hitting “Add location” and selecting an additional point. Furthermore, all the current search locations are listed underneath the “Add location” button. If a user wants to remove a location from their search, they can simply press the red remove button that appears to the right of the location name.

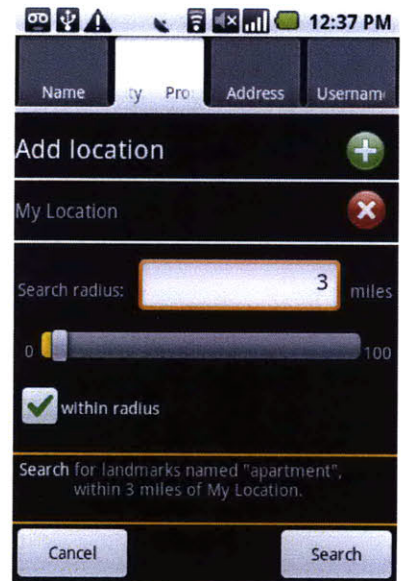


Figure 5-31: Tabbed activity panel used to search for landmarks by proximity.

Address: The address tab allows users to filter their search by the physical address of a landmark. Users can search using a landmark's street, city, state, or zip code. Each of the four fields is paired with a button that will start the Google recognizer. The results from these activities are simply pasted into the corresponding field. Users can also use the talk button at the bottom of the page and enter a complete address (which is similar to the design of the *Enter an Address* activity, pg. 82). In the case where the talk button is used to enter an address, the Search activity tries to be clever. It assumes the user is entering a specific address for a single landmark. Consequently, using the talk button will start the search activity for the application, strictly based on the provided address.

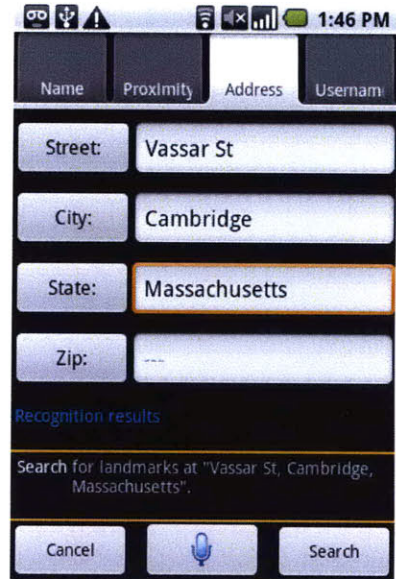


Figure 5-32: Tabbed activity panel used to search for landmarks by address.

Username: The Username tab allows users to search for landmarks based on the username of the landmarks author. This search feature becomes very useful if a user has downloaded content from a specific user and would like to find all the landmarks for that user. The pages layout is very simple. It contains a single text field that a user can either type or say the author's username.

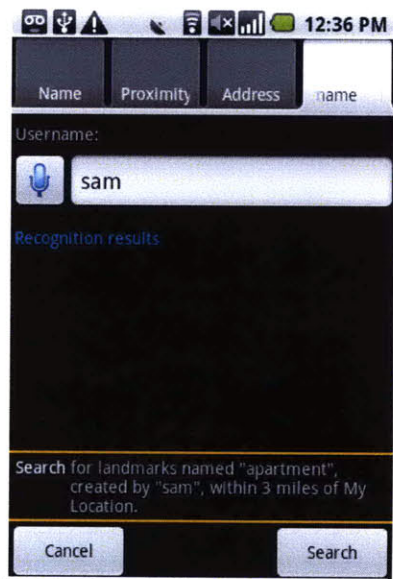


Figure 5-33: Tabbed activity panel used to search for landmarks by username.

Executing the Search: When a user selects the “Search” button, all the criteria that have been entered into the 4 tabs are used to complete the search. If anything has been entered into the landmark name, description, author username, street, city, state, or zip code fields, then these values are combined into a single search request. Additionally, if the proximity search is used, the bounding box (using latitude and longitude coordinates) for the search radius is calculated. The search request then requires that all landmark coordinates be within the bounding box. This step is performed in order to filter out the majority of landmarks that don’t meet the proximity criteria, since calculating the actual distances between points can be time consuming. This request is then passed to the phone’s database, and the results are returned.

If the search results in no matches, then a message is displayed to the user that notifies them that no results were found for the query. In the case where there are results and a proximity search was used in the query, then the results are further refined, since the initial request only required landmarks to reside within the bounding box of the proximity request. To accurately return the landmarks that meet the proximity requirements, the distance between each resulting landmark and the proximity points is calculated, and all landmarks that aren’t closer than the specified radius are removed.

If there is a single landmark that meets the criteria, then the Search activity immediately returns the landmark’s ID and closes. If there are multiple search results, however, then they are listed (using name, description and author) in a dialog box. The user can then select a single landmark by tapping it, or choose all the listed landmarks.

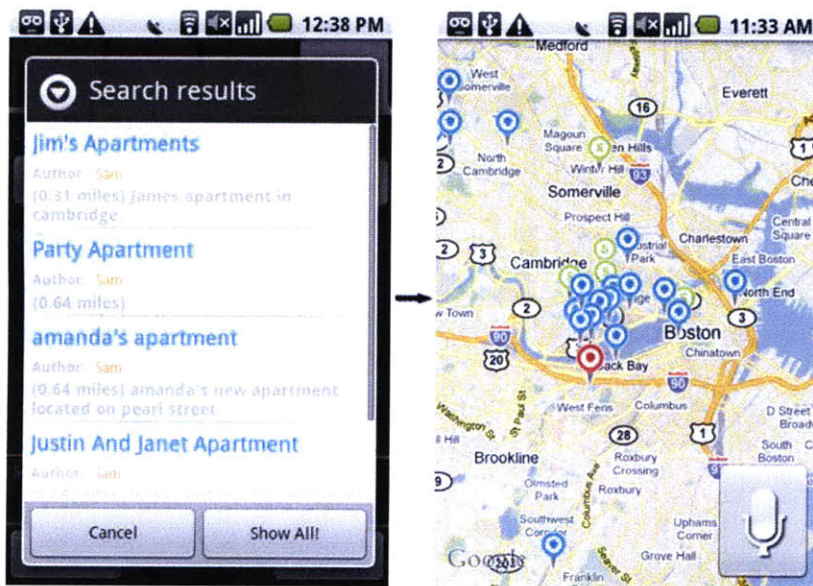


Figure 5-34: Process for displaying search results. After the Search button is hit, all valid results are listed and the user can select one landmark or all of them (left). All selected landmarks are then displayed on home screen as green icons (right).

Interface design: There are a couple user interface design choices related to the Search activity that are worth pointing out.

All the input fields for this activity are speech enabled with the Google recognizer. In other parts of the MapDroid application, N-best results returned by the recognizer are stored in the autocomplete list of the corresponding text field. Because many of the input fields for the Search activity contain a list of landmarks, it is not desirable to replace this list with the N-best results. Instead, a label is included at the bottom of each tab, labeled “Recognition results”. The field is blue, indicating that it is an active link that users can click. Clicking the label brings up a list of the N-best results from the last speech request. Clicking any of the items on this list will automatically paste it into the field that the last request corresponded to this. By including this, MapDroid is able to preserve the ability to filter landmarks with text, and also display recognition results.

One important feature of the Search activity is that all the search criteria are displayed in an easy to read format directly above the Search button. This field can be seen at all times, as users switch from tab to tab. It is provided so that users can get a better understanding of how the application is interpreting their search input, and to remind the users of the criteria they have entered. It is important to remind users about the values they've entered, since they lose focus if the user switches to a different tab.

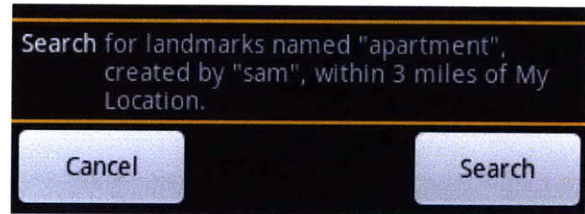


Figure 5-35: Textual representation of search criteria display.

Download Landmarks

Task description: One of the main features of MapDroid is the ability to share content with other users. In order to share content, a user must upload landmarks to the server (all “Public” landmarks are uploaded when the phone is synchronized). To retrieve this content, a user must be able to “download” it. Users can download landmarks by starting the Download activity. This activity uses the same interface and mechanism as the *Search Activity*, pg.100. The only significant difference between the Search activity and the Download activity is that the search request that is generated by the activity is sent to the external database (when Downloading) instead of the phone’s database (when Searching). When there are multiple landmarks, the number of landmarks is displayed in the result dialog’s “Download All” button, so that users can easily see how many results matched their query and whether they actually want to download all the listed landmarks.

There is a limit on the number of landmarks a user can download. This limit has been arbitrarily set to 1000. The chief concern is that if a user accidentally downloaded too many landmarks, the system could become very slow and unresponsive.

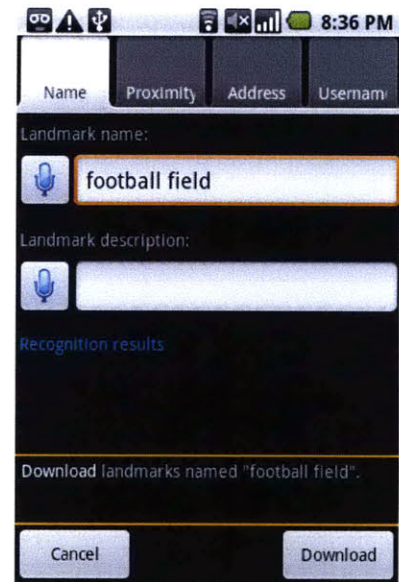


Figure 5-37: Tabbed activity used to download landmarks. The activity uses the same interface as the Search Activity.

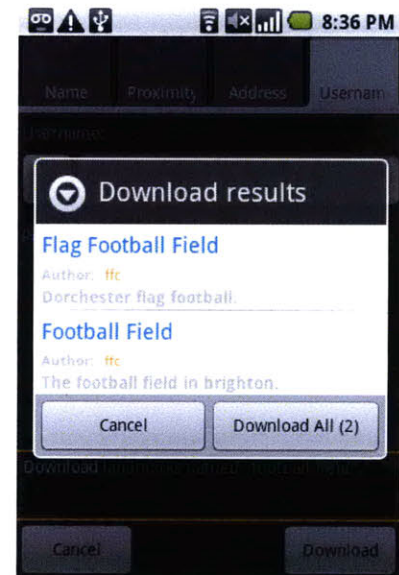


Figure 5-36: List of download results users can select from.

The actual number of visible landmarks that would make the application too slow to use is not currently known, since there are only 100 available landmarks currently used in testing. This maximum limit could be adjusted, however, if users ever ran into this problem.

Manual operation: Menu > Online Options > Download Landmark

Voice command: "Download landmark."

Clear Search or Download Results

Task description: Because search results are represented by different icons and users can download hundreds of landmarks, MapDroid offers a couple of simple options to clean these results up. A user may choose to clear search results, which will remove the “search status” from landmarks and make them appear as regular landmarks. Additionally, users may clear all downloads, which will remove all downloaded landmarks from the phone’s database, thus removing the landmarks from the screen.

Manual operation:

- Menu > Landmarks > Clear Search Results
- Menu > Online Options > Clear Downloads

Voice command:

- “Clear search results.”
- “Clear downloads.”

Sync Phone

Task description: As described in *Sharing Content*, pg. 39, users must have the ability to sync their phones with the external database. This allows a user to push all the new public data from their phone into the server database, and pull all the new content for downloaded landmarks onto their phone's database. If a user attempts to sync their phone, but does not have internet connectivity, a message is Toasted to the screen that notifies them that their phone can't be synchronized right now. If their phone can be synchronized, a progress dialog appears that notifies them that their phone is being synchronized.

Additionally, if users only want to update an individual landmark, they can do so by selecting this option from the landmarks options list in its Edit tab (see *Display Edit Options and Commands for a Landmark*, pg. 79). This saves time for the user, since they don't have to update every landmark on their phone, which can be a time consuming process if there are a lot of un-synchronized photographs.

Manual operation:

- Menu > Online Options > Sync Phone. (sync all landmarks)
- Long press landmark > Edit tab > Sync landmark. (sync individual landmark)

Voice command: Because this operation should seldomly be used, no voice commands were included for this operation.

Undo or Redo the Last Command

Task description: Many of the commands that are speech enabled on MapDroid result in the map changing state. For example, a user may be looking at a landmark in Florida, but if they say, “Zoom in on Stata Center,” the map may zoom in on a building in Massachusetts. If that’s not what the user intended to do, then this can be difficult to correct since it might require them to zoom way out, find Florida, and then try to zoom in on the point they were looking at. Because misrecognition happens, this becomes a very real problem when using voice commands.

The “undo” command on MapDroid helps combat this problem by restoring the map state from the previous command. Map state consists of the

- map type (street or satellite)
- map overlays (traffic or street view)
- map center (latitude, longitude)
- zoom level
- current state of My Location (on or off)

When undo is called, the values for these map fields are assigned to the last known state. Users can also undo multiple steps. For completeness, MapDroid offers a “redo” feature that allows users to reverse the “undo” process by reapplying a map state.

Because the application has to write the map state to the phone’s database before it can execute a new command, this slightly increases the response time of the system after a voice command. Since users may not even want to use the Undo/Redo capabilities of the application, this feature is initially turned off to improve system performance. Users can turn this feature on/off however, by selecting the appropriate option from the “More” menu item.

It should be noted that the name of the operation is misleading, since it doesn't actually undo the last command. For example, if the last command was "Clear all downloads", then calling "undo command" won't restore all the landmarks that were just deleted by the previous command.

Manual operation:

1. Menu > More > Turn on Undo/Redo (if option is off)
2. Menu > More > (Undo | Redo) Command

Voice command: "(Undo | redo) last command."

Help for Voice Commands

Task description: Novice users of MapDroid likely won't know what they can do or say. In order to help users figure out what commands are available, and how to use speech to call them, MapDroid provides a helpful list of commands coupled with their corresponding voice commands. Each list item has a description of a MapDroid capability displayed as single white line. Below the function, in blue text, is an example of how to execute the command using speech.

Manual operation: Menu > More > What can I say?

Voice command: "What can I say?"

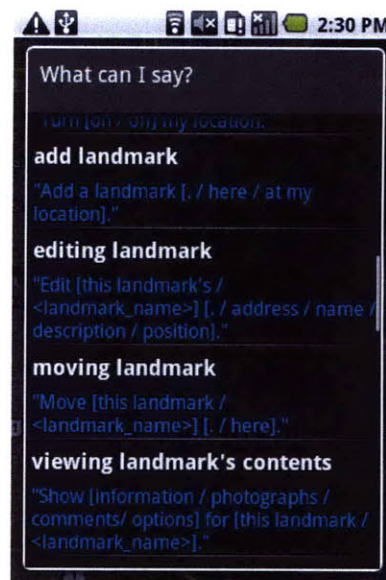


Figure 5-38: Help panel used to teach users what actions they can invoke using voice commands.

Help for Main Menu

Task description: To help users learn what functions can be performed using the main menu, MapDroid provides a help dialog for the main menu. Almost all the actions a user can perform using the main menu are listed in white. Below each function, in blue text, is a description of how to navigate the menu in order to start the corresponding action.

Manual operation: Menu > More > Help for main menu

Voice command: “Help for main menu.”

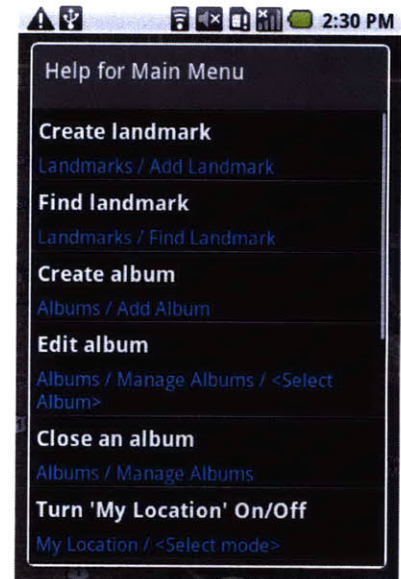


Figure 5-39: Help panel used to teach users what actions they can perform using the application's Main Menu.

Close the Application

Task description: While the phone's back button performs the same process, for completeness, MapDroid includes both manual and speech options to exit the application. To protect against cases when a user accidentally attempts to exit the application, a dialog message is shown that forces the user to explicitly close the application.

Manual operation: Menu > More > Close application

Voice command: "Close application."



Figure 5-40: Confirmations dialog displayed to users before they exit the application.

6 Using Speech in MapDroid

6.1 Introduction

To recognize speech, MapDroid uses an external recognizer that is run on a server. On start up, MapDroid sends the recognizer a JSGF grammar⁶ so it will know what to listen for. When a user inputs speech, the speech is recorded, and sent to the recognizer. The recognizer then returns an N-best list of the results it produced from the audio. These results are processed by MapDroid, and the most likely result is executed.

6.2 Building the Grammar

MapDroid uses a JSGF grammar that specifies the expected input commands which defines what users can say. The grammar has been built in a very straightforward way. Every line of the grammar corresponds to a specific command. For each command MapDroid offers a number of different ways to say the command, so there may be multiple lines in the grammar that are all dedicated to the same command. Each line also includes a result code. The result code is simply a number that specifies which operation should be performed by the application if the command is spoken.

In order to make the grammar as natural as possible, it is important for users to be able to reference a variety of values that are associated with landmarks when issuing voice commands.

These values include:

- Name of landmark
- Description of landmark

⁶ <http://java.sun.com/products/java-media/speech/forDevelopers/JSGF/>

- City and State landmark is located in

MapDroid's grammar allows users to reference these fields in appropriate commands. When these fields are used in a command, the application must extract their values from the result string, and use them as parameters in the operation that is invoked by the command.

A typical command may look like, "zoom in on Rudys [r=312]". The result code for this command, 312, is used to start that operation that zooms in on a landmark. The name "Rudys" must be extracted in order for the application to know which landmark to zoom in on.

Because MapDroid's grammar doesn't allow it to handle freeform speech, it includes a set list of 35 descriptive tags that describe types of landmarks (eg. beaches, colleges, parks, etc...). Users can reference any of the provided tags when they are searching for landmarks by description. For example, a user can say, "Show me all the colleges". MapDroid can extract the "colleges" tag, and check the description of every landmark to see if it contains the term "college". Any landmarks that contain the tag will then be displayed as search results.

For the most part, MapDroid uses a static template for its grammar. This template includes all the commands, descriptive tags, state names, and other terms a user may reference. However, the template does not include a list of landmark names or cities. Producing a premade list of landmark names is impossible since new landmarks are constantly being created. Additionally, there are so many different cities that including all of them in the grammar would make the recognition process very slow. For this reason, MapDroid must generate an appropriate list of landmark names and cities every time it sends the recognizer its grammar. In order to make these lists, MapDroid grabs all the landmarks (regular and downloaded) from the phone database, and iterates through them. For each landmark, it stores the landmarks name and city in a set. These sets are then appended to the grammar template, which can then be sent to the

recognizer. In order to handle dynamic content, MapDroid simply repeats this process of extracting cities and names every time a landmark is added, updated, or deleted. It then resends the new grammar to the recognizer, so the user can reference the most up-to-date content.

If the application hasn't had any landmarks added to it, it uses a truncated version of the main grammar, that doesn't include any commands that reference existing landmarks. For example, "zoom in on this landmark," is not included in the initial grammar, since MapDroid can't even execute this command given its current state. As soon as a landmark has been added, MapDroid switches to the main grammar, which contains all references to landmarks and their contents. A version of the full grammar is included in Appendix B under Grammar, pg. 149.

6.3 Handling the Results

When the results are returned by the recognizer, MapDroid has to do two things:

1. Execute the most likely command
2. Display the most likely command, and provide the N-best list of results for viewing

In order to execute a command, MapDroid must extract the result code value. If the command includes parameters (names, descriptive tags, cities, states, or numbers), then these values also must be extracted from the result. The result code is then passed to a method that checks the value against all known result codes. If there is a match, the method or function associated with a given code is called. If the operation uses parameters, then the values that were stripped from the result string are passed in.

6.3.1 Disambiguating landmarks

When landmark names are used as input this can cause problems since multiple landmarks may have the same name. A simple case would be if the user generated landmarks at a few different colleges and naively labeled two different landmarks as “Student Center.” This becomes a challenging problem when the user uses this name in a command such as “Zoom in on Student Center.”

To ensure that a single landmark is selected in this situation, MapDroid matches the input name against all visible landmarks. If the name is shared by multiple landmarks, a Dialog window is shown that lists the multiple landmarks with their name, description and author listed. The user must then select one of these landmarks in order to complete the command.

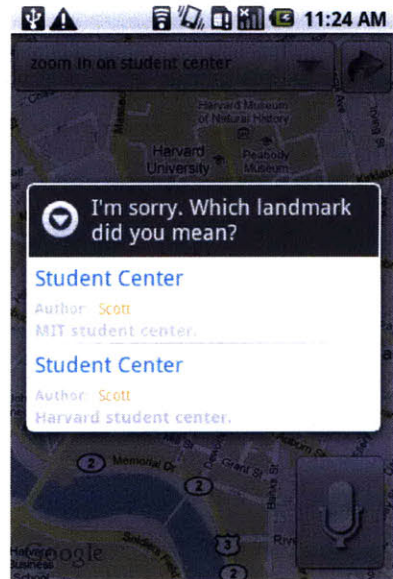


Figure 6-1: Dialog displayed to users that forces them to choose between landmarks of the same name.

6.4 Displaying Results

In order to combat misrecognition, it is important to provide an N-best list of results that a user may select from if the best result produced by the recognizer doesn't match the user's input.

Since the recognizer often produces many variations of the same command, simply displaying all the results that are returned isn't very helpful for the user. To solve this problem, MapDroid only shows commands that will result in different actions being performed. Only the most likely result for each command is displayed, and all variations of the same command are ignored.

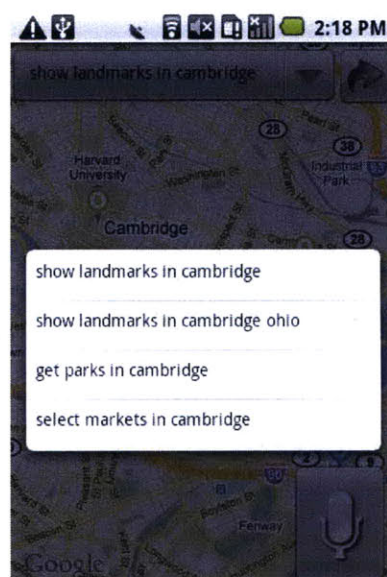


Figure 6-2: N-best list of recognition results for a command.

Initially, when speech results are returned, only the command that was just performed is shown in a drop down menu (see *Use of Speech*, pg. 46, for a fuller description of the interface design). If a user opens the drop down menu for the speech results, they are presented with the N-best list of commands. When an item from the N-best list is selected, the list collapses, and the selection is displayed as the current selection. The raw command that was stored with the displayed command is then passed into the method that handle's recognizer results, and the selected command is executed. This process allows users to easily correct a misrecognition error if the desired command has been listed.

6.5 Restriction of Grammar

MapDroid uses a context-free grammar that is specified in a JSGF file. One of the biggest problems with using a context-free grammar is that it only covers a finite range of what users

might say. The recognizer that is built using the application's grammar can only attempt to fit the input audio to the word sequences provided in the grammar. If a user says an out of vocabulary word, or even switches the order of words in a command, misrecognition is guaranteed.

This restriction is especially visible with respect to the use of cities and landmark names. Since MapDroid builds the list of cities from landmarks then only cities that contain landmarks can be referenced. This is a reasonable expectation, however, since allowing users to search for landmarks in cities that obviously don't contain any landmarks doesn't benefit the user. Similarly, this restriction also places a strain on the user to remember the exact name of a landmark when they reference it. If they leave out a word, or swap words, it can cause the recognizer to misrecognize the command.

An alternative way to recognize speech is to train an n-gram language model, which would allow for a much broader range of input, and increase the flexibility in the way users can verbally interact with the application. However, this would require natural language processing in order to determine the intent of the command and the values (names, cities, etc...) contained in the command. This idea is discussed in more detail under *Improving Speech Recognition*, pg. 135.

6.5.1 Defense of Context-Free Grammars

The main reason that MapDroid uses a context-free grammar is because it was not built to be a conversational interface. It is built around a limited number of features and commands. An expert user, who wants to be as productive and efficient as they can be with the application, will learn a finite number of verbal commands to interact with the system. Because a context free

grammar only has to consider a small range of phrases, it has an excellent chance of recognizing something correctly, assuming the user says a command that is provided in the grammar.

6.6 Teaching MapDroid Syntax

Since MapDroid uses a set list of commands, the system's success depends on users learning the proper way to say commands. MapDroid attempts to teach users what to say in two ways. The most basic feature offered is a help menu that gives examples of actions, and how to call them with voice commands. This help panel can be opened by asking, "What can I say?"

Additionally, MapDroid use labels for panels and menu items to help teach users the correct syntax. Since almost every operation available in the application's main menu can also be invoked using a voice command, MapDroid uses a 1-to-1 naming technique. Every menu item, list option, and panel name is labeled with the same phrase that can be used to call the command. For example, if a user is adding a landmark, the corresponding panel that is used to add the landmark, is labeled "Add Landmark". This naming scheme can become a little awkward when labeling menu items, like "Turn on my location", instead of simply providing a succinct menu item label such as "On".

We also attempted to broaden the application's grammar by learning what users would like to say. MapDroid used Amazon's Mechanical Turk⁷ site to post tasks that asked users to submit a variety of commands that they would theoretically say if they were using the

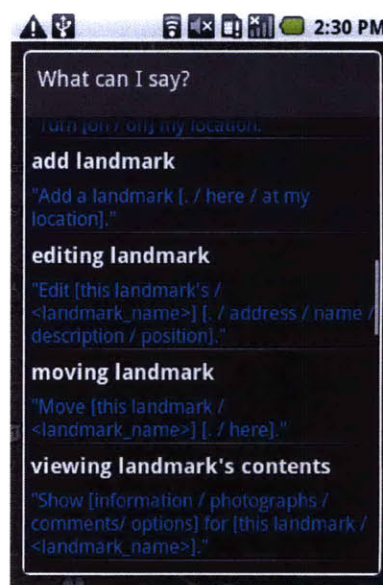


Figure 6-3: Help dialog for teaching users what they can say to the application.

⁷ <https://www.mturk.com/mturk/welcome>

application. The results from these tasks were used to expand the applications current grammar and make the system more robust. The process and results of this experiment are presented in *Using the Mechanical Turk Service to Help Build a More Robust Grammar*, pg. 123.

7 Using the Mechanical Turk Service to Help

Build a More Robust Grammar

7.1 Problems with Context-Free Grammars

MapDroid's use of a context free grammar imposes some very strict restrictions on what users can say. If users say a command in a way that is not included in the application's grammar, then it is likely that the recognizer won't be able to correctly recognize the command. In order to build a more robust grammar, we attempted to learn what users might say by gathering input through the use of Amazon's Mechanical Turk⁸ service.

7.2 Using Mechanical Turk to Gather Possible Commands

The first step in bolstering the application's grammar, was gathering the various ways a user might interact with the system. To do this, we used Mechanical Turk,⁹ which is an Amazon web service. The site allows requestors to post Human Intelligent Tasks (HITs) for Turkers (users who complete the HITs) to complete. The basic strategy was to get Turkers to enter various ways they would verbally interact with MapDroid. These commands could then be used to improve the application's grammar, since sample input from other users provides a wider range of key words and phrases than a grammar created by a single person. Results from Turk Task 2.0 are included in the Appendix under *Turk Task 2.0 Results*, pg. 141.

⁸ <https://www.mturk.com/mturk/welcome>

⁹ <https://requester.mturk.com/mturk/resources>

7.3 Setting up the Turk Tasks

The hardest part of gathering helpful user input is creating a HIT that doesn't bias the Turkers input, but provides enough structure so that the Turkers understand what types of interactions they should simulate. For this project, three different types of HITs were used. Each included one paragraph of background information that briefly described the MapDroid application and its purpose.

7.3.1 Turk Task 1.0

The first task that was posted simply included the application's background and various screenshots of the application in use. The HIT then requested the Turkers to supply four ways they might verbally interact with the system.

The results we received from this HIT were not very useful because the HIT was too vague. A large portion of the inputs were out of domain, meaning that they were supplying interactions that MapDroid does not have the capability of performing (such as, "Where is the closest zoo?").

7.3.2 Turk Task 2.0

The second task that was posted included a screenshot of the application after it had finished performing a command. The speech results for that command were still visible on the provided screenshot. Turkers were asked to use the provided command and come up with four different ways they might similarly interact with the application.

The results from this task were mostly in domain (roughly 70%) since users were inputting different ways to say an in-domain command. However, the input was also noticeably biased by the language used in the given command. For instance, if the given command was “Show all my landmarks,” then almost all the Turkers used the word “*landmarks*” in their commands. This is not particularly helpful since the purpose of this task was to expand the system’s vocabulary and grammar.

7.3.3 Turk Task 3.0

Because some of the Turker input from *Turk Task 2.0* was still out of domain, the third Turker task added a little more structure. Turk Task 3.0 used HITs that provided a screenshot of the application in an initial state, and then provided a second screenshot of the application in a different state after executing a verbal command. The speech results for the command were visible in one of the two screenshots. Turkers were then asked to supply four different ways they might verbally interact with the system in order to produce the same transition or result.

Providing more information had mixed results. In some situations, such as “Add a comment to this landmark,” providing a screen shot of the application adding a comment helped the Turkers focus their input on how to request the application to add a comment to a point of interest. However, for the HIT “Add a photograph,” a screenshot of the phone preparing to take a picture of an office was provided, which caused many of the Turkers to start to focus on the screenshot’s contents. A number of Turkers added phrases like “Take a picture of my office,” which isn’t useful. From this iteration of HITs, it became clear that a developer has to be careful in choosing what they show Turkers, since providing any extraneous information often leads to poorer results.

7.4 Data Collected

From the three batches of HITs, over 300 hits were collected (roughly 100 for each batch). Only about 2/3 of the responses were used, which resulted in about 800 useful phrases (four responses were required per HIT). Many of the phrases were only two to three words in length, which is more a product of Turkers trying to quickly complete each HIT, and isn't a very good representation of how a real user would interact with the application (usually between 3-5 words based on preliminary user tests).

7.5 Modifying MapDroid's Existing Grammar

In order to make use of the data that was collected from the three small batches of HITs, all the phrases that were deemed relevant (roughly 800) were sorted alphabetically and by frequency of occurrence. This made finding the most common ways potential users would interact with the system easy. The most popular phrases and terms were then integrated into the existing grammar. The changes to the grammar involved inserting optional words (eg. "Show [me] [all] [the] landmarks in..."), adding new words to word classes (eg. Placemark or marker to the <landmark> class), and adding entirely new phrases (eg. "Bookmark my current location").

7.6 Collecting Test Data

In order to test the effect these modifications to the grammar had on the application's recognition rates, a small batch of user tests were performed. To collect test data, users were asked to finish 18 different tasks using speech on the application. These tasks almost covered the complete

range of functionality that MapDroid has to offer. The range of tasks included adding landmarks, photographs, and comments, getting directions, and finding landmarks. Before starting the tasks, each user was briefed on the application's purpose, and was given five minutes to play around with the application using speech commands.

Five different users completed this set of tasks. From these tests, 193 spoken phrases were collected. Of the 193 interactions, 160 were in-domain, while 33 were out of domain (eg. "Show medieval weapon stores").

7.7 Testing the Grammar Modifications

To test how altering the application's grammar would affect the recognizer's recognition performance, three different grammars were used. The original grammar that the recognizer used while the users performed the tasks was used as a baseline. The recognition rates of the original grammar were compared against the performance of the new grammar that included the various phrases and words that were extracted from the Turker data. Lastly, to compare the confusability of the actual commands, we built an oracle grammar out of only the spoken phrases that were used in the test data. Consequently, the oracle grammar can be used as a measure of the best possible results that can be expected.

When running these tests, there were three ways to measure the recognition accuracy. The first two were word error rate and sentence error rate. The third way to measure the misrecognition rate was "command error rate", which represents how often the recognizer produced a result that did not match the spoken command. This is the most useful measure, since the application is primarily concerned with completing the correct action.

The three different grammars were tested against two different test sets. The first test set included all test data (193 commands). The second test set included only the in-domain data (160 commands). The different test sets were used to see if the new grammar would perform significantly better on out-of-domain data than the less robust grammar.

Results

	Word Error Rate	Sentence Error Rate	Command Error Rate
Original grammar	36.4%	70.5%	51.3%
New grammar	31.4%	60.1%	43.5%
Oracle grammar	1.6%	6.7%	4.1%

Table 7-1: Error Rates for the Complete Test Dataset.

	Word Error Rate	Sentence Error Rate	Command Error Rate
Original grammar	28.0%	64.4%	43.1%
New grammar	20.4%	46.3%	28.1%
Oracle grammar	1.9%	7.5%	4.8%

Table 7-2: Error Rates for the In-Domain Dataset.

As expected the tests show that making a more flexible grammar, with an increased vocabulary, improves the recognition performance. In both test sets, there was a significant decrease in Command Error Rate when the *New Grammar* was used. One hypothesis before the results were determined was that adding too many similar sounding phrases and words might lead to an increase in misrecognition rates, but the data shows that a broader coverage of phrases lowers each type of error rate.

Unfortunately, even when users only said in-domain phrases, the application still got the command wrong over a quarter of the time (28.1 percent error rate) – even with the improved grammar. Some of these errors result from similar sounding words like (“on” and “off”) being misrecognized, but as we can see from the results of using the Oracle grammar, this accounts for only about 4-5% of the errors. The other errors are mostly due to test phrases still not including

the exact same sentence structure provided in the grammar, and the recognizer trying to force a fit.

8 Testing MapDroid

8.1 Releasing MapDroid on the Android Market

In order to test the application, a version of MapDroid was released for free on the Android Market¹⁰. In its first 6 days, there were over 1150 downloads. Over 10,000 interactions¹¹ were recorded by the SLS server. This beta test of MapDroid produced mixed results.

Of the 10,000 interactions, approximately 600 were voice commands spoken in English. Of these 600 English spoken commands, only 255 were actually in domain. The other 350+ commands were almost entirely related to finding a distinct location, such as “Portland, Oregon.”

The purpose of MapDroid appears to be unclear to new users. Many users who download it try to issue commands similar to Google Map commands, such as, “London.” Users expect the recognizer to understand the location name, and center the map on the provided location. As discussed in *Restriction of Grammar*, pg. 119, MapDroid does not provide this functionality. It is apparent from this initial test that MapDroid needs to be able to handle these free-form location requests (see *Recognition for Locations*, pg. 136).

The application is being downloaded and used by people outside the U.S. Since the application only uses an English grammar, it can only recognize commands spoken in English. An estimated 15% of the recorded commands were spoken in a language other than English. Also, all the text used in the application is in English, so the application is essentially unusable for non-English speaking users.

¹⁰ <http://www.android.com/market/>

¹¹ An interaction is defined as a recording of a .wav file by the server. A recording is started every time the user clicks the talk button or presses down on the application’s home screen.

One notable success is the “What can I say?” help page (see *Help for Voice Commands*, pg. 112). Of the 255 valid voice commands, 23 were “what can I say?” After issuing this command, users typically experimented with phrases provided in the voice command help page.

It should also be noted that while many of the voice commands have been out of domain, there is evidence that users have started to make landmarks. In the grammar sent to the server, we can check the list of landmark names in the grammar. Many users have created 1-3 landmarks, the most common being “home”. This is encouraging, since it indicates that users who have experimented with the application are starting to utilize the application’s true functionality.

Of the 1150+ downloads, there are 809 active installs, which means that 70% of users have decided to keep MapDroid on their phone. We consider this a positive success rate.

8.2 Voice Command Results

To evaluate how well the application’s recognizer is performing on in-domain commands, we took the 255 in domain commands and tested the recognizer on them. The results from this test are provided in Table 8-1 under the MapDroid grammar row. To test the confusability of the commands themselves, we also created an oracle grammar that only included the actual phrases that were used in the test data.

Grammar Used	Word Error Rate	Sentence Error Rate
MapDroid	79.9%	36.1%
Oracle	93.5%	13.3%

Table 8-1: Results of running the recognizer on sample user commands, using both the normal MapDroid grammar, and an Oracle grammar comprised of only the test phrases.

The results show that 36% of the commands were recognized correctly. However, since a simple substitution such as “the” is counted as misrecognition, then the actual number of misrecognized commands was probably closer to only 30%. This means that the application was correctly executing about 70% of spoken commands.

Part of the reason the recognizer struggled was that many of the commands had significant background noise. Some of commands had traffic noise, radio music, TV audio, and even a crying baby included in the background. This is illustrated best by the fact that the oracle grammar produced a 13.3% sentence error rate. This means that 1/3 of the commands couldn't be recognized by the simplest, all inclusive grammar possible, which leads us to believe that these commands probably had poor audio quality.

Also, the recognizer had a noticeably hard time recognizing 2-3 word commands, and fared much better at deciphering commands that consisted of 4 or more words. For the 255 in-domain commands, there were 692 total words, which means the average command length was 2.7 words. This is the result of novice users issuing simple commands such as “zoom in” and “My Location”.

9 Summary and Future Work

9.1 Summary

This paper presents the design and development of a multimodal map-based application for a mobile device. It explains how the device's many hardware features can be used to generate informative content for geographical locations. We have also shown how this content can be dynamically integrated into the system's vocabulary so that users can reference landmarks by the information they have provided. The preliminary findings from the release of MapDroid on the Android Market suggests that improvements still need to be made to the system, but that users still find the application interesting and useful.

9.2 Future Work

Although the application is functional, and a trial version has been released on the Android Market, there are still a number of important features that could, and most likely should, be added to later versions of MapDroid.

9.2.1 Automatic Comment Transcription

Automatic comment transcription was one of the basic goals for MapDroid when it was being designed. The idea was that users could record spoken comments, and store the audio and transcription with a landmark. Text-to-speech, as we have mentioned in this thesis, can easily be performed by the Google Recognizer. However, the Google Recognizer does not let the application capture the audio that is input to the device. Consequently, MapDroid can only store

the audio file for a comment, and the user must manually write the transcription if they want to include it. Future versions of MapDroid should find a way to provide both forms of a comment.

9.2.2 Export Geo-Tagged Photos

MapDroid currently has the ability to export all the photos that are attached to a user's set of landmarks. These photos are all converted into JPEG images, and stored in a folder on the phone. This allows users to view the pictures outside of the application if they want to. One piece of information that is missing with each exported photo is where it was taken. Many websites currently allow users to share geo-tagged photos. These sites often group photographs by the geographical location that they were taken at, much like MapDroid does. This can be automatically done by extracting the coordinates of the photograph from its meta-data. The photographs exported by MapDroid, however, don't include this information in them.

It seems unreasonable that MapDroid contains both the photograph, and the information about where it was taken, but can't export the photograph with the location included in its meta-data. Manually writing meta-data is not a simple process. Consequently, this feature was not included. If a developer knew how to write out meta-data to a file, however, it would be very easy to provide this feature.

9.2.3 More Flexible Sharing Policy

One of the main goals of MapDroid is to allow users to share landmarks. Users are able to do this by defining the privacy settings for landmarks. Currently, MapDroid only supports two privacy settings: public or private. This means that no one can view a particular landmark, or

everyone can. This does not offer enough flexibility for users. It is quite possible that a user will generate content that they would like to share with coworkers, friends or family, but not with the general public.

One proposed solution would be to provide a mechanism to send landmark data to other users. Users could choose to send a landmark or album of landmarks to users selected from their contact list. The landmarks would be uploaded to a private table on the server. The people receiving the landmarks would get a message notifying them that a friend sent them landmarks, and they could then download the landmarks onto their phone, using the same mechanism that is used to download public landmarks. It would require a significant amount of time to create an entire sending system, so this feature was left out of the current version of MapDroid.

9.2.4 Improving Speech Recognition

It is obvious from the preliminary user tests that significant improvements need to be made to the application's recognition capabilities. One approach may be to keep the context-free grammar, and work on improving it like we did with the Mechanical Turk data. Since all the audio for users is stored, we can manually transcribe valid voice commands. These transcriptions can be processed and any valid phrases that couldn't be recognized can be integrated into the existing grammar.

Another approach may be to collect a large variety of possible interactions, using Mechanical Turk and data from test users, for each command available in MapDroid. These interactions can be used to train up a separate n-gram recognizer for each command. This would improve the recognizer's flexibility with respect to how users phrase their commands or requests, but it still needs to be determined whether it could actually decrease the command error rate.

Furthermore, there are over 100 distinct commands used by MapDroid. If we were to use the application's existing command structure, this approach would require 100 different n-grams. Generating enough data to sufficiently train these n-grams in a useful way would be a large task.

9.2.5 Recognition for Locations

The most common error that leads to misrecognition is when users try to reference locations that are not included in the application's grammar. It appears that users have become familiarized with map programs, such as Google Maps, that allow them to search for any known location. Consequently, this same functionality is expected of MapDroid. For example, users will issue a command like, "Find Staten Island", even though they have not created a landmark at Staten Island. Since MapDroid has no content associated with Staten Island, no reference to Staten Island is included in the application's grammar, which means this command is unrecognizable. This type of error leads to almost half the voice command errors, which makes it the single most important problem to address.

Since the SLS group does not use a generic recognizer, it appears that the Google Recognizer would have to be used in order to recognize addresses and location names. One quick fix would be to include a drop down search tool in the MapDroid home screen. This search tool would use the Google Recognizer and allow users to search for locations. They could then use the regular MapDroid commands to add content to a location once it had been found. This could be confusing, however, since there would be two ways to issues voice commands on the home screen.

A more elegant solution would be to run both recognizers using the same audio input. If the SLS recognizer could not recognize a valid command from the input, the transcription

produced by the Google Recognizer could be sent to a geocoder. If the geocoder produced a valid address, this location could be displayed on the map. This solution is preferable, since it wouldn't require two separate search activities. Because the Google Recognizer is a separate activity, it can't be run at the same time as MapDroid, which means there is currently no way to run both recognizers at the same time.

9.2.6 How to Correct Users

One recurring error, related to voice commands, is that novice users often don't know what they can say to the system. When they say something out of domain, the application will simply show them a message that informs them that there were no recognizable results returned by the recognizer. Not knowing any better, users will typically repeat the same mistake 2-4 times.

It would be helpful if future versions of MapDroid could predict when users are repeating the same phrase, and automatically correct them by opening the "What can I say?" help page (see *Help for Voice Commands*, pg. 112). The most straightforward solution would be to keep track of the number of consecutive unrecognizable commands, and if that number exceeded a threshold in a specified amount of time, then the command help page could automatically be opened. For example, if there were three straight unrecognizable commands in less than 20 seconds, it can be assumed that the user is struggling with the voice commands.

This feature was left out of the current version because automatically showing a help menu could be more annoying than helpful. Naively using unrecognizable commands as a measure of confusion is error prone, since unrecognizable audio could have more to do with noisy background conditions than with validity of the commands. After our initial user tests,

however, it is apparent that some form of automatic help correction should be included in the application.

9.2.7 Offline Speech Capabilities

One problem with using external recognizers is that speech recognition is only available if the phone has internet connectivity. While most options on MapDroid also can be performed without speech, the inability to use speech commands causes a significant reduction in usability.

To address this issue, MapDroid experimented with the use of Pocket Summit. Pocket Summit is a speech recognizer that can be run locally on the phone. It was created by the SLS group, and has been adapted to run on the Android phone. The same grammar that is passed to the external recognizer can be used by Pocket Summit. Pocket Summit also allows the application to use Dynamic classes, which allows the application to alter groups of items efficiently, because it doesn't have to rebuild the entire recognition lattice. This is especially useful when the application only needs to update the grammar's list of landmark names and cities.

The main reason Pocket Summit was not included in the initial version of MapDroid is that it takes too long to build the recognition network. For the grammar used by MapDroid, it takes Pocket Summit about 15 seconds before it is ready to process speech, which is not an acceptable latency for the application. One solution to this problem is to build the network for the template grammar in advance. This recognition could be stored in a zipped file. When MapDroid is installed on the phone, this file would be included in the MapDroid package, and saved on the phone. It could then be unzipped and immediately used by the Pocket Summit recognizer. This would save the phone from having to rebuild the grammar each time.

This solution was not explored because Pocket Summit was not deemed an essential component. Since the map that the application displays loses visibility once the phone has lost internet service, this causes the application to be mostly unusable. It was not apparent that providing offline voice recognition would make the system significantly more usable.

10 References

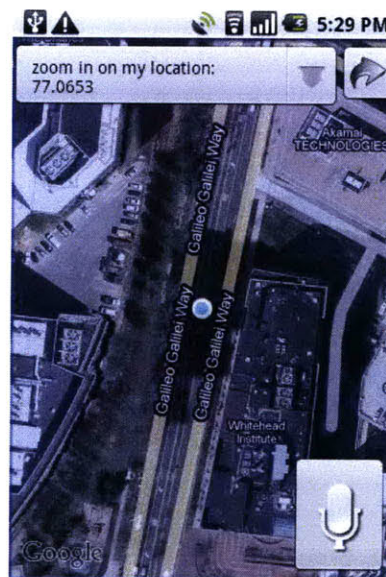
- [1] Gruenstein, A., Ian McGraw, and Ibrahim Badr. The WAMI Toolkit for Developing, Deploying, and Evaluating Web-Accessible Multimodal Interfaces. In ICMI (October 2008). <http://groups.csail.mit.edu/sls/publications/2008/Gruenstein_ICMI.pdf>
- [2] Gruenstein, A., Reimer Bryan, et al. City Browser: Developing a Conversational Automotive HMI. In CHI (2009), 4291-4296.
- [3] Gruenstein, A. and Seneff, S. Releasing a Multimodal Dialogue System into the Wild: User Support Mechanisms. In *Proc. SIGdial* (2007), 111-119.
- [4] Liu, Sean. Multimodal Speech Interfaces for Map-based Applications. Thesis (2010).

11 Appendix A

Turk Task 2.0 Results

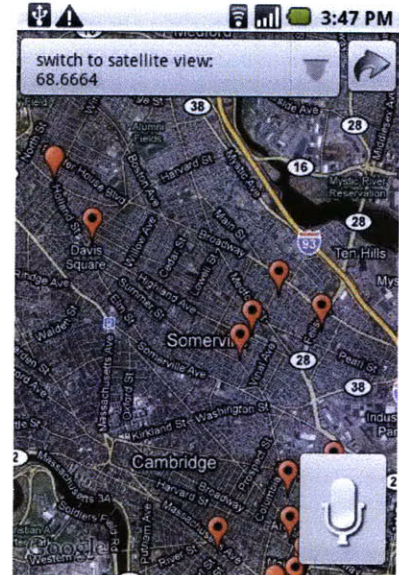
Zoom in on my location

bookmark this place, 1
close in on current position, 1
closer look, 3
closer view, 2
current location zoom, 1
different location, 1
display current location, 1
enlarge my location, 1
exit, 1
find me and zoom, 1
go in closer, 1
goto lat (degrees) long (degrees), 1
increase mao, 1
locate me, 2
look closer, 3
more zoom, 1
search google for akamai technologies, 1
show blue map marker, 1
show me my location, 1
show me where i am, 1
traffic patterns of this area, 1
update current location info, 1
what is the distance from (here) to (there), 1
what street am i on, 1
where am i, 2
xloser look, 1
zoom, 4
zoom 770653, 1
zoom closer, 1
zoom here, 2
zoom in, 8
zoom in location, 4
zoom in on location, 1
zoom in on me, 1
zoom in on my location, 1
zoom in on my position, 1
zoom in where i am now, 1
zoom on location, 1
zoom to me, 1



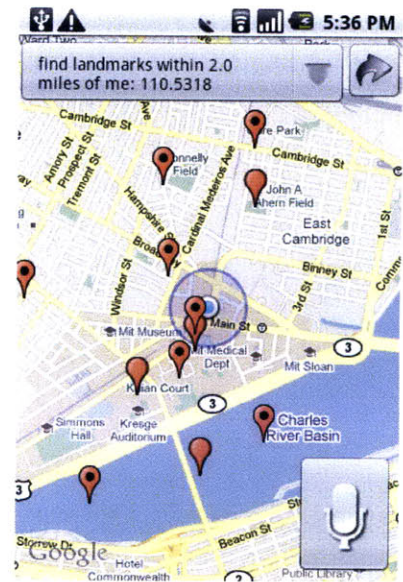
Switch to satellite view

- bird's eye, 1
- center on somerville, 1
- change map mode, 1
- change to satellite, 1
- display favorites, 1
- display satellite view, 1
- display saved locations, 1
- find points of interest in cambridge, 1
- focus out full, 1
- get directions from (here) to (there), 1
- go to satellite view, 1
- image satellite, 1
- look from satellite, 1
- look from satellite view, 1
- look from the sky, 1
- name the river, 1
- normal view, 6
- satellite, 4
- satellite look, 1
- satellite view, 5
- satellite view of (city state), 1
- show (laundromats) around somerville, 1
- show aerial view, 1
- show all locations in satellite view, 1
- show bookmarks, 1
- show favorites, 1
- show me satellite imagery, 1
- show must-see places in cambridge, 1
- show satellite, 1
- show satellite view, 2
- show sites to visit in cambridge, 1
- show tourist hotspots in cambridge, 1
- sky view, 5
- space view, 1
- stellite view, 1
- switch to map view, 1
- switch to satellite view, 1
- view options ----> satellite, 1
- zoom in/out, 1



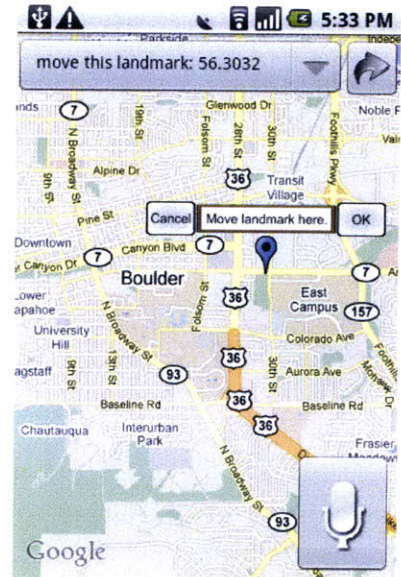
Find landmarks with 2 miles of me

close landmarks, 2
display all landmarks within 2 miles, 1
find all child care facilities with 5 miles of my home, 1
find all parks with play areas in a 2 mile radius, 1
find all pubs with in walking distance, 1
find all retail stores with in a 20 mile radius, 1
find close landmarks, 2
find landmarks, 3
find landmarks near by, 1
find landmarks with in a mile, 1
find landmarks within 2 miles, 1
find me landmarks, 2
find some landmarks, 1
in 2 miles turn right, 1
in 69 miles turn left, 1
landmarks, 1
landmarks close by, 1
landmarks close to me, 1
landmarks near by, 3
landmarks near me, 1
local attractions within 2 miles of me, 1
locate close landmarks, 1
locate landmarks, 2
look for landmarks, 2
main street, 1
mile away landmarks, 1
near by landmarks, 1
poi 2 miles to me, 1
search landmarks, 1
show landmarks, 1
show landmarks within 2 miles of me, 1
show me landmarks, 1
tell me landmarks within 2 miles of my location, 1
then turn left, 1
turn left again, 1
view landmarks within 20 miles of me, 1
what landmarks are close, 1
what landmarks are near by, 1
what's within two miles of me, 1



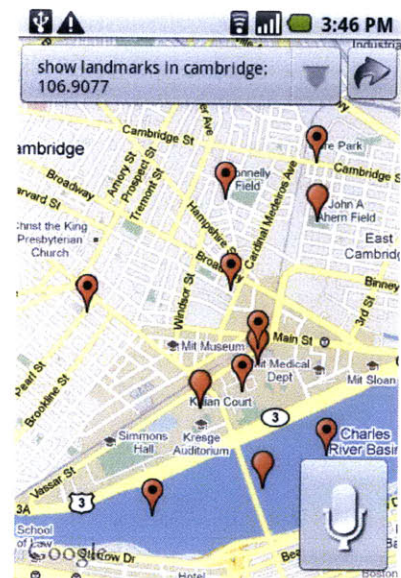
Move this landmark

- adjust landmark coordinates, 1
- alter landmark position, 1
- cancel, 1
- different location, 1
- move landmark, 7
- move location, 1
- move my landmark, 1
- move this landmark, 6
- ok, 1
- push landmark to here, 1
- relocate landmark, 5
- relocate landmarks, 1
- relocate this landmark, 4
- relocate this landmark, 1
- switch location of landmark, 2
- switch the location of landmark, 1
- zoom out, 1



Show landmarks in Cambridge

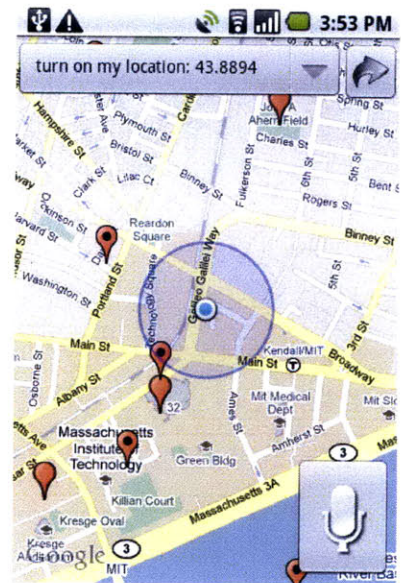
- all possible destinations, 1
- cambridge landmarks, 3
- cambridge landmarks near by, 1
- cambridge's landmarks, 1
- find landmarks in cambridge, 1
- find landmarks near cambridge, 1
- find local park, 1
- go to christ the king presbyterian church, 1
- go to mit medical department, 1
- go to simmons hall bypassing vassar street, 1
- landmarks, 1
- landmarks for cambridge, 1
- landmarks in cambridge, 1
- landmarks near cambridge, 2
- landmarks on cambridge, 1
- locate landmarks, 1
- poi in cambridge, 1
- show landmarks here, 1
- show alternate route, 1
- show closest saved locations, 1
- show distance from home, 1
- show landmarks, 2
- show landmarks in cambridge, 4
- show landmarks near by cambridge, 1
- show landmarks near cambridge, 1
- show me some landmarks, 1
- show me stuff in cambridge, 1
- show nearby hotels, 1
- show nearby restaurants, 1



show nearest landmarks, 1
view cambridge's landmarks, 1
what is in the area, 1
what's in cambridge, 1

Turn on My Location

activate my location, 1
display my location, 1
find me, 5
find me on the map, 1
find my location, 4
gps -----> on, 1
locate me, 5
locate my location, 1
locate where i am, 1
location -----> on, 1
my location, 2
prompt my location, 1
show directions to, 1
show location addresses, 1
show my location, 2
start my location, 1
trigger my location, 1
turn on 3rd st, 1
turn on 5th st, 1
turn on albany st, 1
turn on main st, 1
turn on my location, 2
what is my location, 1
where am i, 5
zoom in, 1
zoom out, 1



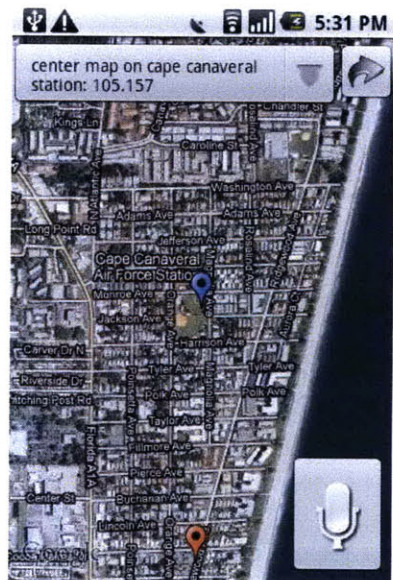
Zoom in here

- airport, 1
- closer look, 2
- closer view, 1
- display closest airport, 1
- go straight, 1
- look closer, 1
- make a u turn, 1
- plane's runway, 1
- runway of airport, 1
- show best resolution airport map, 1
- show my airport, 1
- show terminal d, 1
- tarmac, 1
- turn left, 1
- turn right, 1
- use earth view, 1
- zoom, 2
- zoom in, 2
- zoom in location, 1
- zoom to jfk airport, 1
- zoom to nearest airport, 1



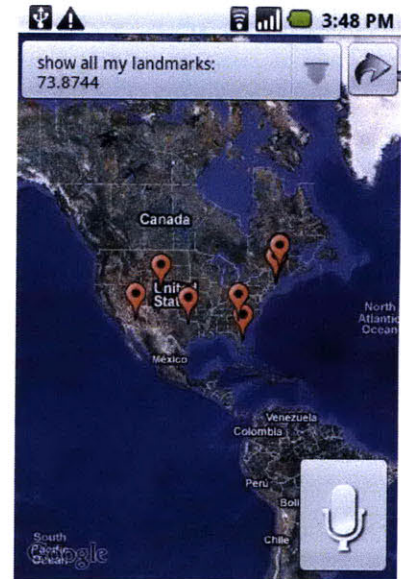
Center map on Cape Canaveral Station

- center, 2
- center it, 1
- center map, 6
- center map on overall destination cape canaveral, 1
- center view, 3
- display cape canaveral, 1
- display cape canaveral station, 1
- even out map, 1
- find cape canaveral and center, 1
- find cape canaveral station, 1
- go to cape canaveral, 1
- middle look, 1
- middle view, 2
- move map to final location cape canaveral, 1
- point me to cape canaveral station, 1
- save a location to your favorite list possibly with directions, 1
- show cape canaveral, 1
- show cape canaveral and surrounding area, 1
- show cape canaveral station, 1
- tell it whom to send a map to (from your address book), 1
- tell the app where you want to go and get directions, 1
- zoom to cape canaveral, 1



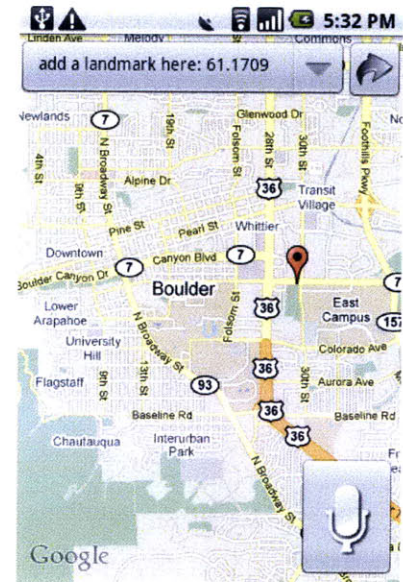
Show all my landmarks

all landmarks, 1
different location, 1
display all landmarks, 1
display all locations, 1
exit, 1
find my landmarks, 1
hot spots, 1
land marks, 1
landmarks, 2
list all my landmarks, 1
list my favorite locations, 1
list my locations, 1
list my markers, 1
list my vacation spots, 1
locate my landmarks, 1
my landmarks, 1
my vacation spots, 1
number one, 1
pinpoint all my landmarks on map, 1
see favorite spots, 1
show all locations for roadtrip, 1
show all my poi, 1
show all user created landmarks united states, 1
show everything i've marked, 1
show favorites, 1
show landmarks, 3
show locations of my relatives, 1
show marks, 1
show me all my landmark locations, 1
show my landmarks, 3
show my places, 1
show my points of interest, 1
show my saved landmarks, 2
show points of interest, 1
show saved landmarks, 1
show sites to visit for my vacation, 1
view landmarks, 2
what are my landmarks, 1
where are my landmarks, 1
zoom in, 1



Add a landmark here

add landmark, 3
add landmark at this location, 1
add landmark here, 1
add landmark near here, 1
add landmarks, 2
add mark at 611709, 1
add this landmark, 1
add this place, 1
add to landmarks, 4
add to my landmarks, 3
close landmarks, 1
close landmarks please, 1
find a close landmark, 1
find landmarks, 1
landmark add, 1
landmarks, 1
landmarks close, 1
landmarks near by, 1
mark at 611709, 1
near by landmarks, 1
pin at 611709, 1
pin point at 611709, 1
save landmark, 1
save to landmarks, 1
turn down boulder, 1
turn down baseline rd, 1
turn down glenwood dr, 1
turn down n broadway st, 1



12 Appendix B

Grammar

```
#JSGF V1.0;
grammar MapGrammarLandmarks;

public <top> = <command>;

<command> = zoom in {[r=6]}
| zoom out {[r=7]}
| zoom in (<here> | on <this_place>) {[r=1]}
| <show> street view ([<for_of>] <here> | [<at>] <this_place>) {[r=2]}
| <show> <all> [of] [my] <landmarks> {[r=3]}
| <take> (<a_photograph> | [a] [<new>] <photograph>) {[r=4]}
| <add> (<a_photograph> | [a] [<new>] <photograph>) {[r=4]}
| <record> (<a_comment> | [a] [<new>] <comment>) {[r=5]}
| (<add> | <attach>) (<a_comment> | [a] [<new>] <comment>) {[r=5]}
| <add> ((<a_landmark> | [a] point) | [a] [<new>] (<landmark> | point)) {[r=60]}
| (<add> | <put>) ((<a_landmark> | [a] point) | [a] [<new>] (<landmark> | point)) (<here> | <at> <this_place>) {[r=61]}
| (<add> | <put>) ((<a_landmark> | [a] point) | [a] [<new>] (<landmark> | point)) <at> <my_location> {[r=73]}
| <add> (<here> | <this_place>) (to <landmarks> | as a <landmark>) {[r=61]}
| <add> <my_location> (to <landmarks> | as a <landmark>) {[r=73]}
| (remember | bookmark) <my_location> {[r=73]}
| (remember | bookmark) (<here> | <this_place>) {[r=61]}
| <show> [the] (<information> | address) <for_of> [<this>] <landmark> {[r=62]}
| <show> [the] (<information> | address) <for_of> <landmark_names_grammar> {[r=301]}
| <show> [<this>] <landmarks> (<information> | address) {[r=62]}
| <show> <landmark_names_grammar> (<information> | address) {[r=301]}
| <show> <photographs> (for | of) [<this>] <landmark> {[r=63]}
| <show> <photographs> (for | of) <landmark_names_grammar> {[r=302]}
| <show> [<this>] <landmarks> <photographs> {[r=63]}
| <show> <landmark_names_grammar> <photographs> {[r=302]}
| <show> <comments> for [<this>] <landmark> {[r=64]}
| <show> <comments> for <landmark_names_grammar> {[r=303]}
| <show> [<this>] <landmarks> <comments> {[r=64]}
| <show> <landmark_names_grammar> <comments> {[r=303]}
| <edit> [<this>] <landmark> {[r=65]}
| <edit> <landmark_names_grammar> {[r=304]}
| <show> <options> <for_of> [<this>] <landmark> {[r=65]}
| <show> <options> <for_of> <landmark_names_grammar> {[r=304]}
| <show> [<this>] <landmarks> <options> {[r=65]}
| <show> <landmark_names_grammar> <options> {[r=304]}
| (<add> | <attach>) (<a_photograph> | [a] [<new>] <photograph>) (to | <for_of>) [<this>] <landmark> {[r=66]}
| (<add> | <attach>) (<a_photograph> | [a] [<new>] <photograph>) (to | <for_of>) <landmark_names_grammar> {[r=305]}
| comment on [<this>] <landmark> {[r=67]}
| comment on <landmark_names_grammar> {[r=306]}
| (<add> | <attach>) (<a_comment> | [a] [<new>] <comment>) (to | <for_of>) [<this>] <landmark> {[r=67]}
| (<add> | <attach>) (<a_comment> | [a] [<new>] <comment>) (to | <for_of>) <landmark_names_grammar> {[r=306]}
| <edit> ([<this>] <landmarks> | <landmark>) address {[r=68]}
```

```

|<edit> <landmark_names_grammar> address {[r=307]}
|<edit> address for <landmark_names_grammar>{[r=307]}
|<edit> ([<this>] <landmarks> | <landmark>) name {[r=74]}
|<edit> <landmark_names_grammar> name {[r=308]}
|rename [<this>] <landmark> {[r=74]}
|rename <landmark_names_grammar> {[r=308]}
|<edit> ([<this>] <landmarks> | <landmark>) description {[r=74]}
|<edit> <landmark_names_grammar> description {[r=309]}
|<edit> ([<this>] <landmarks> | <landmark>) <position> {[r=75]}
|<edit> <landmark_names_grammar> <position> {[r=310]}
|(<move> | <switch> [the] <position> <for_of>) ([<this>] <landmark>) {[r=75]}
|(<move> | <switch> [the] <position> <for_of>) <landmark_names_grammar> {[r=310]}
|<move> ([<this>] <landmark>) (<here> | <to> <this_place>) {[r=70]}
|<move> <landmark_names_grammar> (<here> | <to> <this_place>) {[r=311]}
|zoom in (<to> | on) [<this>] <landmark> {[r=71]}
|zoom in (<to> | on) <landmark_names_grammar> {[r=312]}
|<show> street view <for_of> [<this>] <landmark> {[r=72]}
|<show> street view <for_of> <landmark_names_grammar> {[r=313]}
|(<center> map on | <show>) [<this>] <landmark> {[r=76]}
|<center> map on <landmark_names_grammar> {[r=314]}
|<turn_on> <my_location> {[r=40]}
|turn my location on {[r=40]}
|<turn_off> <my_location> {[r=41]}
|turn my location off {[r=41]}
|zoom in (<to> | on) <my_location> {[r=42]}
|<show> street view <for_of> <my_location> {[r=43]}
|<center> map on <my_location> {[r=44]}
|(<find> | go to) (<my_location> | where i am) {[r=40]}
|where am i {[r=40]}
|<show> <my_location> <options> {[r=45]}
|<show> <options> for <my_location> {[r=45]}
|<add> [an] album {[r=80]}
|manage [my] albums {[r=81]}
|(open | close) an album {[r=81]}
|(<show> | <switch> [map [mode]|view] to) (satellite | aerial) [map [mode]|view] {[r=21]}
|(<show> | <switch> [map [mode]|view] to) (street map | map) [mode|view] {[r=20]}
|(<show> | <turn_on>) street view [overlay] {[r=24]}
|(<turn_off> | <clear>) street view [overlay] {[r=25]}
|(<show> | <turn_on>) traffic [overlay | map] {[r=22]}
|(<turn_off> | <clear>) traffic [overlay | map] {[r=23]}

|(<find>) [<directions_types>] <directions> {[r=130]}
|<find> [<directions_types>] <directions> ((from | for) <here> | <starting_at> <this_place>) {[r=131]}
|<find> [<directions_types>] <directions> (from | for) <starting_at> <my_location> {[r=133]}
|<find> [<directions_types>] <directions> (from | for) <starting_at> [<this>] <landmark> {[r=132]}
|<find> [<directions_types>] <directions> (from | for) <starting_at> <landmark_names_grammar> {[r=320]}
|<find> [<directions_types>] <directions> (to <here> | <ending_at> <this_place>) {[r=134]}
|<find> [<directions_types>] <directions> (to | <ending_at>) <my_location> {[r=136]}
|<find> [<directions_types>] <directions> (to | <ending_at>) [<this>] <landmark> {[r=135]}
|<find> [<directions_types>] <directions> (to | <ending_at>) <landmark_names_grammar>{[r=321]}

|<find> [<directions_types>] <directions> (from <here> | <starting_at> <this_place>)(to | [and] ending at) [<this>]
<landmark> {[r=120]}
|<find> [<directions_types>] <directions> (from <here> | <starting_at> <this_place>)(to | [and] ending at)
<my_location> {[r=121]}
|<find> [<directions_types>] <directions> (from | <starting_at>) [<this>] <landmark> (to | [and] ending) (<here> |

```

<at> <this_place> { [r=122]}
 | <find> [<directions_types>] <directions> (from | <starting_at>) [<this>] <landmark> (to | [and] ending at)
 <my_location> { [r=123]}
 | <find> [<directions_types>] <directions> (from | <starting_at>) <my_location> (to | [and] ending at) [<this>]
 <landmark> { [r=125]}
 | <find> [<directions_types>] <directions> (from | <starting_at>) <my_location> (to | [and] ending) (<here> | <at>
 <this_place>) { [r=124]}

| <find> [<directions_types>] <directions> (from <here> | <starting_at> <this_place>)(to | [and] ending at)
 <landmark_names_grammar> { [r=322]}
 | <find> [<directions_types>] <directions> (from | <starting_at>) <my_location> (to | [and] ending at)
 <landmark_names_grammar> { [r=323]}
 | <find> [<directions_types>] <directions> (from | <starting_at>) [<this>] <landmark> (to | [and] ending at)
 <landmark_names_grammar> { [r=324]}
 | <find> [<directions_types>] <directions> (from | <starting_at>) <landmark_names_grammar> (to | [and] ending at)
 [<this>] <landmark> { [r=327]}
 | <find> [<directions_types>] <directions> (from | <starting_at>) <landmark_names_grammar> (to | [and] ending at)
 <my_location> { [r=326]}
 | <find> [<directions_types>] <directions> (from | <starting_at>) <landmark_names_grammar> (to | [and] ending)
 (<here> | <at> <this_place>) { [r=325]}
 | <find> [<directions_types>] <directions> (from | <starting_at>) <landmark_names_grammar> (to | [and] ending at)
 <landmark_names_grammar> { [r=401]}

| ((how far [away] am i (from | to)) | (<dist_start> [the] distance (from | to))) (<here> | <this_place>) { [r=111]}
 | ((how far [away] am i (from | to)) | (<dist_start> [the] distance (from | to))) [<this>] <landmark> { [r=112]}
 | ((how far [away] am i (from | to)) | (<dist_start> [the] distance (from | to))) <landmark_names_grammar> { [r=341]}
 | <dist_start> [the] distance from <my_location> to (<here> | <this_place>) { [r=111]}
 | <dist_start> [the] distance between <my_location> and (<here> | <this_place>) { [r=111]}
 | how far [away] is [it (from | to)] <my_location> (from | to) (<here> | <this_place>) { [r=111]}
 | <dist_start> [the] distance from <my_location> to [<this>] <landmark> { [r=112]}
 | <dist_start> [the] distance between <my_location> and [<this>] <landmark> { [r=112]}
 | how far [away] is [it (from | to)] <my_location> (to | from) [<this>] <landmark> { [r=112]}
 | <dist_start> [the] distance from <my_location> to <landmark_names_grammar> { [r=341]}
 | <dist_start> [the] distance between <my_location> and <landmark_names_grammar> { [r=341]}
 | how far [away] is [it (from | to)] <my_location> (to | from) <landmark_names_grammar> { [r=341]}
 | <dist_start> [the] distance from (<here> | <this_place>) to <my_location> { [r=111]}
 | <dist_start> [the] distance between (<here> | <this_place>) and <my_location> { [r=111]}
 | how far [away] is [it (from | to)] (<here> | <this_place>) (to | from) <my_location> { [r=111]}
 | <dist_start> [the] distance from (<here> | <this_place>) to [<this>] <landmark> { [r=110]}
 | <dist_start> [the] distance between (<here> | <this_place>) and [<this>] <landmark> { [r=110]}
 | how far [away] is [it (from | to)] (<here> | <this_place>) (to | from) [<this>] <landmark> { [r=110]}
 | <dist_start> [the] distance from (<here> | <this_place>) to <landmark_names_grammar> { [r=340]}
 | <dist_start> [the] distance between (<here> | <this_place>) and <landmark_names_grammar> { [r=340]}
 | how far [away] is [it (from | to)] (<here> | <this_place>) (to | from) <landmark_names_grammar> { [r=340]}
 | <dist_start> [the] distance from [<this>] <landmark> to <my_location> { [r=112]}
 | <dist_start> [the] distance between [<this>] <landmark> and <my_location> { [r=112]}
 | how far [away] is [it (from | to)] [<this>] <landmark> (to | from) <my_location> { [r=112]}
 | <dist_start> [the] distance from [<this>] <landmark> to (<here> | <this_place>) { [r=110]}
 | <dist_start> [the] distance between [<this>] <landmark> and (<here> | <this_place>) { [r=110]}
 | how far [away] is [it (from | to)] [<this>] <landmark> (to | from) (<here> | <this_place>) { [r=110]}
 | <dist_start> [the] distance from [<this>] <landmark> to <landmark_names_grammar> { [r=342]}
 | <dist_start> [the] distance between [<this>] <landmark> and <landmark_names_grammar> { [r=342]}
 | how far [away] is [it (from | to)] [<this>] <landmark> (to | from) <landmark_names_grammar> { [r=342]}
 | <dist_start> [the] distance from <landmark_names_grammar> to <my_location> { [r=341]}

```

|<dist_start> [the] distance between <landmark_names_grammar> and <my_location> {[r=341]}
|how far [away] is [it (from | to)] <landmark_names_grammar> (to | from) <my_location> {[r=341]}
|<dist_start> [the] distance from <landmark_names_grammar> to (<here> | <this_place>) {[r=340]}
|<dist_start> [the] distance between <landmark_names_grammar> and (<here> | <this_place>) {[r=340]}
|how far [away] is [it (from | to)] <landmark_names_grammar> (to | from) (<here> | <this_place>) {[r=340]}
|<dist_start> [the] distance from <landmark_names_grammar> to [<this>] <landmark> {[r=342]}
|<dist_start> [the] distance between <landmark_names_grammar>and [<this>] <landmark> {[r=342]}
|how far [away] is [it (from | to)] <landmark_names_grammar> (to | from) [<this>] <landmark> {[r=342]}
|<dist_start> [the] distance from <landmark_names_grammar> to <landmark_names_grammar> {[r=402]}
|<dist_start> [the] distance between <landmark_names_grammar> and <landmark_names_grammar> {[r=402]}
|how far [away] is [it (from | to)] <landmark_names_grammar> (to | from) <landmark_names_grammar> {[r=402]}
|undo [last] [voice] command {[r=140]}
|redo (last undo | [last] [voice] command) {[r=141]}

```

```

|download <a_landmark> {[r=180]}
|<find> <a_landmark> {[r=160]}
|<find> <a_landmark> <using> <name> {[r=161]}
|<find> <a_landmark> <using> (<position> | proximity) {[r=162]}
|<find> <a_landmark> <using> address {[r=163]}
|<find> <a_landmark> <using> <contacts> {[r=164]}

```

```

|(<find> | (go | (take | bring) me) to)<landmark_names_grammar> {[r=314]}
|<find> [all] [the] (<plur_tags> | [<tags>] <landmarks>) {[r=507]}
|<find> [all] [the] (<plur_tags> | <landmarks> | <tags> <landmarks> | <landmarks> <named>
<landmark_names_grammar> [[located] <in> ((<city_grammar> | <state_grammar> | <city_grammar>
[<state_grammar>))] {[r=507]}
|<find> [all] [the] [<near_by>] (<plur_tags> | [<tags>] <landmarks>) [<near_by>] {[r=601]}
|<find> [all] [the] (<plur_tags> | [<tags>] <landmarks>) (within (<search_number> miles | [a] [<search_number>]
mile [radius]) [of] | <near>) [<my_location>] {[r=601]}
|<find> [all] [the] (<plur_tags> | [<tags>] <landmarks>) (within (<search_number> miles | [a] [<search_number>]
mile [radius]) [of] | <near>) (<this_place> | <here>) {[r=602]}
|<find> [all] [the] (<plur_tags> | [<tags>] <landmarks>) (within (<search_number> miles | [a] [<search_number>]
mile [radius]) [of] | <near>) [<this>] <landmark> {[r=603]}
|<find> [all] [the] (<plur_tags> | [<tags>] <landmarks>) (within (<search_number> miles | [a] [<search_number>]
mile [radius]) [of] | <near>) <landmark_names_grammar> {[r=604]}
|<find> [all] [the] (<plur_tags> | [<tags>] <landmarks>) (on | along) this street {[r=610]}

```

```

|<clear> search (results | <landmarks>) {[r=165]}
|<clear> downloads {[r=166]}
|what can i say {[r=100]}
|help with voice commands {[r=100]}
|voice command help {[r=100]}
|what can i do {[r=101]}
|help for main menu {[r=101]}
|<close> [this] (application | app) {[r=102]};

```

```

<at>          = at | <to> | for;
<new>         = new;
<close>      = close | exit;
<in>         = in | within | <near>;
<this_place> = this (place | point | spot | <position> | address);

```


<here> = here;
 <near> = near | close to | around;
 <nearest> = nearest | closest;
 <near_by> = nearby | close by;
 <all> = all [of] | each of;
 <only> = only | just;
 <to> = to;
 <dist_start> = <show> | calculate | what is;
 <starting> = starting | beginning | from;
 <starting_at> = starting at | beginning at;
 <ending_at> = ending at;
 <go_to> = go to | <show>;
 <show> = show [me] | view | display | get;
 <find> = find | locate | <show> | search for | check for | look for | select | retrieve;
 <add> = add | create | make | leave;
 <my_location> = (([my] current) | my) <position> | me;
 <put> = put | place;
 <attach> = attach | leave;
 <take> = take | capture;
 <record> = record | capture;
 <edit> = edit | change | alter | adjust;
 <switch> = switch | change;
 <move> = move | relocate;
 <options> = options | commands;
 <turn_on> = turn on | enable | <show>;
 <turn_off> = turn off | disable | remove | dismiss | hide | get rid off;
 <center> = center | focus;
 <clear> = clear | remove;
 <using> = using | by;
 <for_of> = for | of;
 <information> = information | info | data | contents;
 <contacts> = contacts | username;
 <this> = this | selected | highlighted | current | that | indicated;
 <description> = description | tag;
 <name> = name | title;
 <named> = named | titled | called;
 <position> = position | location;
 <landmark> = landmark | placemark | marker | bookmark;
 <a_landmark> = [a] landmark | [a] placemark | [a] marker | [a] bookmark;
 <landmarks> = landmarks | placemarks | markers | bookmarks;
 <photograph> = photograph | photo | image | picture | pick | snap shot;
 <a_photograph> = [a] photograph | [a] photo | [an] image | [a] picture | [a] pick | [a] snap shot;
 <photographs> = photographs | photos | images | pictures | pics | snap shots;
 <comment> = comment | note | audio | note;
 <a_comment> = [a] comment | [a] note | [an] audio | [a] note;
 <comments> = comments | notes | audios | notes;
 <directions> = directions;

<digit> = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 ;
 <double_digit> = 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | (20 | 30 | 40 | 50 | 60 | 70 | 80 | 90) [1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9] ;
 <number> = <double_digit> | <digit> <digit> | <digit>;

<search_number> = {<sasrchxxx>}({<intxxx>}<number>{<xxxint>} | {<intxxx>}<number>{<xxxint>} point {<decxxx>}<number>{<xxxdec>} | point {<decxxx>}<number>{<xxxdec>}){<xxxsasrch>};

```
<directions_types_list> = driving  
| transit  
| walking  
| biking;
```

```
<directions_types> = {<dtxxx>}<directions_types_list>{<xxxdt>};
```

```
<tags_list> = airport  
| apartment  
| bank  
| bar  
| beach  
| bus (stop | station)  
| camp ground  
| cinema  
| coffee (shop | house)  
| college  
| convenience store  
| dormitory  
| gas station  
| golf course  
| hospital  
| hotel  
| library  
| [shopping] mall  
| market  
| motel  
| movie theater  
| museum  
| park  
| parking lot  
| pharmacy  
| police station  
| post office  
| restaurant  
| river  
| school  
| store  
| subway [station]  
| super market  
| t stop  
| theater  
| train station  
| university  
;
```

```
<tags> = {<tagxxx>}<tags_list>{<xxxtag>};
```

```
<plur_tags_list> = airports  
| apartments  
| banks  
| bars  
| beaches  
| bus (stops | stations)
```

| camp grounds
 | cinemas
 | coffee (shops | houses)
 | colleges
 | convenience stores
 | dormitories
 | gas stations
 | golf courses
 | hospitals
 | hotels
 | libraries
 | [shopping] malls
 | markets
 | motels
 | movie theaters
 | museums
 | parks
 | parking lots
 | pharmacies
 | police stations
 | post offices
 | restaurants
 | rivers
 | schools
 | stores
 | subways | (subway [stations])
 | super markets
 | t stops
 | theaters
 | train stations
 | universities;

<plur_tags> = {<plurtagxxx>}<plur_tags_list>{<xxxplurtag>};

<state_list> = alabama | alaska | arizona | arkansas | california | colorado | connecticut | delaware | florida | georgia | hawaii | idaho | illinois | Indiana | iowa | kansas | kentucky | louisiana | maine | maryland | massachusetts | michigan | minnesota | mississippi | missouri | montana | nebraska | nevada | new hampshire | new jersey | new mexico | new york | north carolina | north dakota | ohio | oklahoma | oregon | pennsylvania | rhode island | south carolina | south dakota | tennessee | texas | utah | vermont | virginia | washington | west virginia | wisconsin | wyoming;

<state_grammar> = {<\$STATE_LIST>}<state_list>{</\$STATE_LIST>};

<landmark_names_list> = camp
 | davis
 | fish market
 | football field
 | shaws
 | bridge
 | fenway park
 | beaver pond connection 3 4
 | goody gloves
 | 2nd beaver pond
 | apartment place
 | kendall t stop
 | cape canaveral station

| raytheon
| alewife
| big beaver pond
| next house
| flag football field
| russell field
| cowboy stadium
| my apartment
| justin and janet apartment
| simmons dormitory
| daves house
| skillings bridge
| stata center
| logan international airport
| m i t
| beaver pond
| highland bus stop
| student center
| simmons hall
| linden
| jims apartment
| home
| beaver pond 3
| beaver pond 4
| 85 bus stop
| gram and gramp
| foxwoods mgm
| skillings cabin
| janets house
| football
| amandas apartment
| party apartment
| tree farm;

```
<landmark_names_grammar> =  
{</$LANDMARK_NAME_LIST>}<landmark_names_list>{</$LANDMARK_NAME_LIST>;
```

```
<city_list> = strong  
| somerville  
| boston  
| cambridge  
| farmington  
| augusta  
| kingfield  
| cocoa beach cape canaveral  
| milliken  
| sun city  
| irving  
| abington  
| charlottesville  
| clearwater  
| st pete beach  
| arlington  
| miami;
```

```
<city_grammar> = {</$CITY_LIST>}<city_list>{</$CITY_LIST>;
```