Reach Out and Grab It!

Seo Ho Kang
Nicholas Milton
Fabio Vulpi
Problem

• Visually impaired people have difficulties to detect and retrieve objects in unknown environment.

Status Quo

• Orcam
• Aira
• Personal Assistance
User Requirements

• Fashionable
• Can pull it out only when you need it
• Lightweight
• Voice control like siri
• distinguish between coke and pepsi

Semester Goals

• Computer vision navigation
• Haptic feedback glove to guide hand
• Audio interface for object request
Actuation Hardware

rFduino on custom PCB on Glove

- Compact and lightweight
- Vibrating Motors protected
- Single wireless article
- Gazelle proprietary communication Protocolc

Bluno Beetle on watch

- Compact and lightweight
- Vibrating Motors in rings
- More Stable camera mount
- Arduino ide support
PCB schematic and board made with Eagle software, showing Bluno Beetle, four identical motor control circuits (transistor, resistor, diode for back EMF) voltage regulator, switch and battery support.
Software

1. Voice
2. Android Phone App
   - Text
3. Processor
   - Raspberry Pi 3
   - Coral Edge TPU
4. Watch Arduino
   - Camera Feed
   - Movement Direction
   - Actuation
## Pronunciation Dictionary

<table>
<thead>
<tr>
<th></th>
<th>Pronunciation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GARAGE G ER AA ZH</td>
</tr>
<tr>
<td>2</td>
<td>IN IH N</td>
</tr>
<tr>
<td>3</td>
<td>IS IH Z</td>
</tr>
<tr>
<td>4</td>
<td>OKAY OW K EY</td>
</tr>
<tr>
<td>5</td>
<td>OPEN OW P AH N</td>
</tr>
<tr>
<td>6</td>
<td>PI P AY</td>
</tr>
<tr>
<td>7</td>
<td>RAMONA R AH M OW N AH</td>
</tr>
<tr>
<td>8</td>
<td>SHUTDOWN SH AH T D AW N</td>
</tr>
<tr>
<td>9</td>
<td>START S T AA R T</td>
</tr>
<tr>
<td>10</td>
<td>THE DH AH</td>
</tr>
<tr>
<td>11</td>
<td>THE(2) DH IY</td>
</tr>
<tr>
<td>12</td>
<td>TIME T AY M</td>
</tr>
<tr>
<td>13</td>
<td>TRANSLATOR T R AE N S L EY T ER</td>
</tr>
<tr>
<td>14</td>
<td>TRANSLATOR(2) T R AE N Z L EY T ER</td>
</tr>
<tr>
<td>15</td>
<td>WEATHER W EH DH ER</td>
</tr>
<tr>
<td>16</td>
<td>WHAT W AH T</td>
</tr>
<tr>
<td>17</td>
<td>WHAT(2) HH W AH T</td>
</tr>
</tbody>
</table>
1-1) Run script with command below.

[Command]
python3 demo/object_detection_ver5.py \
--model test_data/mobilenet_ssd_v2_coco_quant_postprocess_edgetpu.tflite \
--label test_data/coco_labels.txt
2-1) FPS optimization-used parser to organize script

[Snap of object_detection_ver5.py]

```python
parser = argparse.ArgumentParser()
parser.add_argument(  
    '--model', help='Path of the detection model.', required=True)
parser.add_argument(  
    '--label', help='Path of the labels file."
parser.add_argument(  
    '--input', help='File path of the input image."
parser.add_argument(  
    '--output', help='File path of the output image."

args = parser.parse_args()
```
2-2) FPS optimization-use array input

We chose to input image array to “mobilenet_ssd_v2” directly after flatten the image. We previously input image but after changing it to input array, FPS increased by 5~10. Flatten step was done because Google TF lite api asked us to put flatten image. Plus, in order to get fast FPS you should follow the way we implemented with pi-camera.

```python
camera.framerate = 60
raw_capture = PiRGBArray(camera, size=(IM_WIDTH, IM_HEIGHT))
raw_capture.truncate(0)
start = time.time()
counter = 0
for frame1 in camera.capture_continuous(raw_capture, format='bgr', use_video_S:
    frame = frame1.array.flatten()
    ans = engine.DetectWithInputTensor(frame)
    raw_capture.truncate(0)
```
3-1) Communication (Android app <-> Raspberry Pi3)

We use socket serial communication between app and rpi3. App converts voice to text and send it to raspberry pi3.

```python
from socket import *
import time
import socket
import RPi.GPIO as GPIO

HOST = '
PORT = 21567
BUFSIZE = 1024
ADDR = (HOST, PORT)
tcpSerSock = socket(AF_INET, SOCK_STREAM)
tcpSerSock.bind(ADDR)
tcpSerSock.listen(5)

def setup():
    with open('coco_labels.txt') as mf:
        ta = mf.readlines()
    labels = [i[:-1] for i in ta]

def get_label(tcpSerSock):
    while True:
        print('Receiving data...')
        tcpclisock, addr = tcpSerSock.accept()
        print('Address: %s, %s' % (addr))
        try:
            while True:
                print('Now start receiving data...
)
                data = tcpclisock.recv(BUFSIZE)
                print(type(data))
                data = str(data)
                print(type(data))
                if not data:
                    break
                for i in labels:
                    if label in data:
                        return label
            else:
                print('error')
        except:
            print('error')
```
3-2) Communication (Arduino <-> Raspberry pi3)
We use ttyACM0 serial communication.

```python
ser = serial.Serial("/dev/ttyACM0", 115200)
ser.baudrate = 115200
```
4) SSD Object detection model
Input is [300, 300] image array and we use label, and box (x,y) coordinate output. We use pretrained model.

```python
ans = engine.DetectWithInputTensor(frame)
raw_capture.truncate(0)
# print('captured %s' % filename)
frame1.truncate(0)
# rawCapture.seek(0)
counter += 1

ser = serial.Serial("/dev/ttyACM0", 115200)
ser.baudrate = 115200
if counter == 10000000:
    break
if ans:
    for obj in ans:
        if labels[obj.label_id] == lab:
            box = obj.bounding_box.flatten()
            box *= 300
            center_x = ((box[0] + box[2]) / 2)
            center_y = ((box[1] + box[3]) / 2)
            if center_x > 75 and center_x < 225 and center_y > 75 and center_y < 225:
                # further processing
```
Object Detection Different Light

Average Time per foot to grab items in different Lightings

lux

- 75 lux
- 120 lux
- 152 lux

AVG SEC/FT
Object Detection Different Objects

![Average Time per Foot of distance to Grab Objects](chart)

- bottle: 5.248958333 seconds
- cup: 14.7 seconds
- backpack: 5.8 seconds
- Bottle around many: 7.708333333 seconds
- BANANA: lux
Object Detection Different Scenario

Average Time per foot to grab a bottle alone vs surrounded by objects

<table>
<thead>
<tr>
<th></th>
<th>ALONE</th>
<th>SURROUNDED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>4.8</td>
<td>7.708333333</td>
</tr>
</tbody>
</table>

lux