Autonomous Tour Guide
Increment 1

Project group 6

Sindhu Reddy Golconda - 8
Simon Moeller - 15
Prudhvi Raj Mudunuri - 16
Sudhakar Reddy Peddinti - 21

University of Missouri - Kansas City
CS5542
February 19, 2016

Introduction

The project for group 6 is for a robot that can autonomously discover the layout of a new location, and then act as a guide for the human wearing the linked smart watch. This is a take on an autonomous search and rescue or reconnaissance drone. The idea is that the controller would wear the linked smartwatch, release the robot into a new building or location to search and map. After sufficient time the controller can ask the watch to take him or her to a particular location within the building. The robot will then find its way back to the user, and will lead the user to the desired location.

The robot is controlled by a smartphone. The smartphone sends a continuous stream of images to the backend analytics server. The backend analytics server compares these images to a database of known objects. As objects and locations are identified, their location is saved. After a sufficient area is mapped, with some objects and locations identified, the user's smartwatch is notified.

Once notified that the area is sufficiently mapped, the user will verbally ask the smartwatch to take the user to one of the identified objects or locations. The smartwatch uses a direct link to talk with the robot using wifi or bluetooth or some other useable link. Using location and signal strength information, the robot attempts to find the wearer of the smartwatch. Once the robot locates the wearer of the smartwatch, the robot audibly informs the user. The user then gives the verbal command to start, and the robot will lead the user to the desired object or location within the building.

An example would be to utilize the robot to assist a person with limited mobility to find the best route to a desired location within a new structure. The robot would be able to map the inside of the structure, and when instructed, lead the user to the desired location using a path that adheres to the user's particular restrictions.

Project Goal and Objectives

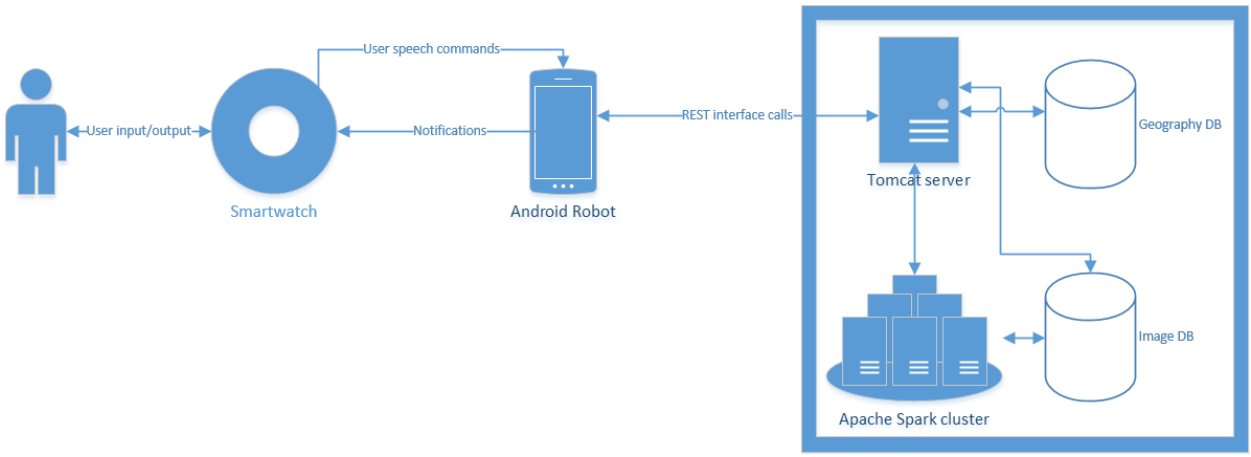
This project is a very large and complex undertaking, and as such the features of the project have been prioritized to favor primary learning goals for the team. Many teams have spent countless thousands of hours attempting to build robotic guides similar to what what is proposed here. Due to the level of difficulty in producing a fully functional autonomous tour guide robot, the desired features for this project have been stack ranked. The more critical learning objectives are ranked higher, and less critical learning objectives are ranked lower.

Project goals:

- Stack rank High
    - Speech to text through smartwatch, for sending commands to the smartphone controlled robot
    - REST communications interface between smartphone and backend analytics server
    - Capture and send images from camera to backend analytics server at beneficial intervals
    - Backend analytics server to process image identification against each image received
    - Backend tracking of location of identified images in relation to the robot as it moves
    - Provide updates to the smartwatch each time a new image is identified
    - Use a MapReduce based algorithm on Apache Spark for image detection
    - Image identification/recognition
- Stack rank Medium
    - Identify and track the direction and distance from the camera for each identified image
    - Self learning what images should look like, based on a user provided list of objects to expect
    - Be able to retrace steps to find a previously identified image
- Stack rank Low
    - Support the full range of robot movement (ranked low due to the known defects with the API)
    - Obstacle avoidance when the robot moves
    - High success rate of accurate image detection
    - Have the robot find the wearer of the smartwatch based on available methods
    - Identify and use shortest path between points when the robot either finds the smartwatch or finds a previously identified image

Design

The design of the autonomous tour guide is based on four primary components. These are the user's smartwatch, the Android powered robot, the backend Tomcat server, and the backend Apache Spark cluster. The user's smartwatch is the primary user interface device, providing a speech to text capability and user notifications. The Android robot is responsible for physically navigating about the area, and sending a regular stream of images with geolocation information to the backend systems. The Tomcat server is responsible for keeping a record of the geolocation information received from the robot, and provides the "brains" for calculating movement commands to direct the robot. The Apache Spark cluster is responsible for image calculations, including keypoint identification and similarity scoring between images.



Individual components that have already been worked are described below.

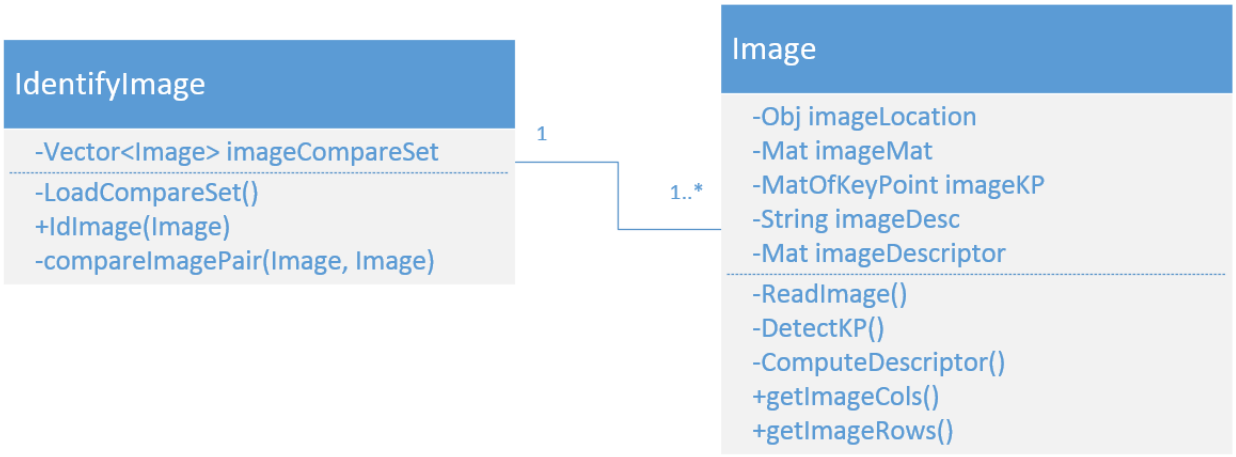
Image matching engine

Source code repository: https://github.com/smoeller1/BigData-Spring2016-TourGuide/tree/master/src/ImageId/src/ImageId

The image matching engine is written as Java classes utilizing libraries from the OpenCV project. The first class is the Image class, responsible for identifying and saving the key points for the image. The second class is the IdentifyImage class, responsible for comparing different Image's to each other.

Each individual image is saved as an Image object. Upon instantiation, the Image object reads in the image file using the appropriate method depending on the image source location. Next, the OpenCV FeatureDetector is used to identify the key feature points using the ORB method, though other methods can be utilized. Last, the OpenCV DescriptorExtractor extracts the descriptors of the image using the ORB method.

The IdentifyImage class automatically creates Image objects for each of the comparison images upon instantiation. The goal is to have the comparison images be self-learned based on a keyword list of images that should be learned. However, at this time the list of comparison images is hard coded into the LoadCompareSet method. When a new image needs to be identified, the calling program calls the IdImage method with the new image that needs to be identified. The IdImage method will compare this new image against the database of known comparison images to attempt to find a match. The image comparison is done using the OpenCV Imgproc MatchTemplate to attempt to identify subsections of the new image, when possible.



Speech to text

Source code repository: https://github.com/SindhuReddyG-sgdd7/SpeechToText

The speech to text functionality utilizes the microphone on the smartwatch to allow the user to speak commands to the Android robot. The speech to text allows for text based commands to be easily processed by the Android robot. Upon receiving the commands the android robot will either makes appropriate web service calls to the central server or respond back to the watch with the content it has onboard in the form of notifications. The communication link between the smartwatch and the robot is the critical aspect of the autonomous tour guide application.

REST Interface

Source code repository: https://github.com/smoeller1/BigData-Spring2016-TourGuide/tree/master/src/ImageId/src/ImageId

The REST interface uses JSON formatted data to communicate between the Android robot and the backend servers. This interface is used for sending the images captured by the Android robot, as well as receiving messages from the servers when items are identified within the images, so that this information can be relayed on to the smartwatch.

Camera controller

Source code repository: https://github.com/smoeller1/BigData-Spring2016-TourGuide/tree/master/src/ImageId/src/ImageId

The camera controller works on both a timer and a movement tracker. Each time the timer counts down, the camera controller will check to see if there has been movement since the last image was captured. If there has been movement, a new image is captured. If there has not been movement, then no image is captured and the timer is reset.

Statistical Analysis

The caputred images are sent to the central server, where the image analysis is performed. The images identified will be subjected for statistical analysis to infer insights from the obtained data. As part of image identification/recognition algorithm, multiple captured images are grouped together based on the content of the images and all the grouped images are given for the image comparator for identifying the correct object present within the image. For achieving this, statistical method of k-means is used to group the captured images. The working of the image grouping is tested based on assumptions which are expected to be completed for the final stages.

Sample Image data:

ImageName	Q1_saturation	Q2_saturation	Q3_saturation	Q4_saturation
Image0.png	0.3	0.5	0.6	0.2
Image1.png	0.1	0.6	0.2	0.5
Image2.png	1.4	0.0	1.3	1.4
Image3.png	1.1	2.0	0.4	2.6
Image4.png	3.4	2.1	2.0	0.9
Image5.png	4.2	4.4	1.0	3.1
Image6.png	3.1	0.4	4.1	4.3

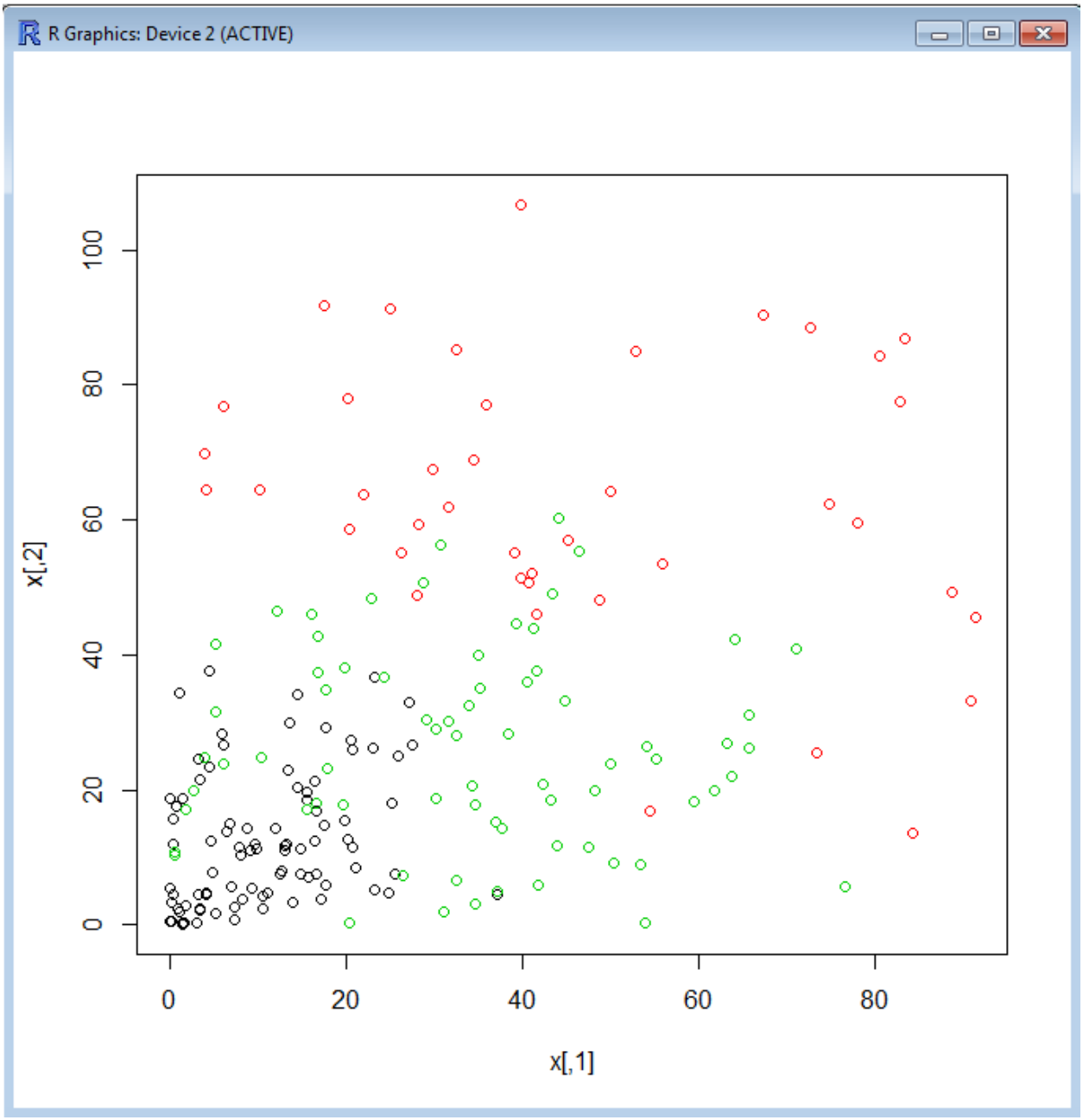


Figure 1: K-means clustering of images

Above screenshot show sample analysis of grouping the images using k-means method based on the features identified in the images.

External APIs

OpenCV - Image processing libraries. http://opencv.org

Project Plan

The project is divided into four iterations. The iterations are described in more detail below. In general, with each sprint each team member is focusing on finishing a single feature. At the start of each sprint the remaining features are divided among the team members, with all team members working on all parts of the system over the course of the project.

Iteration 1

The four features that were worked during iteration 1 were speech to text, REST communications interface, camera controls, and image identification. These four features are described in more detail above. The work this sprint was completed by the following team members:

Sindhu Golconda - Speech to text
Simon Moeller - Image identification
Prudhvi Mudunuri - Camera controller
Sudhakar Peddinti - REST communications interface

We attempted to use ZenHub to help track work, however ZenHub went down (or just stopped working) for all team members partway through the iteration. Since the team members already understood what needed to be done, we continued to work without ZenHub.

All team members communicated well during the iteration, and the team met in person multiple times over the course of the iteration. This good communication also helped the team to largely achieve the goals of the iteration. There are a few minor defects still being resolved in some of the code from this iteration, but it is anticipated that these debugging efforts will be complete by this weekend.

Bibliography

OpenCV libraries. http://opencv.org