Autonomous Robotics: Greeter Robot Project Proposal

Lauren Dieker, Anne Oursler, Leah Holden

I. Intro

At times, Halligan can feel like a lonely or unwelcoming space. This is especially true when Halligan is almost empty or when you are surrounded by people who do not know you by name. As the core building for the Computer Science department, Halligan should feel like a welcoming and friendly space to encourage collaboration, provide a sense of community, and reduce stress. Our project aims to address this problem by creating a greeter robot. Our greeter robot would alleviate the loneliness of the building and provide a friendly face by greeting people it knows by name and learning the faces and names of newcomers. Our greeter robot is designed with Halligan in mind but could easily be used in other spaces on campus or in a non-university setting.

Another potential use of our robot would be to keep a record of who has been in the building, who is currently in the building, and how frequently people visit the building. This skill could provide improved security as well as a method of learning the habits and behaviors of the people the robot interacts with frequently. However, it would be important to ensure that we do not implement functionality that makes people feel watched or unsafe. We intend to alleviate this concern by being open about how and where the data is going to be used as well as by making the greeting robot look cute and behave welcomingly.

The main goal of our project is to create a greeter robot that can:

1) Learn and retain people's faces and names using data from a camera feed
2) Recognize faces in the camera feed that it has seen before and greet them by their associated name
3) Interact with users either through text or speech in order to ask for the names of people it does not recognize

II. Plan

The plan for the greeter bot can be broken down into several main goals:

1. Basic Facial Recognition
2. Learn New Faces While Recognizing Known Ones
3. Text Communication
4. Speech Communication

To allow the robot to recognize human faces, we will start by mounting a camera around the head level of the average person on top of the TurtleBot and connecting its feed to the robot. The TurtleBot already has a camera but the current camera is mounted too low to see faces— at its current level it will only see feet. We will use the feed from the new camera to produce image data for training and an existing open source ROS package for facial recognition in order to teach our robot to recognize different people.

Once the robot is able to recognize faces, we will adapt the facial recognition to learn a new face if it comes across an unknown face or a face it is unsure of. This will be more difficult because the existing facial recognition packages, like many machine learning algorithms, are designed to be used in two phases. First, the supervisor compiles a set of training images with known names and trains the classifier on that. Second, the supervisor uses the classifier to guess who it thinks a face in a new image belongs to based on its data. For our application, these two phases to be more integrated so that the robot can modify its classifier so that it can recognize new faces while in deployment.

As part of the facial recognition process, we will start by allowing the robot to communicate via text with a person. This communication will consist of prompting a user for their name if the robot does not recognize their face and greeting people by name if the robot recognizes them.

After this type of interaction is functional, our next goal would be to use speech to communicate with the people in Halligan. This is ultimately more useful because speech is a more natural way to greet someone. The robot would need to have programmed some welcome messages that can have names inserted into them and the question "What is your name?" When the user responds, we also want to filter their name out of any surrounding words or noise. This ability may be complicated by ambient sound in Halligan, so we may need to add the ability for the robot to re-ask questions if it cannot parse the speech it hears.

III. Related Works

In "Eigenfaces for Recognition," Matthew Turk and Alex Pentland describe a "near-real-time computer system that can locate and track a subject's head, and then recognize the person by comparing characteristics of the face to those of known individuals". Since we are also going to need to use 2D images to identify and recognize faces, this is extremely relevant to us. Turk and Pentland also focused on

making the learning real time, which will be something that we want to emulate in the facial recognition of our robot so that it interacts more smoothly with humans.[1]

In "Designing robots for long-term social interaction," the authors discuss Valerie the roboceptionist's functionality and their design decisions. Their goal was to program Valerie so that people are more likely to interact with her more naturally and for longer periods of time. Valerie's functionality mostly encompassed tasks like looking up office numbers, providing directions, and reporting the weather forecast. What is really interesting about Valerie is that the Drama group at Carnegie Mellon helped create a name, personality, backstory, and several storylines for the robot in order to make her more compelling. They programmed Valerie to talk about her "active love life and singing career that she pursues in her free time." This type of character development for the robot seems like an excellent way to engage with people. If we were doing a more long-term project, this would be an interesting direction to go with our TurtleBot after we implement facial recognition because we could program the robot to store information about its interactions with specific people, i.e. what stories it has told them or details about them. This would make conversation between the robot and a human user much more natural as well. Overall, the discussions of what design decisions worked well or poorly for a robot in the role of a receptionist will be an excellent groundwork for our design.[2]

The STRANDS project, which looks at long-term use of robots in the service and security industries, would also be a natural work to consider in our project. Their discussion on how to design robots to function as autonomously as possible in navigation can definitely translate to how we think about programming our robot to handle problems as much as possible.[3]

IV. Technical Approach:

Our first step in implementing this project will be to mount and set up a camera at approximate face level. This will be accomplished primarily with materials from Home Depot. This camera either needs to be stable enough to be driven around on a TurtleBot or removable, since our robot does not need to drive for this project. Once the camera is mounted at a reasonable height, we will attempt to connect the camera feed from it into the TurtleBot, similarly to the TurtleBot's current camera at shin level. We will use this camera feed to get the data required to learn and recognize faces.

---

[1] https://www.mitpressjournals.org/doi/abs/10.1162/jocn.1991.3.1.71
[2] http://ieeexplore.ieee.org/abstract/document/1545303/?anchor=references
[3] http://ieeexplore.ieee.org/abstract/document/7948740/

There are already multiple methods of facial recognition implemented to work with ROS. We will start with trying to use the ROS facial recognition package described on http://wiki.ros.org/face_recognition, since this is most likely to be well-maintained and work on the TurtleBots. In addition, this method of facial recognition has existing tutorials which will help us get started. We will try to use the existing functions within this method to simultaneously identify known faces and learn new ones. If this method fails we can alternatively use patilnabhi's github repository (https://github.com/patilnabhi/tbotnav) by isolating the facial recognition sections in their code and modifying them for our purposes.

Once we get facial recognition fully running on a TurtleBot, we will be able to start programming the robot to interact with the people it meets, first with a text based interface and then with a spoken interface. The text based interface will exist primarily to test the facial recognition, since a robot flashing some text a knee level is unlikely to get very many responses. Getting the robot to prompt users with a text to speech pattern will be the first step, since we will not have to work so much with natural language processing for this step. We intend to have the robot combine phrases such as "Welcome to Halligan" to the name of the person if they are recognized. If they are not recognized, the robot will give a welcome phrase and then ask for the person's name.

Getting the person's name will be more complicated, since humans rarely respond with only their name. Instead of translating the entire sound bite to text and storing it, we will have to filter out data such as 'oh!' and 'my name is' and attempt to get only the person's name. We hope that we will be able to utilize gstreamer speech to text or google's speech to text (possibly both so that we can use google's better speech to text occasionally, but the free method more often). We hope to use packages such as gspeech or gstreamer to filter someone's name out of their response, however failing this we intent to use the TurtleBot to prompt the person to only say their name.

To implement all of this, we will need a video feed, sound feed, and text input from the TurtleBot. We will also need to use the Turtlebot computer's speakers and possibly screen to express the robot's greeting or question. Using this data, we should be able to execute the following action loop:

0. Take images from camera feed and look for a face

1. If a face is found, attempt to identify the face

2. If the face is known, greet the person by name (i.e. "Hello, [name]! Welcome to Halligan!")

3.a. If the face is unknown, greet the person without their name and ask for their name (i.e. "Hello! Welcome to Hallgan! What's your name?")

3.b. Wait for a name response by taking microphone or text input.

3.c. Learn the face associated with the name and greet the person again (i.e. "It's nice to meet you [name]! Hopefully next time we meet I will know your name!")
4. Goto 0.

V. Evaluation Plan:

From a qualitative perspective, we will evaluate our project by how well it fulfills our main goals. These goals include learning new faces, recognizing old faces, and interacting with people using their names. If our robot can satisfy these three requirements, we will be on the right path. If our robot can interact with its users in a natural flow of speech, we will have done quite well implementing the interaction side of this project.

Fortunately, we can quantify how well our robot learns and recognizes faces.. In machine learning, there are many ways to express and examine how well your classifier performs. We can evaluate the quality of our classifier after training it on some images by giving it a set of new images of faces it already knows to test how many people it can identify correctly. The number of correct identifications divided by the total number of images we present to the robot is the robot's facial recognition accuracy.

The performance of a classifier can vary widely, but one hard line is that our classifier must identify more than 50% of the faces given to it correctly. This line is the difference between a weak and a strong classifier. If the classifier has an accuracy less than 50%, we need to go find a different package for facial recognition or somehow approach this problem differently. The criteria for a true success would be if the robot could identify more than 80% of the faces it encounters. This ratio, four faces identified correctly out of five, is enough to prove that we can train a robot to recognize faces well. Essentially, reaching percentiles above this might require programming more complicated classification algorithms which would be outside the scope of this project unless we found ourselves with extra time.