Department of Electrical & Computer Engineering

Boston University
Electrical & Computer Engineering
EC463 Capstone Senior Design Project

First Prototype Test Report

OPTICLE

by

Team 3
Opticle

Team Members

Annamalai Ganesh (amg1998@bu.edu)
Luca Guidi(lucag@bu.edu)
Jami Huang (jamih@bu.edu)
Stefan Wong (spwong@bu.edu)
Nancy Zheng (nancyzhe@bu.edu)

I. Equipment/Required Materials

A. Hardware:

- Raspberry Pi 4 Model B (with 16 GB MicroCenter SD card)
- 15W Power Supply (Pi 4)
- OpenCV AI Kit: OAK-D Camera
- Linear Resonant Actuators
- SB300 Solderable PC Breadboard
- Desktop monitor
- Keyboard
- Mouse

B. Software:

- Python 3 Scripts:
  - Object detection (YOLO with SSD)
- Motor Control:
  - Raspberry Pi GPIO

C. Other:

- Platinum Extreme Accessory Kit (Chest Mount)

II. Set-Up

Our set-up was consistent with the first prototype test plan. Prior to the demo, we put markings on the ground at one meter intervals up to five meters from the test subject with tape so that the “moving obstacles” knew how far they needed to stand during testing. Jami served as the test subject wearing the chest mount, while Stefan, Nancy, and Luca served as “moving obstacles”.

Our setup included both software and hardware components: an object detection Python script that is run on a Raspberry Pi 4, which was connected to an OAK-D camera and wired to two linear resonant actuators. The hardware was attached to a chest mount where the camera was mounted to a screw in the middle and the Raspberry Pi was attached to the left strap of the chest mount. The Pi was connected to a monitor display via HDMI so that the Raspberry Pi Desktop could be used to run the script.

For the software, the python script *spatial_tiny_yolo.py* was prepared on the terminal and was to be run on the OAK-D camera. The python script uses a trained algorithm Yolo, which performs real-time object detection. When an object is detected, an appropriate label is assigned to it as well as the spatial coordinates (XYZ) of the object relative to the camera. Using the z-coordinate, the strength of a PWM signal is scaled accordingly and sent to the appropriate GPIO pin on the Raspberry Pi. The closer an object, the stronger the vibration. To determine which of the two actuators should vibrate on the chest mount (right or left), the x-coordinate of the object is used; any negative coordinates should cause the left actuator to vibrate and positive coordinates should cause the right actuator to vibrate. Each actuator was wired to its respective GPIO pin to receive a PWM signal when the camera detects an obstacle. Jami held both linear actuators in each hand to show that they were vibrating accordingly.

III. Measurements

Key:

Annamalai Ganesh	Presenter
Jami Huang	User
Luca Guidi	Left Obstacle
Stefan Wong	Right Obstacle
Nancy Zheng	5m Obstacle
Motor 1	Left Actuator
Motor 2	Right Actuator

1. We started the demo by running the file *spatial_tiny_yolo.py* and ensuring that the OAK-D camera was running properly. This was satisfied as the monitor showed both the depth map window and the camera window. The camera window showed the objects detected by placing blue boxes with the object label and the XYZ coordinates.
2. Based on the objects detected, if there was a person recognized, we ensured that the linear resonant actuators produced a vibration of an intensity ranging from 0 to 100.
3. Luca and Stefan simulated obstacles and stood 2 meters away from the user of the device (Jami). This produced a medium strength intensity of ~65 on both motors as anticipated as shown on the monitor.
4. We then asked Luca to step forward by 1 meter resulting in an increased vibration intensity for motor 1 as it changed from ~65 to ~80 while motor 2 remained at ~65. This shows that each motor was attached to a specific range of x-coordinates. We then asked Stefan to also step forward by 1 meter resulting in the vibration intensity for motor 2 to change from ~65 to ~80 as well.
5. Subsequently we asked Luca and Stefan to leave the frame of the OAK-D camera showing that the vibrations were only present when a “person” is detected. This criterion was satisfied as well as not only did the motors stop vibrating, the terminal on the monitor stopped providing the vibration intensities for the corresponding motors.
6. Lastly, we asked Nancy to stand 5 meters away to the right of the user and check to see if the motor 2 still vibrated. This is because one of the requirements from the client is to check and see if all obstacles under 5 meters are detected. Thus, if objects at a distance of 5m can be detected, objects under 5m will be detected as well. This requirement was satisfied as motor 2 did vibrate with a very small relative vibration strength of ~10.

IV. Conclusions

Based on the results of our testing, we can conclude that our proof of concept works when it comes to detecting people and vibrating actuators based on the distance of the person from the user. The correct actuator vibrated when the camera detected a person based on whether the obstacle was on the left or right side of the camera frame. This tells us that the camera's spatial detection is working properly and we can rely on using the x-coordinates to determine which actuator should vibrate and the z-coordinate to determine the strength of the vibration.

We also gained some key insights during the test such as the scope of the camera detection and how we should plan to detect objects that are within the detection view but may not be directly in front of the user. A concern such as a chair that would be in the corner of the camera's frame came up since it could be detected even though it is not directly in the user's path causing the user to overcompensate their change in path for an obstacle that isn't necessarily dangerous. We plan to implement a range of five actuators across the user's body to more accurately inform the user where oncoming obstacles are. This way, even if they receive a vibration, they can know if it is on their far left or far right side allowing them to make a decision on how they would like to adjust. There will also be audio output that announces what the detected obstacle is, which can further supplement the user with information to help them make a decision on where to walk. We also can look into splitting up the camera frame further to include the y-coordinate so that obstacles that may be on the edge of the frame do not need to be detected and told to the user. Another insight that was gained was that the machine learning model we currently use is not at the accuracy we would like and plan to train our own model to have better consistency and only detect the obstacles that we want to look for.