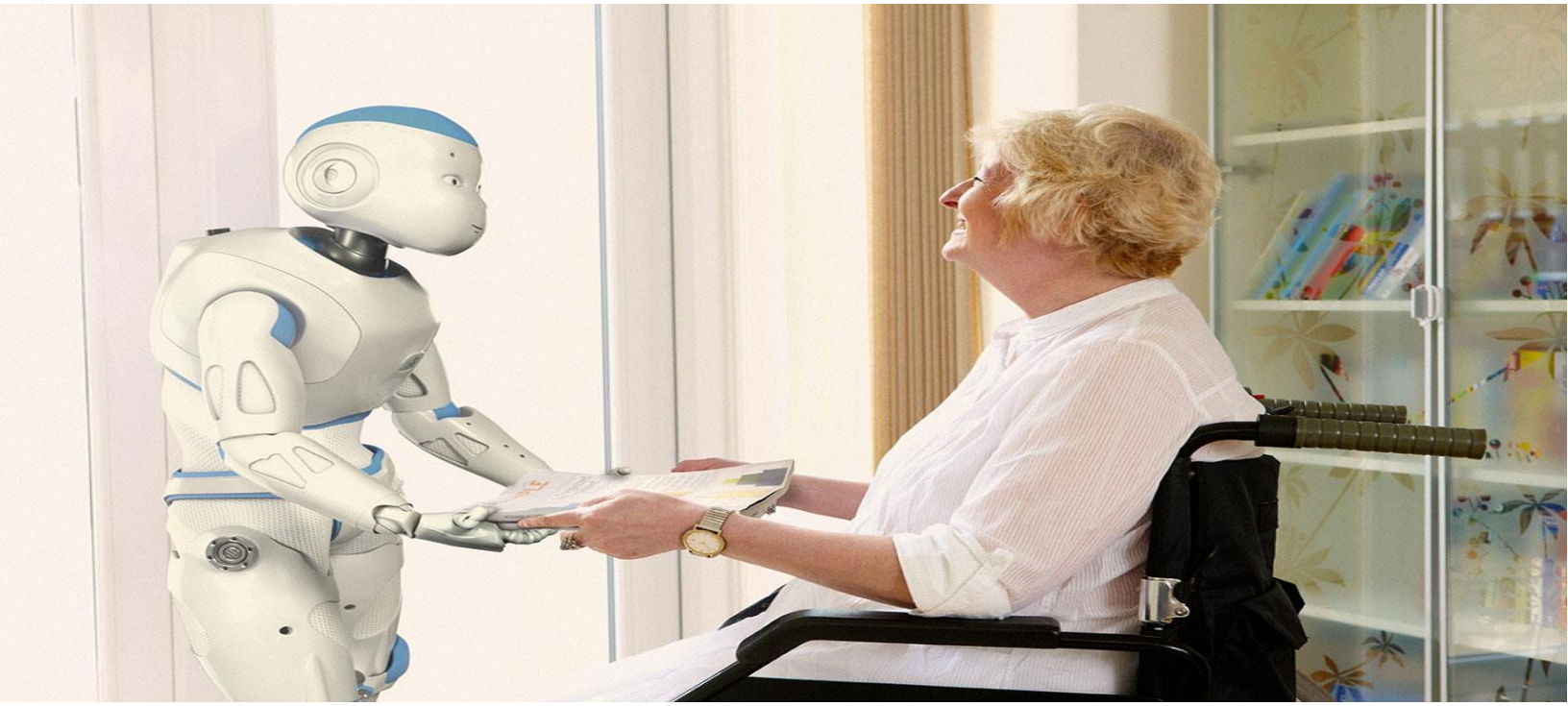


Robot-Assist

Big Data Analytics and Applications Project
Second Increment Document

Project Group #10

Sarika Bommavaram

Vaishnavi Aienampudi

Gouri Priya Vangavargu

Phanideep Salapaka

Increment #2

We can do many things with the help of robots. They can help people in many fields like doing some household work, helping in medical field, technological field etc.. The idea of this project comes from the question “Can a robot be a personal assistant to a disabled person?” We believe that humans can do this work but at some point of time or in some cases they become lazy or may not be able to do some work. So if we think of a robot, it can help humans to complete their work. This motivated us to make our robot as a personal assistant for a disabled person.

As a part of second increment we implemented the following features.

1. **Feature Extraction And Detection**

   As a part of our project, we have to make our robot to detect all the traffic signals. For traffic lights, we used openCV to detect colors and move according to them. But for remaining signals like STOP, GO, TURN LEFT and TURN RIGHT, we have to capture the complete image and then recognize it. So, for this we used feature detection.

   We implemented feature extraction with the help of SURF and OpenCV. SURF is one of the algorthims used for feature extraction. There are many such algorithms for feature detection like SIFT etc.. Feature detection is done with the help of FLANN and OpenCV. FLANN is an algorithm used to detect and match the extracted features.

2. **Face Recognition**

   We already implemented this feature for increment #1 but not completely. Here we first give the training data and then test data. What all the data for training is given, only that data is recognized. It can’t recognize new data. For increment #2, we integrated this with a greeting message. Whenever it recognizes a known face it will give a greeting message.

3. **Adding Reminders**

   We can add reminders by accessing our calendars and reminders from our iPhone. It will give an alarm sound for the reminder by displaying the text.

4. **Gesture Recognition**

   Our robot also senses gestures from the android controller. If the android controller moves to the left, the robot also turns to the left direction.

5. **Simple Question and Answers:**

   Our robot will respond to simple questions like what is your name ?, How are you ?,etc.. We implemented this feature with the help of NLP, and text to speech API.

6. **Twitter Streaming and MongoLab:**

   In this, we are collecting tweets based on a keyword and storing them in MongoLab. From the API of the data stored in Mongo Lab, we are retrieving the data using our mobile device.

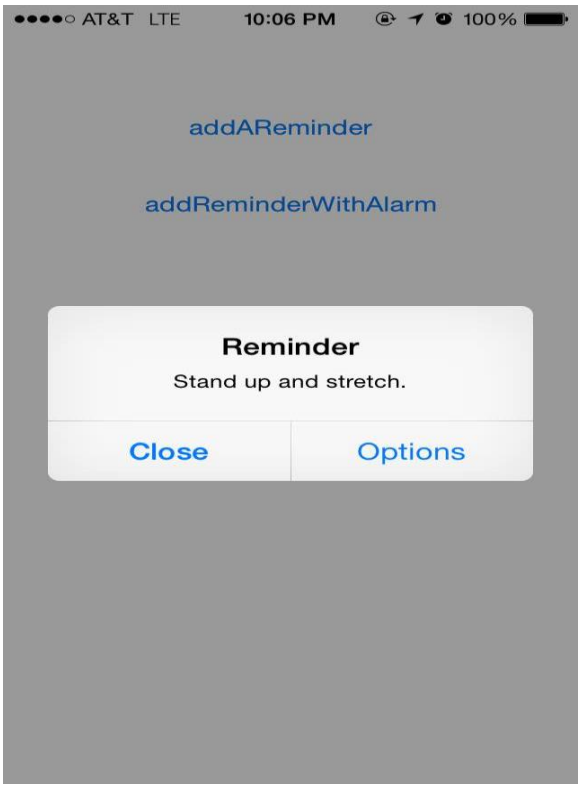
Application Specifications:

Platforms Used	**XCode, IntelliJ**
Languages Used	**Objective C, OpenCV, Scala**
Data Base Used	**MongoLab**
API's Used	**Text-to-speech**

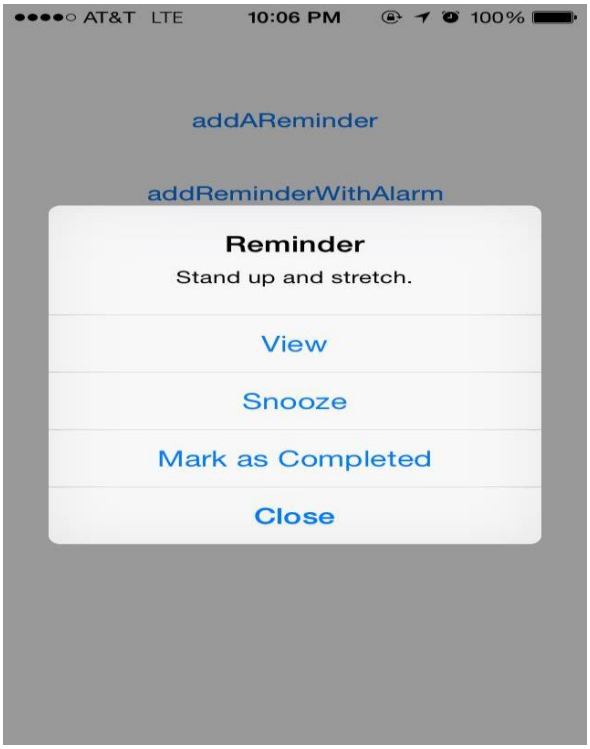
Screen Shots:

1.Reminder:

- Add a reminder will just add a new reminder.
- Add a reminder with alarm will create a reminder an also keeps an alarm for that reminder.



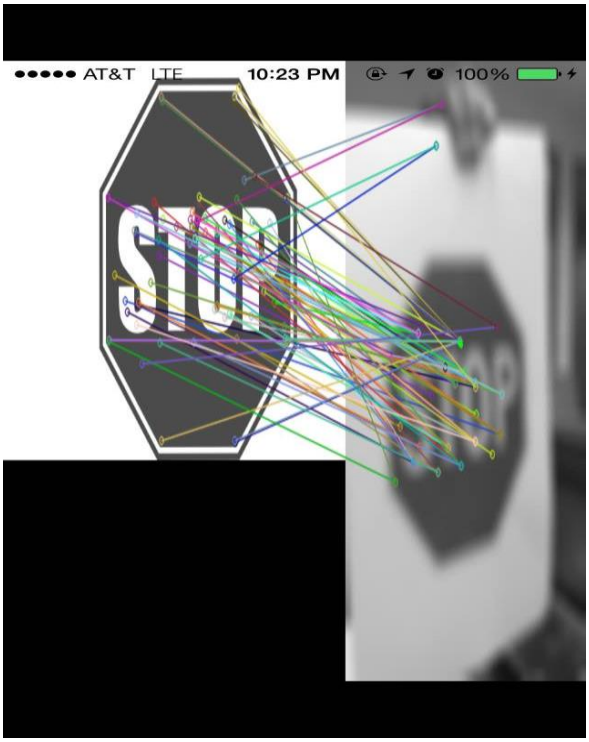
- When its' time to remind, it will give an alarm sound with a dialog box displayed as shown in the figure.
- If the user clicks on close, it will close the dialog box.
- If the user clicks on options, it will display another dialog box as shown in the below figure.



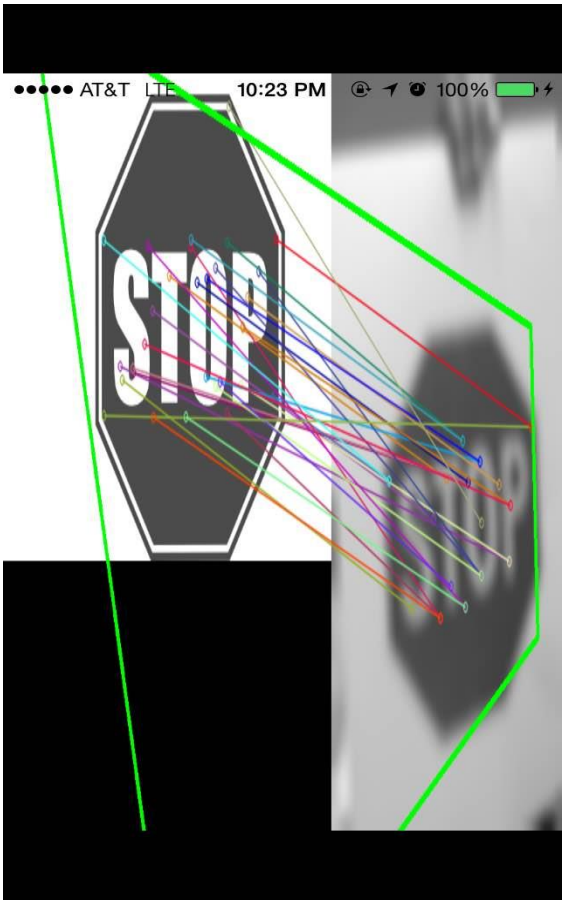
- The following dialog box is popped up when the user clicks on options button.
- If the user clicks on view button, he will be navigated to the reminders page.
- Snooze button is used to snooze the alarm for a particular amount of time.
- If the user completes his task for the reminder, he can click on Mark as completed button. It will off the alarm.
- Close button closes the dialog box.

2. Feature Detection and Extraction

- For feature extraction, first we have to store the image.
- For detecting the images we used camera view ie., live capture. Whenever it captures that image, it extracts the features and matches those with the stored image.
- From the figure we can see that features being extracted.

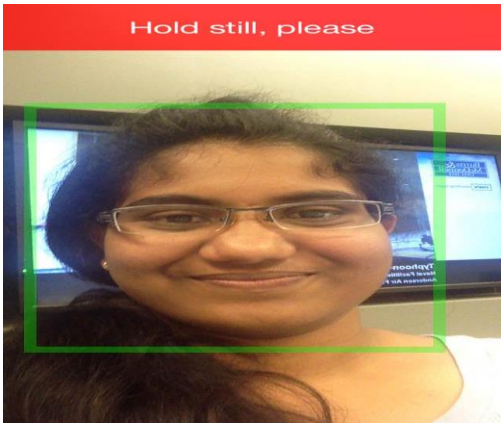


- The extracted features will be matched to the already stored image.
- Here we can see, the first STOP sign which is bright is already stored image and the STOP sign which is somewhat blurred is the image that is being captured.
- Features from the captured image will be matched with the features of the stored image which can be seen in the screenshot.



- From the screenshot we can clearly see the green border being formed around the extracted and matched features.
- Whenever it matches the features, it forms a green border around that.

3. Face Recognition:

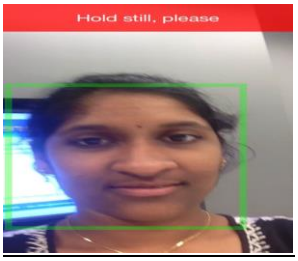


- Here first we gave the training data. Then whenever it captures our image, it will match with the trained data.

```
2015-07-13 23:11:06.432 KairosSDKExampleApp[5132:2391945] transaction came here {
    confidence = "0.798551499843597";
    "distance_apart" = "0.005545556545257";
    "gallery_name" = gallery1;
    height = 152;
    "matching_threshold" = ".75";
    "next_subject" = Sarika;
    "next_subject_confidence" = "0.79300594329834";
    "simularity_threshold" = "0.1";
    status = success;
    subject = 14;
    topLeftX = 78;
    topLeftY = 84;
    width = 152;
}
2015-07-13 23:11:06.433 KairosSDKExampleApp[5132:2391945] Response: success
2015-07-13 23:11:06.433 KairosSDKExampleApp[5132:2391945] Name: 14
```

- From the figure, we can see that the response message is SUCCESS.
- It also displays our name. We implemented this by giving names to the trained data.
- Whenever it captures an image, it will compare with the trained data and will display the name of that matches both the captured image and image in the trained data.

Another Example



```
2015-07-13 23:04:01.879 KairosSDKExampleApp[5122:2389094] transaction came here {
    confidence = "0.803997218608856";
    "distance_apart" = "0.075101792812347";
    "gallery_name" = gallery1;
    height = 197;
    "matching_threshold" = ".75";
    "next_subject" = Sarika;
    "next_subject_confidence" = "0.728895425796509";
    "simularity_threshold" = "0.1";
    status = success;
    subject = Vaishnavi;
    topLeftX = 5;
    topLeftY = 33;
    width = 197;
}
2015-07-13 23:04:01.884 KairosSDKExampleApp[5122:2389094] Response: success
2015-07-13 23:04:01.884 KairosSDKExampleApp[5122:2389094] Name: Vaishnavi
```

Here we can see that the response message as SUCCESS and Name as Vaishnavi. If the captured image does not match with the trained data sets, it will display the response as Failure. This is how we implemented face recognition with Greeting message.

4. Question and Answers

Our robot will respond to simple questions like what is your name ?, How are you ?,etc.. We implemented this feature with the help of NLP, and text to speech API. For this, we are using NLP to get the tags and tokens and we integrated a text-to-speech API for answering the questions.

For some questions like where are you? We are determining the current location of the robot by using google maps SDK.

5. Streaming Tweets

For this we are using Scala programming language and InteliJ IDE for downloading tweets based on the keyword “movies”. We are storing the collected tweets in the Mongo Lab.

From the rest service of the Mongo Lab, we are retrieving the data with the help of our mobile device.

6. Gesture Recognition

Whenever it recognizes a gesture movement, it will change its direction based on the gesture. Additionally it will speak out its direction. For example, it recognizes a gesture movement of left side, it will change its direction to the left and will say I’m turning left.