Team 05 Documentation
Smart Trash Can

Team Members: Po Hsiang Huang, Abirami Sambamoorthy, Andrea Sudharta

Table of Contents

Introduction

As we consume more and more items in our daily lives - with just over 38 billion plastic bottles thrown into landfills each year in the US alone- more clear that we need to switch over to more sustainable lifestyles, and that includes properly sorting our waste into trash, compost, and recycling. At UCSD, even though we have separate trash and recycling bins, because people often times do not separate their waste properly, the school ends up throwing everything into the landfill. To tackle this problem, we have designed a machine that looks at an item and, using image recognition, determines whether the item is trash, recyclable, or compostable, and then sorts it accordingly. This project not only solves the problem of garbage often going to the wrong containers, but also educates the public on what category a piece of trash they throw away falls under.

Materials

- Raspberry Pi
- Raspberry Pi Camera
- Micro servo motor
- Stepper motor
- 5V solar panel
- 4 Rechargeable AA batteries
- Push button
- Laser-cut board
- 3D printed parts
- Cardboards
- Boba straws

Instructions

Designing the Trash Can

We wanted to create a trash can that can deposit the trash detected into the correct bin. This can be achieved by having a plate that propels the trash into the bins and a plate that can rotate and move desired trash bin to the location of propulsion. We used both solar panels and battery pack so that this trash can model is even more sustainable and can function for longer periods of time on its own. The plate we used to hold our 3 bins was created through Inkscape drawing and acrylic sheet laser cutting.

All of our 3D printed parts were created through SolidWorks. First we needed to determine the desired size of each bin. We made the mouth 6 inches and the base 4.5 inches in diameter. This was created through sketches on different planes and the loft feature. The stepper motor attachment was created through the same method but with extrude cut that fitted the axle of the motor. We created two other plates, the top one to hold the solar panel and camera, the bottom one to hold the trash and drop the trash into the bins. The top plate was a simple model created through 2 extrusions which allows the solar panel to rest on top of. The bottom plate was a more complex model because it needs a socket to fit the micro servo motor in order to rotate. Pay close attention to the dimensions of each part because they have to match each other. The sockets for attachment should be slightly larger than the actual component so that they can be attached easily.

Although we planned to 3D print most of our parts, we ended up using cardboards and straws as our side supports because they are more time and cost efficient. It is important to create multiple holes on the support for adjusting the heights of the plates if needed. The base needs to be heavy but small so we filled a small plastic water bottle with rice to hold up the rotating plate.

TensorFlow portion

Make sure you have Python 3, pip, and virtualenv installed in a Bash environment.

First, create a new virtual environment by using the command:

```
virtualenv --system-site-packages -p python3 ./venv
```

Where ‘venv’ is the name you want to give to your virtual environment.

Then, to activate the virtual environment from your home directory, use the command:

```
source ./venv/bin/activate
```

Now install TensorFlow by typing the following to the command line:

```
pip install --upgrade tensorflow
```

After that, create a new folder named ‘tf_files’, and inside that, create three folders inside that will represent the categories you will want to categorize the items into: trash, recycling, and compost. Then, put images in each folder that represents each category. For example, in the recycling folder, put in pictures of recyclable items (like water bottles). After that, download the code from the ‘scripts’ folder in this project’s GitHub repository (minus the pycache folder). [Note that the ‘scripts’ folder also contains the necessary code to run the other components of the trash-sorting mechanism. We will go into depth for this code after discussing TensorFlow]

Then, inside the main folder, set the following shell variables:

```
IMAGE_SIZE=224
ARCHITECTURE="mobilenet_0.50_${IMAGE_SIZE}"
```

Now that everything is set up, it's time to run the TensorFlow retraining script!

```
python -m scripts.retrain \
 --bottleneck_dir=tf_files/bottlenecks \
 --how_many_training_steps=500 \
 --model_dir=tf_files/models/ \
 --summaries_dir=tf_files/training_summaries/"${ARCHITECTURE}" \
 --output_graph=tf_files/retrained_graph.pb \
 --output_labels=tf_files/retrained_labels.txt \
 --architecture="${ARCHITECTURE}" \
 --image_dir=tf_files/flower_photos
```

Congratulations! You have successfully re-trained your TensorFlow model!

Programming the Servo and Motor

First, we need to install GPIO if it isn't already on your shell. To do so, enter the command `pip install RPi.GPIO`

Next, connect the software and the hardware via GPIO and input output wires. To do so, type in the following:

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
```

Now let's look at the code in the GitHub Repository in application.py. This documentation will go over the logic behind the code written but it won't go through line by line what the code is. In the end however you should get a good understanding of what the code does overall.

The purpose of the servo is to tilt the platform holding the trash and dispose it to the appropriate waste bin the stepper motor specifies. Check the function init_servo() and set the GPIO pin to 17. Then to change the position the servo tilts in, use the function ChangeDutyCycle. A duty cycle value of 2.5 corresponds to neutral position. We want it to steadily increase the duty cycle to make it tilt, and then make it go back to 2.5 by doing the following:

```
p.start(2.5) # Initialization
for dc in range(0, 18, 1):
  p.ChangeDutyCycle(float(float(2.5)+float(dc/5.0))
  time.sleep(0.05)
p.ChangeDutyCycle(2.5)
p.stop()
```

Then we work on the stepper motor code. Because the servo tilts in one direction to one location, we want the stepper motor to rotate around to the spot the servo will dump the trash in. First, we need to initialize the stepper in init_stepper() by setting the pins as following:

```
control_pins = [4,17,27,22]
for pin in control_pins:
  GPIO.setup(pin, GPIO.OUT)
  GPIO.output(pin, 0)
```

We then have an array of sequences for the stepper to go through and map those to the pins by doing the following:

```
forward_halfstep_seq = [
  [1,0,0,0],
  [1,1,0,0],
```

```
  [0,1,0,0],
  [0,1,1,0],
  [0,0,1,0],
  [0,0,1,1],
  [0,0,0,1],
  [1,0,0,1]
]

for i in range(steps):
    for fwdhalfstep in range(8):
      for pin in range(4):
        GPIO.output(control_pins[pin],
forward_halfstep_seq[fwdhalfstep][pin])
      time.sleep(delay)
```

Now that we have the code to make the plate with the bins rotate, we do the rotating given the name of the bin we want the trash to be dumped to by doing the following: have a variable representing the 'destination bin'. The destination bin - the starting point, 0 = the number of bins we need to rotate. We set the trash bin to 0, recycling bin to 1, and compost bin to 2.

```
def rotate( bin_name ):
  init_stepper()
  destination = 0
  if bin_name == "recycling":
    destination = 1
  elif bin_name == "compost":
    destination = 2
  print("Preparing to move ", destination*steps, " steps to bin ",
destination )
```

We then call the forward function and pass in the delay (0.001, written at the top of the file) and the number of bins to rotate through * 171 (512 steps corresponds to a full 360 degrees). Then, we tilt the servo once we rotate to the correct bin. Then, we rotate the bin back to the trash bin, which is originally 0.

```
  forward(delay, steps*destination)
  tilt()
  time.sleep(5)
  print("Returning to position 0")
  if destination != 0:
    init_stepper()
    forward(delay, steps*(num_bins-destination))
```

Putting it all together

If you have successfully copied all the code inside the 'scripts' folder of the GitHub Repository, you should have all the code you need to execute the application. Enter the 'scripts' folder and type this command into the terminal:

```
python application.py
```

Now that the program is running, make sure the Pi Camera is positioned correctly, place trash on the tilting platform, and press the button switch on your breadboard.

Conclusion

There were many ways we could improve the smart trash can given more time, funding, and resources. Appearance-wise, we could have 3D printed the third trash bin (we only had time to print two). Functionality-wise, we could have developed a smoother functioning of the tilting mechanism by going deeper into the documentation for servo programming, as it is jerky in its movement. Furthermore, our smart trash can is limited on what it recognizes as recycling and compostable waste - only water bottles are in the recycling data set, and only bananas are in the compostable dataset. So we could add more categories and pictures to the machine learning dataset.

There were two main obstacles that we had while creating this trash can. The first was 3D printing and getting the right materials for the trash can. Without a lot of time dedicated to gathering and creating the items along with the creativity to piece it together, we would not have the design we have right now. The second one was programming the individual motors to work with the TensorFlow code - the sudo command was necessary for controlling the motors, but sudo was not meant to be used when classifying an object as trash, recycling, or compost.

However, the time we dedicated to this and seeing this smart trash can come to life makes everything worth it. We were overjoyed when we were able to take a picture with our Raspberry Pi camera and compute its correct classification. Even greater were the memories we made along this project, from meeting weekly in the CS basement to running around campus late at night to grab materials last minute.

References

Tensorflow for Poets by Google:
https://codelabs.developers.google.com/codelabs/tensorflow-for-poets/

Controlling a Servo From the Raspberry Pi: RPi Labs by University of Ontario Institute of Technology:
https://rpi.science.uoit.ca/lab/servo/

Controlling Stepper Motors Using Python With a Raspberry Pi by Keith Weaver:
https://medium.com/@Keithweaver_/controlling-stepper-motors-using-python-with-a-raspberry-pi-b3fbd482f886

https://www.dosomething.org/us/facts/11-facts-about-recycling

http://eco-growth.com/

http://trialandstyle.com/

https://www.mantecabulletin.com/

Shoutout to Dillon for helping us with many of the roadblocks we came across while making this!