Rosie: Telepresence Robot

1 - Introduction:

Rosie is a telepresence robot designed and developed by Discovery Lab students that features augmented reality capabilities. The goal of Rosie is to create a telepresence robot with the same core functionality as third party telepresence robots such as the Double Robot and the Beam Pro. Additionally, Rosie holds a number of unique qualities that are seen throughout different projects at the Discovery Lab. As Rosie is part of the Computer Vision Research Institute (CVRI), Rosie features a number of Computer Vision applications meant to enhance the user's experience. Rosie also features augmented reality capabilities through the use of the Google Cardboard, which is a low cost, virtual reality headset. Finally, Rosie attempts to augment various Discovery Lab outreach initiatives, both at a high school and college level to make both the Discovery Labs and more complex topics such as Computer Vision more accessible to the public.

2 – Background:

The Rosie project was started in the spring semester of 2015. Over the course of the semester, an initial design was created, materials and hardware components were purchased, the preliminary software programs to control the robot were written, and the first Rosie prototype was assembled.

For improved maneuverability and control, Rosie was designed to be an omni-directional in that she could move in any direction at any time. To accomplish these goals some of the initial hardware components included the Jetson TK1 single-board computer, four omni-directional wheels and high-torque motors with motor control board (a complete list of original and current components with website links can be found in the starter kit). Also, Rosie was designed to offer a more immersive telepresence experience to the end user by incorporating computer vision. A camera, two servos, and a pan/tilt system were purchased to be used as Rosie’s primary camera system. The images captured from the camera would be processed and output to an Android app which would in turn control the direction of the camera such that, when used with the Google Cardboard, Rosie would look wherever the user looked. Rosie’s chassis was also originally designed to be completely built from scratch but, after the lab’s 3D printer was made accessible, a chassis was eventually designed to be printed. Rosie’s camera mounts, wheel hubs, and base with motor mounts were 3D printed and an aluminum pole was used as the main camera stand. While Rosie’s base was structurally stable and she functioned properly, there were structural integrity issues related to camera wobble when she moved.

Although Rosie functioned at the end of the spring, two improvements needed to be made: 1) software needed to be improved to decrease image latency and application control lag, and 2) a

new chassis needed to be designed and built that included a more stable camera stand. Over the summer of 2015, work was done to make the first improvement; an improved communication system was created that allowed for increased transmission of both image and control data between the Jetson and the Android control application. This was accomplished by packet streaming over a direct TCP connection between the Jetson and the app, which was ultimately more efficient than the system devised during the spring. The summer interns also focused on adding audio streaming and voice recognition, and so they were unable to work on a new chassis or improving the stability of the original.

3 – Approach

The ultimate goal for the fall Rosie robot project was to create a new Rosie that was more stable than the original as well as being totally functional relative to the specifications set by Dr. Williams as well as the project's original objectives. While the original objectives of increased immersion and augmented reality have always been and continue to be a top priority, Dr. Williams has asked that Rosie also function as a form of two-way communication between the user and those in the environments they're traversing. To save time and money, an Android tablet will be added to Rosie's camera stand and Google Hangouts will be used for communication.

Rosie is comprised of a number of different systems and hardware components that each serve a specific function. The major or more significant elements of Rosie include the hardware elements such as the Jetson, Arduino, and motors, the communications system, the Android control application, the audio and video systems, and the redesign. Each major portion of Rosie will be discussed in this section.

3.1 – Jetson

The main source of processing power for Rosie comes from the Nvidia Jetson TK1 developer board. It is an ARM-based board that comes with a 32-bit ARM Cortex A15, 2.3 GHz quad core processor. The Jetson also features 2GB of RAM and a Nvidia Tegra TK1 SoC (System on a Chip) that is compatible with Nvidia's CUDA language and holds 192 Kepler GPU cores. This hardware is very powerful given the small size and footprint of the Jetson at only 25 square inches. Additionally, the Jetson has relatively low power consumption that can be powered by a 12 volt, 3 cell Lithium Polymer battery and with a 12 volt, 5 amp power supply.

The main advantage that the Jetson presents is its Tegra TK1 chip. With its processing power, the Jetson is capable of many computationally intensive tasks that would cause a similarly sized system to lag behind or falter. The main intent behind the Jetson is to use it for image processing tasks such as motion tracking and facial recognition. Both of these systems are relatively intense on the hardware that it is run on, making the Jetson an obvious choice with its increased power

yet miniscule physical footprint. Additionally, Nvidia, the developers of the Jetson, provide a modified version of the popular computer vision library OpenCV that is optimized to run on the Jetson and utilize its Tegra TK1 chip.

The Jetson runs off of a modified version of Linux that is also developed and maintained by Nvidia. This distribution, called Linux4Tegra, contains specific drivers and software configurations for it to run on the Jetson's hardware. Unfortunately, running an alternative distribution of Linux, such as Ubuntu or Debian, is not natively possible due to the hardware on the Jetson. Regardless, since the Jetson runs off of Linux, development for Rosie is relatively straightforward, and the resulting applications for Rosie utilize very common Linux applications such as g++, pkg-config, cmake, git, and more. Therefore, development for Rosie is not limited to just the Jetson and can be done on almost any Unix-based machine.

The Jetson interfaces with the rest of Rosie mainly through USB connections. Rosie's Arduino and Motor Driver are connected to the Jetson through a USB to Serial connection. Additionally the webcam on Rosie is connected to the Jetson through USB. Rosie connects to the Discovery Lab network through a wireless connection. A user can then connect to Rosie through the Discovery Lab network. Rosie also has a backup connection method through Bluetooth for local control and for demonstrative purposes.

3.2 – Video

The Jetson captures and processes video from two webcams; a Logitech webcam is attached to the pan/tilt system at the top of the camera stand above the Android tablet and a Sony PlayStation Eye is attached to a stationary mount below the Android tablet. Both cameras are connected via USB and their output is accessed/processed by an OpenCV program that then streams the output through the robot's server program to the connected client.

The OpenCV computer vision library primarily uses the VideoCapture class to capture and process images. The frames that the VideoCapture class retrieves have a resolution of 640 by 480 pixels for both cameras and are in BGR format, meaning each frame has full color. These frames are then processed and sent to the control application for viewing.

3.3a – Image Processing

The Jetson TK1 uses a Nvidia GPU which incorporates Nvidia's CUDA architecture and specialized OpenCV libraries. OpenCV contains a special GPU class which can be used to run image processing programs directly on the Jetson's GPU, which is far more efficient than running on the CPU alone.

The frames that are captured from the robot's primary cameras can be processed in a variety of different ways. Over the spring and summer, a program was developed to track and identify the faces of the individuals working in the lab; this processing could be used to display data about certain individuals to the user in real time.

Ultimately the only image processing currently being utilized by this fall's Rosie is a program which combines the frames from Rosie's two main cameras. The program captures a frame from each camera and then superimposes the frame from the downward facing camera in the bottom-right corner of the frame of the primary camera attached to Rosie's pan/tilt. This way the user can look where they want to look as well as see Rosie's base without having to switch views.

3.3b - Facial Recognition:

The facial recognition system is designed to be run off of the robot and provide real time facial recognition capabilities to the user. The most commonly used computer vision library, OpenCV, is used as it provides a straightforward class of functions for facial recognition called FaceRecognizer. With the FaceRecognizer class, one of three facial recognition algorithms used, including: eigenfaces, fisherfaces, and local binary patterns histograms (LBPH). The facial recognition application uses the fisherfaces algorithm as it performs better in variable lighting environments compared to eigenfaces. The LBPH algorithm has better accuracy over eigenfaces and fisherfaces but only with smaller test sets for each person (<5 images per person), however relatively large test sets (10+ images per person) are made for each individual for this application so the LBPH algorithm does not provide any significant improvement and is not used.

The fisherfaces algorithm takes a holistic approach to facial recognition by reducing the training images to a lower dimensional representation that contains only the most important information. The eigenfaces algorithm performs the same reduction using a statistical procedure called Principal Component Analysis. A high dimensional dataset or image contains a very large number of dimensions, many of which are not significant or useful for classification or recognition. Therefore, the PCA procedure finds the meaningful or useful dimensions that have the greatest variance and can be used for classification. The fisherfaces algorithm is a modification of eigenfaces and instead uses a procedure called Linear Discriminant Analysis, This procedure takes into account classes, which are developed in the data by external sources, such as light. In an environment with variable lighting, the eigenfaces algorithm would fail as it would smear together data with little to no distinct information and fail to classify anything. The fisherfaces algorithm attempts to find features that separate best between classes where data within the similar class are held together and different clusters or classes of data are separated. This creates a more robust algorithm for situations or environments with more external variables.

All three of the facial recognition algorithms require training images for each person the facial recognition algorithm needs to recognize. With the initial development of the algorithm, the

AT&T Facedatabase was used, which contains 10 images for each of 40 different people. This database was used just to test that the algorithm worked. Because the application is being used to detect faces through a video stream, the easiest way to test the application is to create test image sets from volunteers at the Discovery Labs and then attempt to recognize those individuals. In order to create suitable test images, certain guidelines should be followed

- The face should occupy as much of the image as possible.
- The face should be about the size size and be in the same position in the image between all images in the set.
- Any background should be of a single, light, monochrome color such as white or yellow. Note that all images will be grayscale so light colors will suffice.
    - If the background contains any significant or distinct features, such as a painting or a light switch, then the fisherfaces analysis will generate inaccurate data.
- Each image within all test set images for every person must have the same pixel dimensions (width * height).
- Aim to gather at least 10 images for each person the application must recognize.
    - More quality test images will only improve the application at the relatively cheap cost of storage space.
- Several of the images should have the face positioned at slightly different angles. (+- 15 degrees in all directions).

In order to simplify the process of creating test image sets, a Python script was created that attempts to automatically crop and resize the image such that the image adheres to the above guidelines. Additionally, the script automatically creates a .csv file that holds the relative location of each image file and an identifier for which person that image corresponds to.

The application itself is relatively simple due to the streamlined nature of the FaceRecognizer class. The fisherface algorithm is used through a FaceRecognizer object and is initialized with the *createFisherFaceRecognizer()* function call. There are two main functions within this application: *train()* and *predict()*. The *train()* function accepts an array of images, which with OpenCV takes the form of the Mat structure, and an array of labels, which is an integer identifier that the image corresponds to. The array of images and the array of labels both have the same number of elements and for each element within the array of images, the element at the same position in the array of labels must correspond to the original image. After creating both of the arrays, they are used to train the FaceRecognizer object.

At this point, the FaceRecognizer object can take in any image and attempt to recognize a face, however attempting to recognize a face within a relatively large image with the possibility of having many people within the image will lead to false results and poor performance. Therefore, only sections of the image that contain a face are used when attempting to recognize a face. Again, OpenCV provides a number of easy-to-implement functions to recognize and single out

faces within an image. First, a CascadeClassifier is used to detect the location of faces within an image. The CascadeClassifier object is used for object classification. Different cascade files can be loaded into the object in order to detect different types of objects such as faces or vehicles. Each cascade file is trained with a large number of sample images of an object as well as a number of negative images, or random images that do not correspond to the object in question. OpenCV provides a frontal face cascade file which is meant to find regions of interest within an image if the region is likely to show the desired object, which is in this case the frontal view of a face. Using the cascade file and the CascadeClassifier object, the position of each of the faces is found and stored in an array of coordinates. Then for each set of coordinates, the region of interest is extracted and resized to fit the dimensions of the training set images. Using the *predict()* function call, the FaceRecognizer object takes the resized image and returns a prediction for the identifier that the input most corresponds to. With this output, further data can be gathered from a database or from storage to present to the viewer. Currently the application simply prints the predicted identifier of the individual.

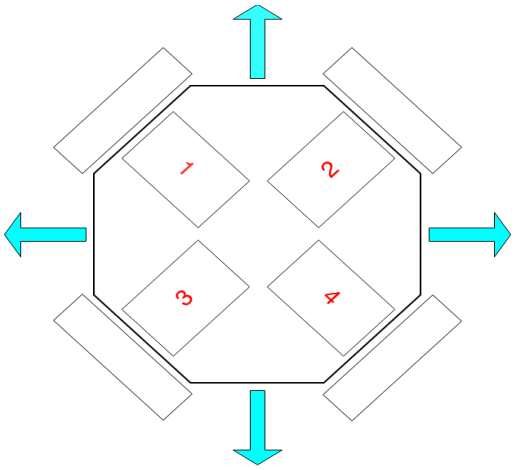
In order to further improve the performance of the facial recognition algorithm, a specific version of OpenCV is used to take advantage of a relatively new feature within OpenCV called the Transparent API. In the older and more common versions of OpenCV (2.4.10), developers can use a specific set of functions and data structures that are optimized to use OpenCL-compatible hardware. These functions and structures can utilize more processing power from the hardware within a system and can generally perform better than a non-OpenCL optimized application. With OpenCV version 3.0.0, this separate library is no longer used and is replaced with a new API that automatically detects whether or not an OpenCL optimized function or data structure can be used and automatically uses the optimized version instead of the unoptimized version. The API accomplishes this without needing to change the source code at all, essentially meaning two applications are created from one source file: one that is optimized for OpenCL hardware and one that is not.

3.4 – Two-way communication

Rosie also features an Android tablet mounted between the cameras and is meant to show the user controlling the robot and allow the user to easily communicate with those they see through the primary cameras. The Android tablet is wireless and is completely separate from the rest of the robot. The tablet's only purpose is to connect to the Google Hangouts application which the user will also have open on their end, either on a computer or another tablet. This solution for two-way communication was chosen to save time and money in an attempt to focus on finishing the development of the other parts of the Rosie build.

3.5 – Motion

Rosie features 4 omnidirectional motors that allow for movement in all directions with ease. These motors are mounted at a 45 degree angle from the main axes of Rosie (see diagram below).



These four motors are connected to a motor driver and an Arduino, which is used to control the motors and therefore the movement of the robot. In order to properly control the direction of Rosie, the motors need to be configured in the following fashion.

Direction	Motor 1	Motor 2	Motor 3	Motor 4
Forward	Counter Clockwise	Clockwise	Counter Clockwise	Clockwise
Backward	Clockwise	Counter Clockwise	Clockwise	Counter Clockwise
Left	Clockwise	Clockwise	Counter Clockwise	Counter Clockwise
Right	Counter Clockwise	Counter Clockwise	Clockwise	Clockwise

The four motors are each 12V DC motors that are powered through the motor driver by a 12 volt, 3 cell lithium polymer battery.

Also attached to the motor driver and Arduino are two servo motors that control the orientation of Rosie's webcam. These servos each have a 180 degree range of motion, meaning that Rosie can only "see" in a hemisphere facing the front of Rosie. The servos are powered through the 5V and ground pin on the Motor Driver and the signal is connected to two pins on the Arduino for control.

3.6 – Robot Server

Rosie utilizes an Apache server which hosts a PHP script that is accessible by the Android client application over the Internet. The script handles variables passed by the Android app using the server's IP address and then writes out to the serial port associated with the Arduino. The commands passed by the PHP script are simply integer values that are then handled by the Arduino to control either the motor driver or the pan/tilt servos.

The output images, processed by the OpenCV program, are also saved to a folder in the server. The image are title "foo.jpg" and are accessible at the same IP address as the PHP script. The JPEG file is continuously re-written when the OpenCV program is running and will not accumulate memory.

3.7 – Android Client Application

The Android client application is what the users will actually use to control Rosie. The Android application can be separated into four Activities which represent each unique window seen in the application: Main Activity, Info Activity, View Only Activity, and View and Control Activity.

The Main Activity is the main menu which allows the user to connect to Rosie, view info about the robot and the project, and select either the View Only Activity or the View and Control Activity. In the Main Activity, the user connects the application to the robot's server via an IP address entered by the user. If the user doesn't have the IP address for the robot or possess an incorrect IP address they will not be able to access the robot. The application allows the user to test their connection with Rosie before attempting to control the robot. Once the user has successfully connected to Rosie they will be able to either simply view the camera output or view the camera and control the robot, and selecting either of these options will open the associated activity. Finally, the Main Activity also allows the user to open the Info Activity which contains information on the projects, the project's past and present participants, as well as contact information.

If the user wants to only view the output from the robot and not actually controls the robot's movement they would select the ViewOnlyActivity. The ViewOnlyActivity simply accesses the "foo.jpg" image on the robot's server via an HTTP request. The request runs continuously in a Java Runnable, decodes the data into a Bitmap, and then loads the Bitmap into each of the ImageViews in the Activity's associated layout.

If the user wants to view the image output AND control Rosie they would select the ViewAndControlActivity. The Activity accesses the same "foo.jpg" image as the ViewOnlyActivity but can also control the robot (obviously). The activity open's a register for the devices orientation sensors and continuously access the real-time data values (tracking head

movement) which are eventually used to control the pan/tilt servos of the robot's main camera. The Activity also handles keyboard input from either a Bluetooth controller or keyboard and, given the associated data value for each key, outputs the value for commands that are used to control the robot's movement. The data values for both the key input and sensor data are output via HTTP POST requests to the URL of the PHP script on the robot's server. As stated the PHP script handles the variables attached to the URL for either the sensors or key input and output the values to the serial port of the attached Arduino for final control and movement.

3.8 - Redesign

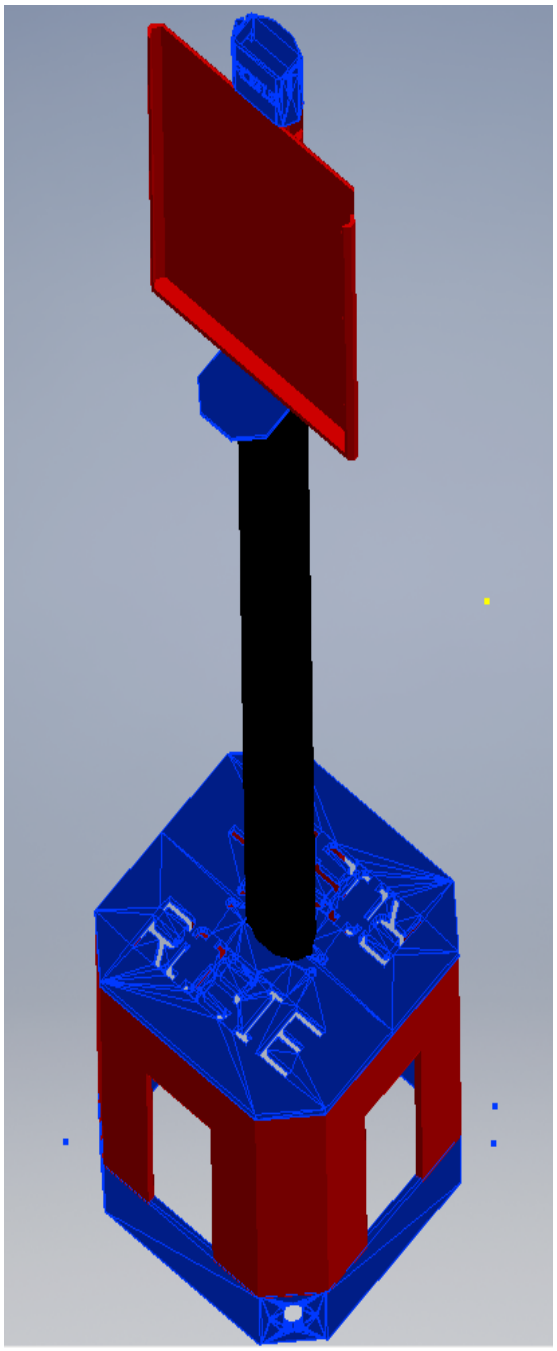
Rosie went through substantial redesign over the fall semester. An entirely new chassis was built from scratch and a housing and more specific components were 3D printed.

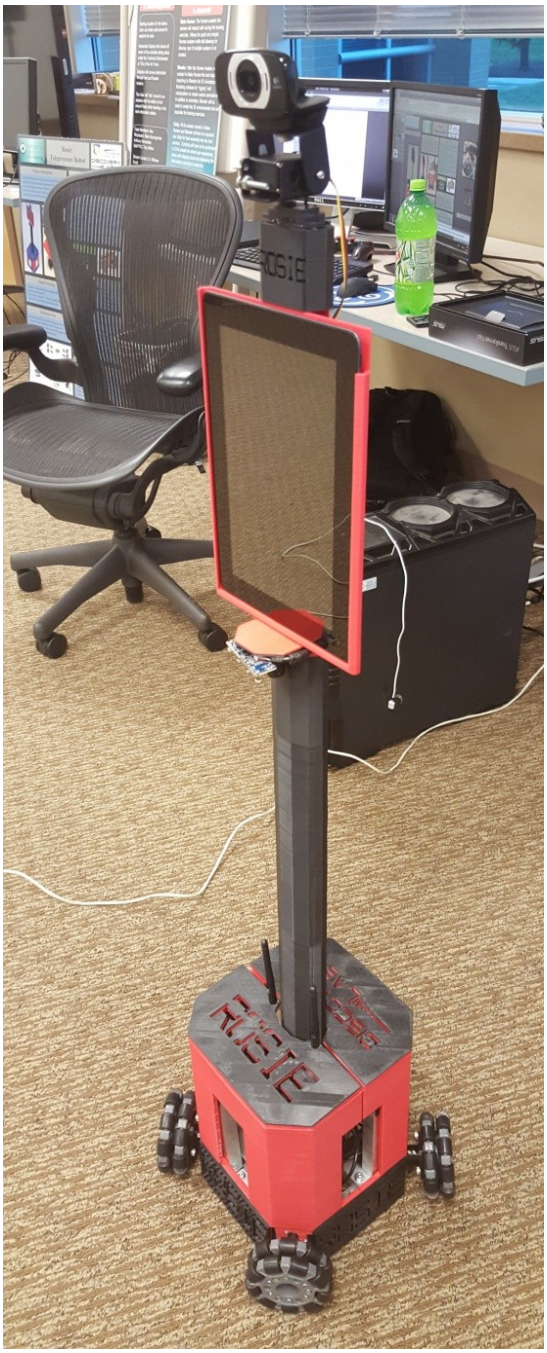
The chassis was designed to be smaller, to more easily navigate congested environments, and more stable, to reduce stand wobble. The chassis was built with a heavy 1/2" wood base and aluminum tubing over several days. The aluminum was used to create a frame for the components of the robot which were eventually mounted vertically due to the small footprint of the robot. The three foot portion of the aluminum tubing was also used as the camera/tablet stand and mount to the base. All aluminum tubing, including the stand, was securely mounted to the wood base using both epoxy and bolts/nuts. The electronic components were mounted using various methods. The batteries for the Jetson and Motor Driver board were simply mounted using Velcro. The Jetson and the Arduino/Motor Driver board were mounted to acrylic panels which were then mounted to aluminum frame. Again, all components were mounted vertically to save space.

Custom 3D printed parts were also created including a housing to cover the robot's chassis, a stand cover with ports for cable management, new wheel hubs to reduce wobble, a new motor frame that was smaller than the original, a tablet holder for the Android tablet, and two custom mounts for both the lower camera and pan/tilt system that held the robot's main camera. The parts were printed on both the Rostock Max v2 and LulzBot Mini 3D printers using both PLA and ABS.

Upon completion of the prints, the chassis housing was attached to the aluminum using Velcro, the webcam and pan/tilt wiring was ran through the stand sleeve which was then went over the aluminum stand, and finally the tablet and camera mounts were attached to the top of the robot (see images below).

Rosie CAD Design **Final Rosie Robot**





4 – Results

At the end of the Y2015F semester, Rosie is completely operational and can allow a user to remotely traverse certain environments with the use of the Google Cardboard and Android application as well as allow the user to hold a two-way audio and video conversation with those in the environment they are traversing.

The primary focus of the semester was the complete chassis overhaul that not only improved Rosie's overall movement and functionality but also made her appear as more of a finished product.

5 – Future Work

Future work is planned to continue to improve Rosie's overall control and performance. Currently, the use of sockets for both image and control data transfer over URL requests is in the developmental stage but it would allow better control of Rosie over a the Internet and reduce the overall lag experience with using simple URL requests. Further image processing is also a possibility as the performance capabilities of the Jetson TK1 board could be tested with respect to more process intensive augmented reality tasks (facial rec., HUD, etc).

All together Rosie is a step in the right direction with respect to remote telepresence. The robot is an affordable solution to basic telepresence that also offers opportunities for further development and learning with respect to image processing. The robot itself can be used for outreach and further research, and is a great learning tool for anyone interested in robotics and computer vision.