A Denunciation of the Monochrome:

Displaying the colors using LED strips for different purposes.

Tijani Oluwatimilehin, Christian Martinez, Sabrina Herrero, Erin Vines

1.1 Abstract

The interaction between robots and humans has been riddled with the gaps between the unskilled and the robots and this has led to the slow acceptance of the robots by humans. Robots, as of now, have no emotional quotient like humans possess and cannot really make people aware of what they are seeing or better still, who they are looking at without the people looking at the life feed from their cameras. This is what we sought to change by making the robots in our lab change the color of the LED strips attached to it to mirror the color of the shirt of whomever it is looking at. Also, the robot 's LED will also display a red color if it is surrounded by more than five people at a time. This will remove the previously inevitable need for the robot to plot a different course to avoid a cluster of people if it can just tell them that it needs to get through. And this will be done using the LED light.

1.2 Introduction

Non-verbal communication is essential to a significant amount of interactions humans have with each-other. Eye contact and other visual cues can provide important contextual information. Robots are incapable of performing the same nonverbal behaviors that humans exhibit. Our plan is to use an LED strip in order to communicate tidbits of visual data without having to look at a monitor display. The LED will also be useful in helping humans to observe the robot's behavior from a distance. We have settled on two main ideas, each of which utilizes the LED strip as a method for the robot to visually communicate information to human companions. The first of these ideas is for the robot's LED strip to display different colors depending on proximity to obstacles. The second is for the robot's LED strip to display the shirt color of the human with whom the robot is interacting. Both of these will allow the robot in question to non-verbally communicate with humans by changing the color of its LED strip.

2.1 Background and/or Related Work

During the course of the semester, we had projects in our FRI stream- Autonomous Intelligent Robotics that helped us get the technical know how to implement this simple but useful project. Since our project involved publishing and subscribing to ROS topics, the tutorials on how to publish and subscribe to ROS topics was very useful and this can be tied to the project we did at the beginning of the semester were we published random float values and also subscribed to the topic that published these values and then printed the values from the subscribing node to the console. Also, for our last project, we made a node that got live feed from the Kinect and detected a specified color from the feed. This project was basically the back-bone of our own project since we needed to get the color from the camera feed and then publish it.

2.2 Technical Approach

The first part of this project involved subscribing to the camera feed to be able to get the images from the camera for processing. To get the right camera topic to subscribe to, we turned on the robots and got went through all the topics from RVIZ which is run when the bwi_launch is run. There were two topics from the Kinect, the "/nav_kinect/rgb/image_raw" and "/nav_kinect/rgb/image_color". The first one gives the feed in black and white so that wasn't useful for our project so we subscribed to the second one. Once that was done, we needed cv_bridge to convert the image coming in from the camera feed into the format we need and also to make sure that the image we would process was 340 * 240 and had the image encoding of rgb8. This image was stored as "outImage" and then we had to send it to the person detect method. By looking at the goal of the project, it is very clear that we need to be able to tell a person that is in the image and to do this, we needed a node from opencv. The node was called person_detect.cpp and it finds the people in an image of 340*240 because anything more than that will cause serious lag. The software was a little inaccurate as it made wrong classifications sometimes but it was not enough to render it unusable.

When a person is detected in an image, a rectangle is drawn around the person and the rectangle's upper-left x and y coordinate are stored. Now, since there may be more than one person in the feed and sometimes, there may even be wrong classifications, I made the person_detect only draw the rectangles if it has an area larger than the current maximum area which was stored in a variable named max_Area. I based this approach on the fact that people will most likely focus on the person closest to them and is this person is closest to you, he or she will appear larger than someone that is further away. So this means that the closest person will have the larger rectangle of all the others so it is the one that will be focused on. Also, the false classifications in the image cannot be as big as a person so they will be ignored and the result is not altered by them.

2.3

Now after getting the right person and their rectangle coordinate, we need to do a little calculation to determine what area in the rectangle will most likely be the shirt area. This was done using the symmetry of the human body. The human head and the human neck take up about one-quarter of the total human height which I assumed will be the length of the triangle. This means that the shirt should start from about ¼ the height of the person for the y-coordinate and about ½ to the right of the upper-left corner of the rectangle for the x-coordinate. This can be seen in the diagram below.

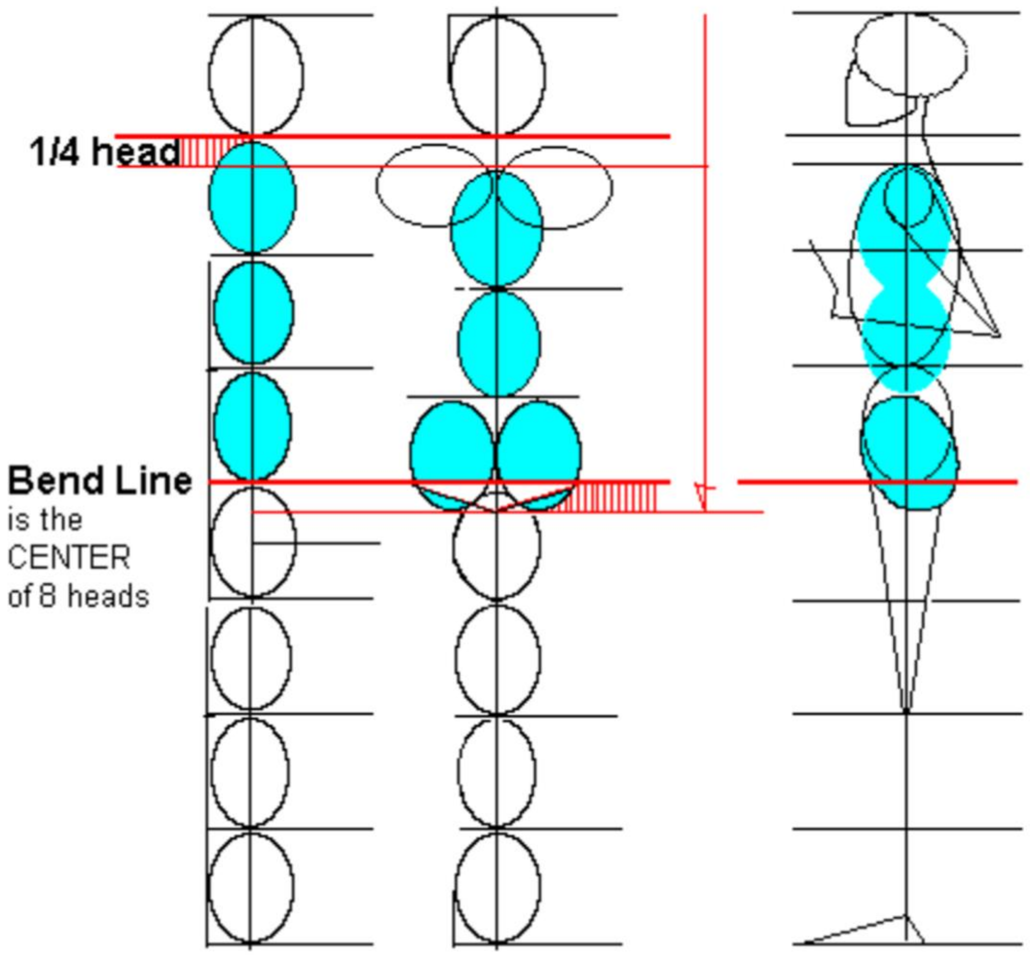

FIG 1.1 The symmetry of the human body showing the ratio to get a point in the image

The formula for getting the x-coordinates and the y-coordinates is:

```
int x_shirt = r_x + (1.0/2.0)* width_val;

int y_shirt = r_y + (3.0/7.0)* height_val;
```

After getting the x and y coordinates for a sure point on the shirt, I loop through five pixels to the right, five pixels to the left and five pixels up as well as down. For each pixel, I add the corresponding red, blue and green values to the total for each color and the final color to be published is gotten by finding the average of each individual color. This is done to handle the case where the shirt has bits of other colors on it and so if the point picked is a different color from the rest of the shirt, then the average will ensure the color displayed is not affected by just that one bit. Now that we have the values to be published, we made a publisher named "color_chatter" that published topics of type rgba. The color_chatter message was picked up by another node which was the Arduino-ROS node. It was subscribed to the color_chatter publisher and it received the rgba messages. These messages were then uploaded to the Arduino which then send the final message of rgb values to each LED in the LED strips and they will light up the color of the shirt. If the number of rectangles detected by the opencv person-detect is more than 5, the rgba value published irrespective of the closest triangle is 255.0, 0.0, 0.0, 1.0 which is red. When this happens, the node does not process the shirt colors at all.

3.1 Experiments and Evaluation / Demonstration

To test this while we were still coding, we recorded a ROS bag of a person walking around in a room wearing a blue shirt with white writings on it. This bag was recorded using the same camera we were subscribed to so it worked as if we were receiving a live feed from the camera. This demonstration worked perfectly and we got different shades of blue ranging from a very light shade of blue to a darker shade of blue depending on how much of the white pixels were around the point we picked as the possible shirt point in our code. We also tested the project with a live feed and we got the LED to change to orange when and orange shirt was weld up by a person and also to red when a red shirt was held up. Although it picked up the colors better when the person was closer, the code seemed to work very fine.

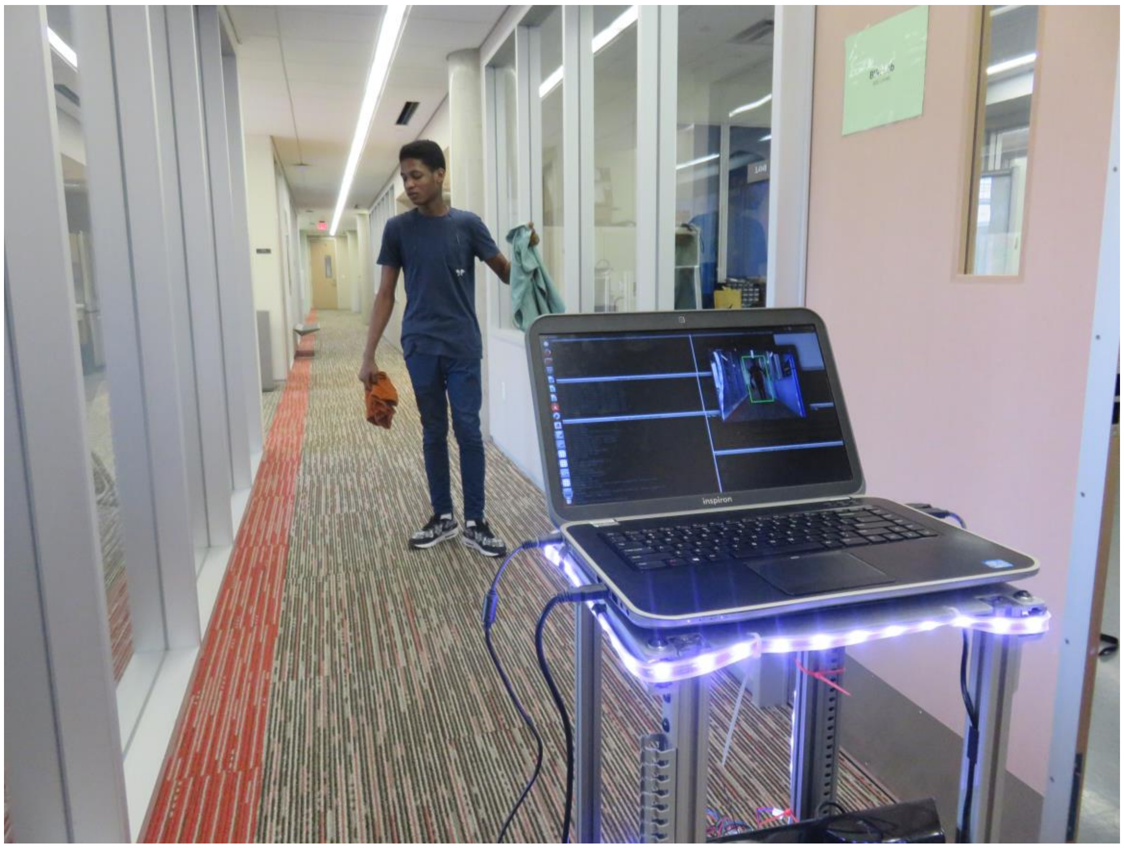

FIG 2.1 Testing the color recognition of the program and the resulting color of the LEDs.

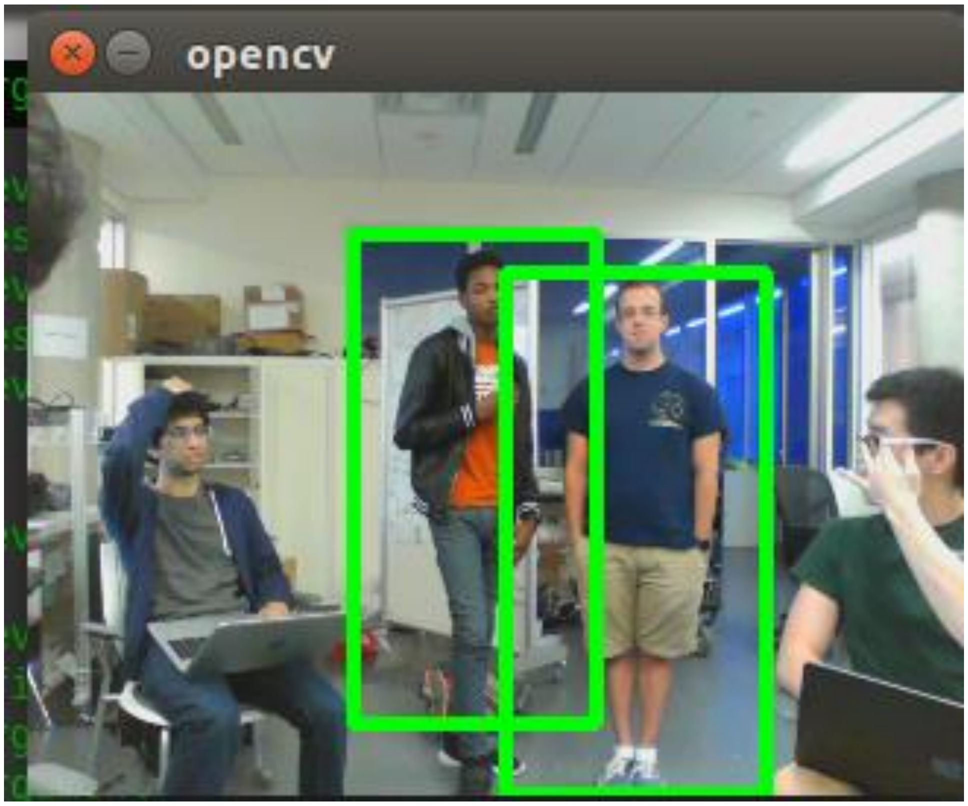

FIG 2.2 A sample of the person detect code that gets the people in the image.

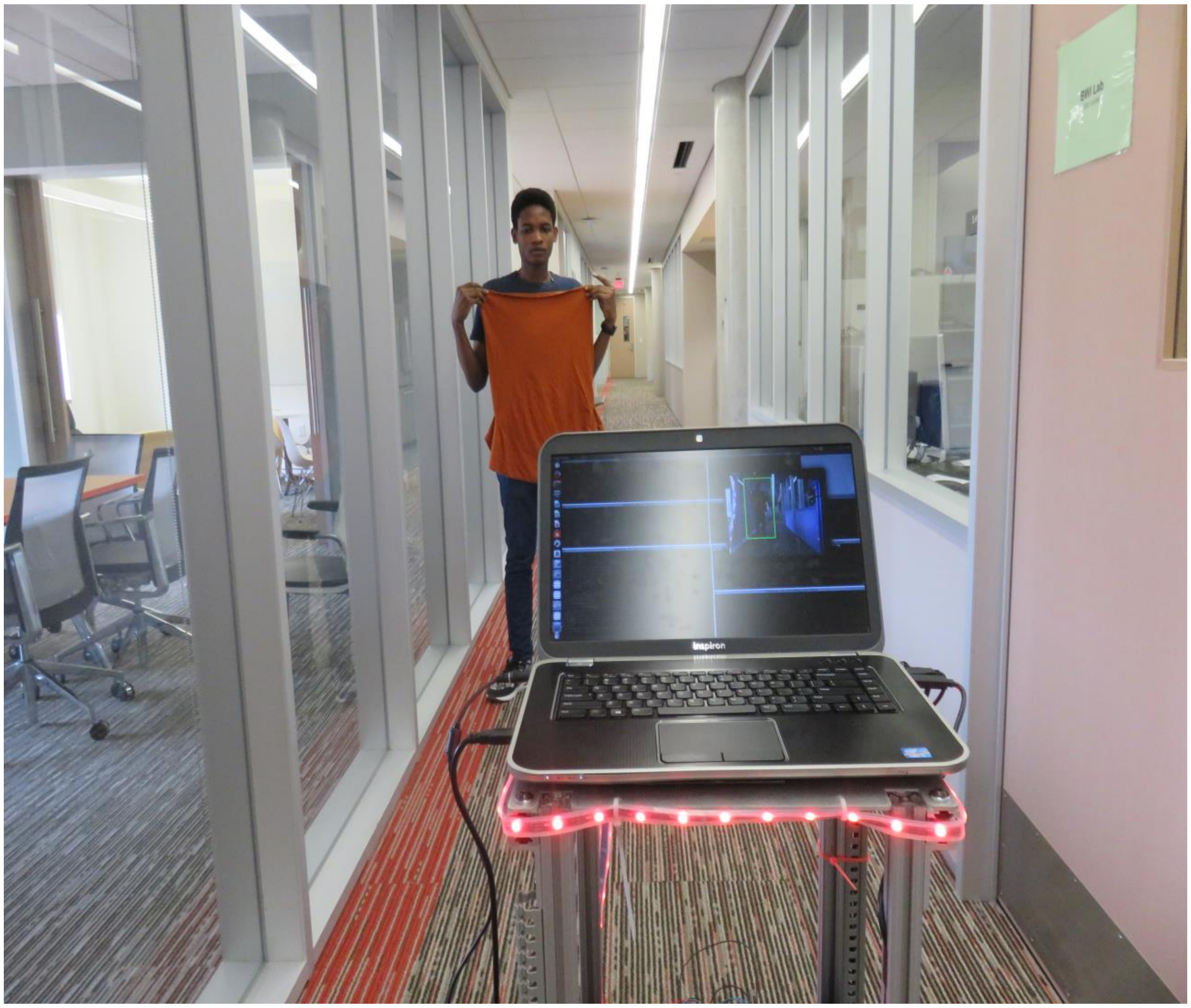

FIG 2.3 The LEDs light up orange to show the color of the shirt that is present in font.

4.1 Conclusion

In conclusion, we have shown that the robots can be more than just gray machines cruising round the lab floor without any form of expressions. They can be more interactive and intuitive in the way they let people know what they see and when they need a clear path. That being said, I would love to be able to undertake a project were a robot can have human-like moods and also decide if a request or a command is safe before carrying it out. For example, if a robot is told to keep walking straight even when there is a wall, if I can fully implement my project it should be able to stop and explain why it didn't go through with the command.

5.1 References

[1] the Arduino IDE and install the ros_lib library a tutorial can be found here:
http://wiki.ros.org/rosserial_arduino/Tutorials/Arduino%20IDE%20Setup

[2] The first step to this project is to download the rosserial package
http://wiki.ros.org/rosserial_arduino

[3] The source code for the person detect:
http://www.magicandlove.com/blog/2011/12/04/people-detection-sample-from-opencv/

[4] Pololu Source
https://github.com/pololu/pololu-led-strip-arduino.git