Nidhi Lawange's BSE Notebook

Project: Raspberry PI Image Recognition

<nidhilawange/Object-Detection-Lock (github.com)>

<Password-Protected Image Recognition | Raspberry-Pi-Image-Recognition (nidhilawange.github.io)>

Jul 5, 2022

Notes:

Setup Raspberry PI

1. Insert micro SD card into the Multifunctional Card Reader.
2. Insert Reader into USB port in computer → the computer extracted the data stored on the SD card for the computer's usage.

(Configuring the Hardware)

1. Modify settings for the Raspberry PI imager and let it download with the newly acquired data.
2. Download Bonjour Print Services, download VNC viewer, and download Visual Studio Code (VS Code).

(**Software**)

1. Used both the Command Prompt (Terminal) and VSCode: Used Simple Command: print("Hello World!").

Jul 6, 2022

Notes:

Install Tensorflow on Raspberry Pi Imager

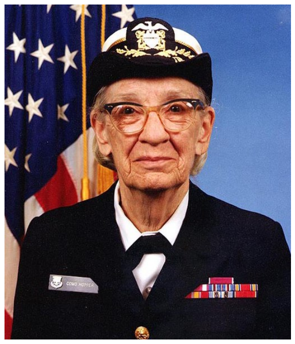
1. Wrote this code:
   ```
   sudo apt-get update
   python --version
   python3 --version,
   ```

   Followed by,

   ```
   sudo apt install libatlas-base-dev
   pip3 install --user tensorflow
   ```

   But, in the second line of the above code segment, I faced a problem. Tensorflow authors had updated the Tensorflow library, and thus the coding and commands had to be updated. I switched from Command Line Terminal to VS code.

Test Code with Installed Tensorflow

2. I switched to VS code in which I pasted code from the Tensorflow authors on github. This code provided a sample machine learning model and sample image that was processed using that model.
3. I faced permission issues with accessing directories though. After modifying the code itself, the machine learning model worked, and outputted the correct result.
   Given this picture:
   
   ,this result was outputted, showing the percentage possibility of each of these items being in the photo above:

   ```
   0.728693: military uniform
   0.116163: Windsor tie
   0.035517: bow tie
   0.014874: mortarboard
   0.011758: bolo tie
   ```

Jul 7, 2022

Notes:

1. Today, I tested another image on the generic image processing model that I worked with yesterday after a few modifications. After downloading an image of a puppy and dragging into VScode files, I changed the default image argument to “puppy.jpg,” and I wrote this command in the terminal:

   `python tftest.py --image color.jpg` [tftest.py - where model was tested, reading in new image]

   This command outputted:

   ```
   0.791826: 209:Labrador retriever
   0.183814: 208:golden retriever
   0.009317: 223:kuvasz
   0.006146: 853:tennis ball
   0.003327: 258:Great Pyrenees
   time: 139.473ms
   ```

2. I also came up with some new ideas today now that the basis of the project was set and working.
   - Connect the Raspberry Pi ArduCam to the Raspberry Pi microcontroller board in order to read in images to the machine learning model through a real camera, not just a downloaded image of the Internet
   - Expand the database of words off of which the model can match the image to
3. Today, I set up the Raspberry Pi camera, copied the Python code into VScode that allows the camera to function (turn on, take picture) (in “camera.py”), making sure to import PiCamera2.
   - Using the Pi camera, I took a picture of a calculator I had on my desk, and fed this image to another program (“controller.py”), in which I imported “camera,” which refers to “camera.py” with the camera’s code
   - This program was connected the image processing model before, so after inputting the photo of the calculator and running “controller.py,” it outputted:

     ```
     0.486765: 591:hand-held computer, hand-held microcomputer
     0.192401: 762:remote control, remote
     ```

```
0.110809: 488:cellular telephone, cellular phone, cellphone, cell,
mobile phone
0.107632: 811:space bar
0.023276: 708:pay-phone, pay-station
time: 138.929ms
```

4. With the camera now working with 1 static image, I added a while loop to “controller.py” so that the camera could repetitively take pictures of what it was pointing at and output results through image processing with the given model.

5. The next step that I am working on is how to input a live feed (dynamic image) into the model for image processing through picture frames taken one after another recursively instead of simply a static image input.

Jul 8, 2022

Notes:

1. Today, I worked on inputting a close-to-live feed of the image to be fed into the Raspberry Pi camera.
    - A problem that I faced was that the size of the image was too large
    - To fix that problem, changed the argument of Pi camera 2's method from
    -

```
      main={"size": (320, 240)}
```

To,

```
config = picam2.create_still_configuration({"size": (320,
240)})
```

Also, I added this piece of code to resize the image.

```
dim = (640,480)

cv2.resize(buffer, dim, interpolation = cv2.INTER_AREA)

color = cv2.cvtColor(buffer, cv2.COLOR_RGB2BGR)
```

- Another problem that I was facing was the delay between each picture being taken and showing up in VScode, so to fix that, I added this piece of code:

```
time.sleep(0.5)
```

0.5 represents a **0.5 second break** between each image appearing, and that worked. I was able to get practically a live feed of what the Raspberry Pi camera was pointing at. I linked "controller.py" to the machine learning model that was configured in "tftest.py" by importing "tftest" into "controller.py" and running the model based on the running of the camera functionality: this means that as the live feed came out of the Raspberry Pi camera, each object in the feed would be processed, and the percentage match would be outputted for it – constant image recognition feedback for dynamic image processing.

Jul 9, 2022

Notes:

1. Things left to do for entire project and some new ideas to finish off project:
    - Expand database of words and images that can be detected
    - Include servo motor functionality that opens like a gate based on image recognition → the image recognition would be a pattern of holding up different objects (ball, comb, bottle) as a password to move servo motor 180 degrees

Jul 11, 2022

Notes:

Setting up the servo motor:

(Hardware)

1. I connected the 3 female pins to the GPIO pins on the Raspberry Pi microcontroller board. I connected the male ends of each of these 3 wires to the servo motor wires, connecting ground, 5 V supply, and PWM signal from the servo motor. I also attached the rotating plastic shift onto the top of the servo motor, from which the motion mechanism can be observed.

(Software)

1. After importing the correct classes and files and setting up the General Process Input/Output (GPIO) pins, I added this piece of code to rotate the servo shaft 180 degrees:

```
servoMove.start(0)

time.sleep(3)

startCycle = 2

while (startCycle <= 12):

    servoMove.ChangeDutyCycle(startCycle)

    time.sleep(7)

    servoMove.ChangeDutyCycle(12)

    time.sleep(2)

    startCycle += 1

    time.sleep(2)

    exit()
```

Jul 12, 2022

1. The first thing I did was record my second milestone video for the attachment of the servo motor. Then I learned more about the machine learning model I was using to complete this project.
2. My next plans are to link image recognition to the servo motor’s functionalities as if the image recognition serves as a password lock.

Jul 13, 2022

1. Today, I linked the servo motor movement to the image recognition model.

   These were the steps I took to linking the image recognized by the model:

   I added this code in “servo.py”:

```
if (__name__ == "__main__"): #if the image recognized passes, turn servo

    angle = 0

    setPos(0)

    time.sleep(0.5)

    setPos(90)

    time.sleep(0.5)

    cleanup()
```

   I then imported “servo.py” into another new python file, “lock.py” in which I defined methods to “lock” and “unlock” the servo motor by simply setting the position of the servo shaft within these locking and unlocking methods. Then within an if statement that checks whether the name of the image processed is the correct one, and then uses the locking and unlocking methods, all in “lock.py”:

```
if (__name__ == "__main__"):

    lock()

    time.sleep(1)

    unlock()

    time.sleep(1)
```

By importing “servo.py” into “lock.py,” the servo motor is now being “used as a lock.” The servo shaft will turn 0 to 1800 degrees if the correct name of the image processed shows up, which leads us to the next piece of code in “tftest.py” and “controller.py”:

tftest.py

```
 return results[top_k[0]], labels[top_k[0]]
```

controller.py

```
       perc,name = tftest.runVision()

       print(name)

       if (name == "575:golf ball"):

           lock.lock()

       else:

           lock.unlock()
```

The return command added to “tftest.py” was put in the *runVision()* method which runs the machine learning model. This line returns the top name and label of the highest probability that the model believes matches with the object in the image after processing it.

Then in “controller.py,” which runs the camera’s live feed + image processing + lock/servo functionality by importing “lock.py” into “controller.py,” extracts the return value from *runVision()* in “tftest.py,” the name and label of the highest probability matching object. Then, the code checks whether this returned name is in fact “575: golf ball.” If this is true, the lock will lock, meaning the servo motor will turn 180 degrees or else it will stay unlocked, meaning the servo motor will stay at 0 degrees and not turn 180 degrees and back.

The combination of all of this new code addition led to the intended working project: when the Raspberry PiCamera recognized the golf ball in front of it with the highest probability, the servo motor shaft turned 0 to 180 degrees and back to indicate the “correct password.”

2. My further ideas are to include a pattern of object/image recognition with the highest probability: If a golf ball, then a tennis ball, then a perfume bottle are shown, then the servo motor shaft should turn.

Jul 14, 2022

1. Today, I enabled the servo motor shaft to turn based on the image captured by the Raspberry Pi Camera and processed by the model. This is the main piece of code that I added to “controller.py”:

```
if __name__ == "__main__":
    password = ["575:golf ball","575:golf ball","575:golf ball"]
    c.configCamera()
    while (True):
        c.runVision()
        tftest.runVision()
        time.sleep(0.5)
        #
        startTime = time.monotonic()
        progress = 0

        while ((time.monotonic() - startTime) < 10):
            perc,name = tftest.runVision()
            print(f"{name}, {password[progress]}")
            if (name == password[progress]):
                progress += 1
                startTime = time.monotonic()
                if (progress == len(password)):
                    lock.unlock()
                    break


        time.sleep(7)
        lock.lock()
```

What this piece of code does: The Raspberry Pi camera takes a picture of whatever is in in front of it. If it is not a golf ball, the servo motor shaft is set at the 0 degree position. A golf ball image must appear at least twice in front of the camera within the ten second period, but if the servo motor doesn’t “lock” or turn within that 10 second period, the servo motor will be restarted with the break from the while loop. The model will process the image as a golf ball if it identifies one.

2. My next step is to have the Raspberry Pi Camera take pictures of multiple different objects, create a pattern out of it, and have that pattern be recognized by the model as the passcode to moving the servo motor.

Jul 15, 2022

1. Today, I worked on implementing a physical structure as a lock, like a cardboard box. I attached the servo motor to the side of the box so that the servo shaft blade would rotate right above the box as a “lock” and back to 0 degrees as “unlocked”.
2. I also created a pattern of objects for the model to recognize, which serves as a passcode, and then the servo motor shaft turns 90 degrees “opening the lock” and enabling us to open the box.

   Here is the part of the code that I changed in “controller.py” to implement this: So now instead of 3 golf balls in a row, there’s a golf ball and a coffee mug that must appear in order. And when those objects are captured and recognized by the model, the servo motor will turn 90 degrees, thus opening the lock and letting me open the box.

   ```
       password = ["575:golf ball","505:coffee mug","575:golf ball"]
   ```

   So now, the goal of having a pattern of separate objects as a passcode to be recognized by a machine learning model to turn a servo motor as a “lock” has been achieved.

   These were the last steps of my project.