Team 11 Documentation
Pill Sorter

Table of Contents

Introduction

In a world full of different, colored, and unique pills, one can accidently confuse between a pair, taking the wrong pill and thus endearing oneself. We are here to solve this problem; using an automated pill sorter, the patient will have to simply put the pill underneath our camera, and wait for the sorter to put it in the right box. That way, the patient will not have to do any differentiation by himself or herself and prevent the danger in confusing pills!

Materials

- Computer (with Python, TensorFlow, Arduino IDE)
- 2 pieces of 12” x 12” wood suitable for laser cutting (or other material)
- 2 x Arduino
- 1 x camera module (arducam mini 2mp)
- 1 x 5V battery source
- 1 x SG90 servo
- 1 x stepper motor with controller
- Wires

Instructions

Begin by setting up the pill sorter structure. The physical structure of the device can vary and be created as the designer sees fit. The physical structure can be 3D printed, laser cut, or machined by hand depending on how the designer wants to go about constructing their parts and or what tools they have available to them. In our case, laser cutting was employed (design was done in Inkscape) and about 2 sheets of 12” x 12” plywood were used in order to get all the necessary parts.

Next, interface with the Arduino. There are two arduinos in this project. One will be used solely to control the camera module while the other will be used to control the motors. Hook up the camera module to an arduino as described by the data sheet associated with the associated camera module. Test the camera module to ensure that pictures can be taken and everything is functioning properly. If using the arducam mini 2mp, go to the arducam github to download the proper libraries and example code.

Next, set up the structure based on the selected design and wire the stepper motor and servo according to their specified data sheets. Write the code to control your motors based on the feedback from the neural network which will be fed images from the camera module.

Next, set up your dataset:

Step 1: position the Arduino camera

Step 2: Using the Arduino interface, capture pictures and send them to a specific folder

Step 3: Install Python 3

Step 4: Iterate over each of the images and add their specific pill type to their name. For example, for image “0.jpg” re-write to “white0.jpg” if it is a white pill.

Next, begin to set up the Neural Network.

**Note: We have attempted to use our own machine learning algorithm in Python using Keras. Due to our lack of success by the last day of build time, Dillon suggested we employ a pre-written code of his.

Step 1 : install all the necessary libraries in Python (TensorFlow, TensorFlow_hub, numpy, OpenCV, random, argparse, collections, datetime, hashlib, os.path, re, sys, ssl)

Step 2: Download retrain.py from E4E Mangrove github (and don’t forget to acknowledge them!). Find the code on this link:
https://github.com/UCSD-E4E/mangrove/tree/master/CNN%20Development

Step 3: run retrain.py and select the image folder as the folder where you saved all your images from the dataset set-up step; save the graph and the labels. You have just trained your network!

Step 4: Download annotated.py from E4E Mangrove github

Step 5: run annotated.py using the graph and labels saved from step 3; this is your classifier

Step 6: create a new file that will monitor the folder to which pictures from the arduino pictures are saved (so that upon any new capture, a new image is added to this folder, and the classifier will run). In this code file:

- Install the watchdog library
- Create an Event Handler function that will handle any change in that specific folder
- When a change occurs in that folder, call your classifier method from annotated.py
- Install the pySerial library (to interface with Arduino from Python)
- Based on the input of the neural network from annotated.py, write to the Arduino (using pySerial) a specific number between 0-3. Based on the number, the Arduino will know how much to turn (in order to put the pill in the right box).

Conclusion

If we had more time, we would have attempted to classify more pills instead of just 4. We would also continue to work on our neural network and try to make it work instead of using Dillon’s.

Our greatest obstacle was our Arduino camera: after a week of attempting to configure it unsuccessfully, we realized the Arduino board itself was the issue, and when we changed it to a new

one, it magically worked. Our greatest moment was on that same day, where by that night, our whole project was working successfully.

References

- Arducam.com for camera libraries and tutorials
- Dronebotworkshop.com for stepper motor tutorials and examples
- Arduino.com for tutorials and example code
- E4E Mangrove github (https://github.com/UCSD-E4E/mangrove/tree/master/CNN%20Development), which was the foundation of our neural network

And finally, thank you to Dillon and our mentor Michael for being there this quarter and answering all of our questions!