Background

In the last two decades, researchers have put effort into creating a robot photographer designed to emulate a human professional photographer in social events, such as parties, family gatherings, and public spaces. In the early 2000’s, robots were equipped with cameras, taught the Rule of Thirds, and placed into such gatherings with facial detection capacities in order to take pictures for individuals and groups. However, the robots were not particularly interactive with their human subjects, unresponsive to signals such as waving and speech, and often got distracted by shiny environmental stimuli. A few years later, microphones and voice detection mechanisms were built into these systems, allowing the robot to use triangulation to detect vocalized interest for camera attention; it could respond to waving now, too. More recently, researchers have been able to train these robotic photographers to learn more advanced photography techniques, such as the using the guidance of composition lines to frame the subject of the photograph.

Abstract

This semester, we have begun designing, building, and testing a robot system that is able to use a camera to take pictures of human subjects while interacting effectively with them. We have conducted studies involving professional human photographers and human subjects, and have seen how each party behaves in a social setting. By examining the subjects' reactions to the photographer’s presence and
actions, we have gained insight into what interactions, instructions, or social cues are amenable to people who want to or are being photographed. We have also investigated methods by which to deliver photographs to their human subjects.

Our system uses a Shutter robot that incorporates human face detection, tracking movements for alignment of the subject for interaction and photo positioning purposes, various input and output channels tied to a dialogue tree, photo-taking capacity, and transfer and sharing of photos into private Google Drive folders.

**Procedure: Design and Implementation**

Our design of the robot photographer revolved around the interactions that it would make with the human subject, which would enhance its functionality beyond that of a normal camera. We decided to use facial detection to initiate the photo-taking process because it signaled the entrance of a human subject. The robot then asks the subject(s) if they want a photo taken, and upon receiving an affirmative answer, emits a positive reaction and proceeds to deliver a 10-second countdown before taking the picture. If the subject is unhappy with the photo, it is retaken until the subject is satisfied, upon which the photo will be uploaded to a private folder on Google Drive, and the subject will be presented with a QR code that links them to the photo. The process is diagrammed in the following figure:

![Interaction workflow for the shutter robot.](image)
This was my first time being exposed to robotics, so I spent some time at the beginning learning ROS and familiarizing myself with the various visualization tools in the environment. To implement the face detection, I wrote a ROS node that used the OpenCV library and kernel to parse potential facial regions from each frame of the camera, before publishing it to the main node to initiate the aforementioned action sequence. After the face detection node was working with test files, I researched the Intel d435 camera that the robot was equipped with, configured it, and linked its API with the face detection node.

Once the person detection, speech-to-text and text-to-speech, and photo capture and uploading were split and completed within our three-person team, we found that during testing, the microphones had significant trouble picking up speech when the slightest noise was present in the environment. Taking into consideration the vision of the robot operating in public spaces, we decided to add an additional communication channel that utilized screen output and keyboard input. Out advisor suggested looking into PyGame, so I read through PyGame documentation and tutorials to write a ROS node that subscribes to a topic that provides a message from the dialog tree in main and blits the message onto the screen one letter at a time, changing color based on the connotation of the message and ending with a blinking underscore to imitate listening to the human or prompting for keyboard input. Because the blinking underscore seemed to fit the structure of an infinite loop, I ran into thread synchronization problems with the callback methods from the other nodes; we fixed this by learning about and using Python’s Lock library for mutual exclusion.
Next, I worked with one of my team members to implement a controller to take an image and its depth image, then publish the 3-dimensional position and orientation of a face. Together, with a provided tracking node from our advisor, this allowed the robot to move its screen to follow one of the subjects in the frame as they moved around its field of view. We had a lot of trouble with not only the bare calculation of the face’s coordinates, but also the flexibility of the robot’s movements and the instability resulting from an overshoot in position and subsequent overshooting adjustments. We lessened the impact of this by thresholding the input rate to only process a frame every 0.5 seconds. We also improved the calculation by adding lower and upper bounds to the depth of the face in order to accommodate either the absence of a face or the extreme positioning of one. At present, the two nodes function quite well together, and the robot is capable of following people’s faces not just to appear more engaged in the interaction, but also to center the subject within the picture for a more appealing photo.

**Next Steps**

Now that we have a basic process functioning, future steps will focus on enhancing the human-robot interaction, as well as experimenting with more photography techniques to improve the satisfaction of the image.