Raspberry Pi Face Recognition Treasure Box

Created by Tony DiCola

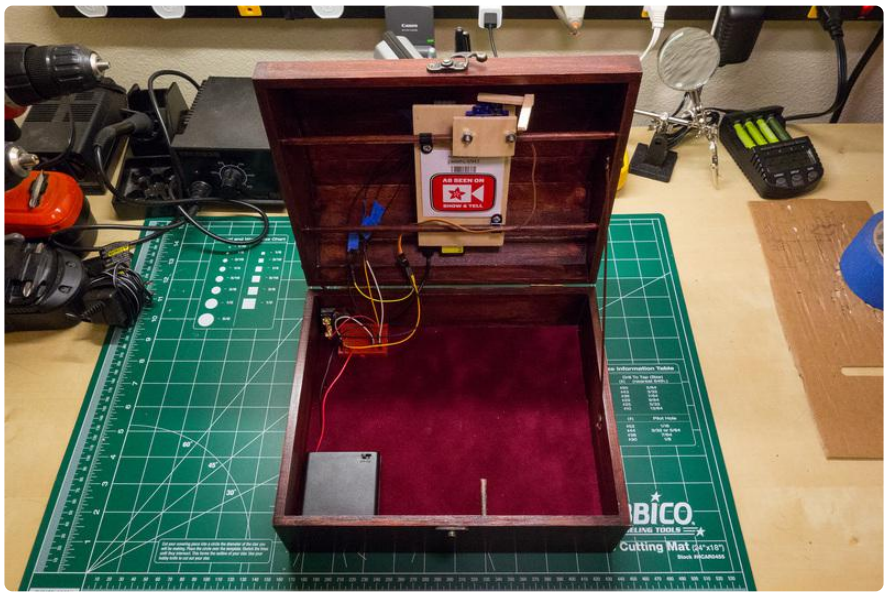
https://learn.adafruit.com/raspberry-pi-face-recognition-treasure-box

Last updated on 2021-11-15 06:08:39 PM EST

Table of Contents

Overview

Face recognition is an exciting field of computer vision with many possible applications to hardware and devices. Using embedded platforms like the Raspberry Pi and open source computer vision libraries like OpenCV (https://adafru.it/cPk), you can now add face recognition to your own maker projects! In this project I'll show you how to build a treasure box which unlocks itself using face recognition running on a Raspberry Pi.



Before you get started with this project, it will help to familiarize yourself with a few things:

- If you haven't setup or used a Raspberry Pi, go through the tutorials on learning the Raspberry Pi (https://adafru.it/aWr) like preparing an SD card (https://adafru.it/aWq), network setup (https://adafru.it/aUB), and using SSH (https://adafru.it/aWc) before attempting this project. Also take a look at the lesson on using a servo (https://adafru.it/aWJ) for more background on how servos work.
- Skim the OpenCV tutorial on face recognition (https://adafru.it/d6h) for an idea of how face recognition algorithms work. Don't worry, you don't need to fully understand the math or code to build and learn from this project.

Continue on to learn about the hardware needed to build this project.

Hardware

You will need the following parts for this project:

- Raspberry Pi, either model A (http://adafru.it/1344) or model B (http://adafru.it/998) will work, running the Raspbian (https://adafru.it/d6i) operating system.
    - Your Pi will need access to the internet to setup the software, so make sure you have either a wired or wireless (http://adafru.it/814) network connection setup with your Pi.
- Raspberry Pi camera (http://adafru.it/1367).
    - Depending on where your camera and Raspberry Pi can be placed inside your box, you might need a longer (http://adafru.it/1648) or shorter (http://adafru.it/1645) camera cable.
- Small box that can fit the Raspberry Pi and locking mechanism inside. I found an inexpensive plain wooden box at a craft store, and finished it with wood stain and polyurethane.
    - Look for a box that has hinges which are screwed in from the outside of the box. Although not terribly secure, it will allow you to disassemble and open the box in case the locking mechanism fails to open.
- Small servo or lock solenoid for the locking mechanism, depending on how your box can be latched shut.
    - A servo (http://adafru.it/169) that rotates a latch can work with most boxes that open from the top or side.
    - A lock solenoid (http://adafru.it/1512) can work with boxes that have a door or drawer. See this locking drawer project (https://adafru.it/d6j) for information on using a lock solenoid. Note: the software for this project is written to use a servo as the locking mechanism, so if you use a lock solenoid you will need to modify the software to actuate the lock with the solenoid instead of the servo.
- Momentary push button (http://adafru.it/1505) that can mount to the box. Depending on how thick your box is, you might need a smaller or larger push button.
- 10 kilo-ohm 1/4 watt resistor to use as a pull-up resistor with the push button.
- Power supply for the Raspberry Pi (http://adafru.it/501) and servo or solenoid. For powering a micro servo, a 4x AA battery pack (http://adafru.it/830) is a simple option.
- Wood, wood glue, and fasteners for building a latch mechanism and frame to support the Pi inside the box. The exact material will depend on your box, but

you can see further below how I used 1/4" dowel and thin bass wood to build the frame and latching mechanism for my box.

- Hookup wires (http://adafru.it/1311) to connect the switch, servo, and servo power supply. Female hookup wires (http://adafru.it/824) work well for connecting directly to the Raspberry Pi GPIO pins, or you could use the Pi cobbler (http://adafru.it/914) with a small breadboard (http://adafru.it/64).

Note: This project is not meant to be a highly secure container. Face recognition is not perfect and can easily be circumvented with a photograph. Don't store valuables in the box!

Assembly

The box you use for your project will dictate exactly how the Raspberry Pi, servo, and latching mechanism need to be mounted. Read the notes below for tips on how to construct your hardware, based on how I built mine:

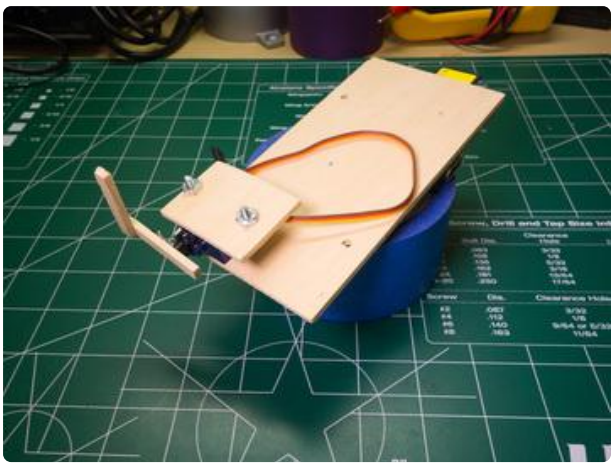
Use a box which can fit the Raspberry Pi, servo, and latching mechanism.

Drill a hole in the top or side for the Raspberry Pi camera. I found a 7/16" drill bit was enough for the camera to comfortably fit. Be careful when drilling large holes--work up from smaller bits so you don't crack the box. If you do damage the box, you can generally use wood filler to hide mistakes.

If they aren't mounted there already, consider moving the hinges to be mounted from the outside of the box. This will allow you to disassemble the box in case the hardware fails in a locked position.

Drill a hole in the box to mount the push button. You can see I mounted mine on the back of the box where it can be reached while a user looks into the camera on the top.

Drill another hole to allow power cables to come into the box for both the Raspberry Pi and servo. I found a 1/2" hole was enough to fit a micro USB cable through for powering the Pi. To the left, you can see the hole I drilled in my box next to the push button.



For the latching mechanism, mount a small dowel perpendicular to the side of the box. Attach a right angle of wood to a servo horn so it can act as a latch that swings down to catch the dowel, locking the box. See the pictures to the left to see the latch mechanism and dowel I built in my box.

If possible, mount the Raspberry Pi in the top of the box or near the hole for the camera. You can see how I mounted my Pi to a small board along with the locking servo, and then attached that board to dowels glued in to form a frame in the box top.

A few screws and a smaller board can act as a clamp to hold the servo in place.

Wiring

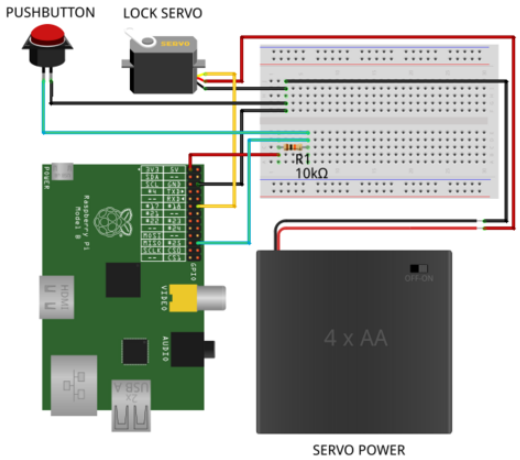
The electronics in this project are fairly simple and involve connecting a servo and push button to the Rasperry Pi. If you have never used these devices with a Raspberry Pi, read the following tutorials for a good overview of their usage:

- Raspberry Pi Lesson 4: GPIO Setup (https://adafru.it/aTH)
- Raspberry Pi Lesson 8: Using a Servo Motor (https://adafru.it/aWJ)
- Bread Board Setup for Input Buttons (https://adafru.it/d6k)

For the servo, connect the signal line to GPIO 18 of the Raspberry Pi. To power the servo I connected a 4x AA battery pack as a power source--connecting the servo to the Pi's 5 volt output could cause problems from noise or excessive current drawn by the servo.

The push button is attached to GPIO 25 of the Raspberry Pi, with a 10 kilo-ohm pull-up resistor to 3.3 volt power from the Pi.

See the diagram below for how to wire the push button and servo to the Raspberry Pi.



Continue on to learn about how the dependencies and software set up for the project.

Software

Note this project was created in 2013 and the Raspberry Pi operating system has since been updated to not be compatible with the RPIO library mentioned below. Use an older version of Raspbian like a wheezy release from 2013 from: http://downloads.raspberrypi.org/raspbian/images/

OpenCV Installation

This project depends on the OpenCV (https://adafru.it/cPk) computer vision library to perform the face detection and recognition. Unfortunately the current binary version of OpenCV available to install in the Raspbian operating system through apt-get (version 2.3.x) is too old to contain the face recognition algorithms (https://adafru.it/d6l) used by this project. However you can download, compile, and install a later version of OpenCV to access the face recognition algorithms.

Note: Compiling OpenCV on the Raspberry Pi will take about 5 hours of mostly unattended time. Make sure you have some time to start the process before proceeding.

First you will need to install OpenCV dependencies before you can compile the code. Connect to your Raspberry Pi in a terminal session and execute the following command:

```
sudo apt-get update
sudo apt-get install build-essential cmake pkg-config python-dev libgtk2.0-dev
libgtk2.0 zlib1g-dev libpng-dev libjpeg-dev libtiff-dev libjasper-dev libavcodec-dev
swig unzip
```

Answer yes to any questions about proceeding and wait for the libraries and dependencies to be installed. You can ignore messages about packages which are already installed.

Next you should download and unpack the OpenCV source code by executing the following commands:

```
wget http://downloads.sourceforge.net/project/opencvlibrary/opencv-unix/2.4.9/
opencv-2.4.9.zip
unzip opencv-2.4.9.zip
```

Note that this project was written using OpenCV 2.4.7, although any 2.4.x version of OpenCV should have the necessary face recognition algorithms.

Now change to the directory with the OpenCV source and execute the following cmake command to build the makefile for the project. Note that some of the parameters passed in to the cmake command will disable compiling performance tests and GPU accelerated algorithms in OpenCV. I found removing these from the OpenCV build was necessary to help reduce the compilation time, and successfully compile the project with the low memory available to the Raspberry Pi.

```
cd opencv-2.4.9
cmake -DCMAKE_BUILD_TYPE=RELEASE -DCMAKE_INSTALL_PREFIX=/usr/local -
DBUILD_PERF_TESTS=OFF -DBUILD_opencv_gpu=OFF -DBUILD_opencv_ocl=OFF
```

After this command executes you should see details about the build environment and finally a '-- Build files have been written to: ...' message. You might see a warning that the source directory is the same as the binary directory--this warning can be ignored (most cmake projects build inside a subdirectory of the source, but for some reason I couldn't get this to work with OpenCV and built it inside the source directory instead). If you see any other error or warning, make sure the dependencies above were installed and try executing the cmake command again.

Next, compile the project by executing:

```
make
```

This process will take a significant amount of time (about 5 hours), but you can leave it unattended as the code compiles.

Finally, once compilation is complete you can install the compiled OpenCV libraries by executing:

```
sudo make install
```

After this step the latest version of OpenCV should be installed on your Raspberry Pi.

Python Dependencies

The code for this project is written in python and has a few dependencies that must be installed. Once connected to your Raspberry Pi in a terminal session, execute the

following commands:

sudo apt-get install python-pip
sudo apt-get install python-dev
sudo pip install picamera
sudo pip install rpio

You can ignore any messages about packages which are already installed or up to date. These commands will install the picamera library (https://adafru.it/d6n) for access to the Raspberry Pi camera, and the RPIO library (https://adafru.it/d6o) for access to the Pi GPIO pins and PWM support (for better servo control with the Pi).

Software Install

After OpenCV and the python dependencies have been installed, download the software for this project from the link below:

Download Code

https://adafru.it/d6p

Copy the archive to your Raspberry Pi and extract it to a folder.

Continue on to learn about training the face recognition algorithm to recognize your face.

Training

This project uses the eigenfaces algorithm in OpenCV (https://adafru.it/d6h) to perform face recognition. To use this algorithm you'll need to create a set of training data with pictures of faces that are and are not allowed to open the box.

Included in the project is a large set of face images that are tuned for training with the face recognition algorithm. This is the database of faces (https://adafru.it/d6q) published by research AT&T Laboratories Cambridge in the mid 90's. These faces make up the set of negative images which represent faces that are not allowed to open the box. You can see these images in the training/negative subdirectory of the project.

To generate images of the person who will be allowed to open the box, or positive

training images, you can use the included capture-positives.py script. This script will take pictures with the box hardware and write them to the training/positive sub-directory (which will be created by the script if it does not exist).

With the box hardware assembled and powered up (servo power is not necessary for this step), connect to the Raspberry Pi in a terminal session and navigate to the directory with the project software. Execute the following command to run the capture positive script:

```
sudo python capture-positives.py
```

After waiting a few moments for the script to load, you should see the following instructions:

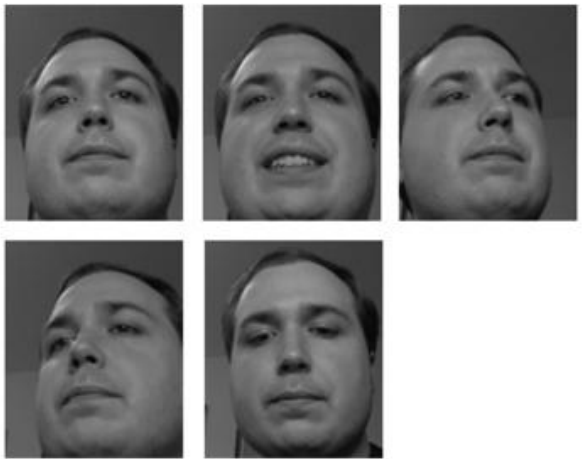
```
Capturing positive training images.
Press button or type c (and press enter) to capture an image.
Press Ctrl-C to quit.
```

The script is now ready to capture images for training. If you see an error message, make sure OpenCV and the software dependencies from the previous step (https://adafru.it/d6r) are installed and try again.

Point the box's camera at your face and press the button on the box (or send the letter 'c' in the terminal session) to take a picture. If the script detects a single face, it will crop and save the image in the positive training images sub-directory. If the script can't detect a face or detects multiple faces, an error message will be displayed.

If see error messages about not detecting a single face, you can examine the full picture that was captured by the camera to understand if it's picking up your face (or if other faces might be in view--be careful, even photos or posters with faces behind you can be detected!). Open the file capture.pgm (located in the directory with the scripts) in an image editor to see the last image that was captured. It's easiest to use a tool like Cyberduck (https://adafru.it/d6s) to view the files on your Raspberry Pi over SFTP (SSH file transfer protocol).

Using the capture-positives.py script, capture some pictures of the face that is allowed to open the box. Try to take pictures of the face with different expressions, under different lighting conditions, and at different angles. You can see the images I captured for my positive training data below:



Training images I captured for my face. Try to get pictures with different expressions, in different light, etc. to build a better face recognition model.

Once you've captured a number of positive training images, you can run the train.py script to perform the training. If it's still running, exit the capture-positives.py script by pressing Ctrl-C in the terminal session. Run the following python script to train the face recognition model:

python train.py

Note that this command does not need to be run as root with sudo because none of the Pi GPIO/hardware is accessed.

You should see a message that the training images were loaded, and the text 'Trainin g model...' to indicate the training calculations are being performed. Training the face recognition model on the Pi will take about 10 minutes.

Once the training is complete you should see the message 'Training data saved to training.xml'. The training data is now stored in the file training.xml, which will be loaded by the box software to configure the face recognition model.

If you're curious you can examine images that are output by the training to visualize the eigenfaces of the model. Open the mean.png, positive_eigenface.png, and negat ive_eigenface.png files in an image viewer. You don't need to do anything with these images, but can learn more about their meaning from the OpenCV website (https:// adafru.it/d6l), Wikipedia (https://adafru.it/d6t), or the original research paper on eigenfaces (https://adafru.it/d6u).

Mean Image

Positive Eigenface Image

Negative Eigenface Image

Eigenfaces generated from my training data.

The positive eigenface is a summary of the features that differentiate my face from the average or mean face. This face will be allowed to unlock the box.

The negative eigenface represents all the other faces in the training set. This face will not be allowed to unlock the box.

With the training.xml file generated, you're ready to move on to configure the servo latch and software for the project.

Configuration

Servo Configuration

In this step you'll determine the pulse width values for the locked and unlocked servo positions of the box. Make sure the hardware is assembled, the software dependencies are installed, and power is applied to both the Raspberry Pi and servo before you continue. Connect to the Raspberry Pi in a terminal session and type the following to run an interactive python shell as root:

sudo python

At the python shell type in the following to load the RPIO library and its servo control class (note you don't need to type the >>>, it's just a reference for what you should see):

```
>>> from RPIO import PWM
>>> servo = PWM.Servo()
```

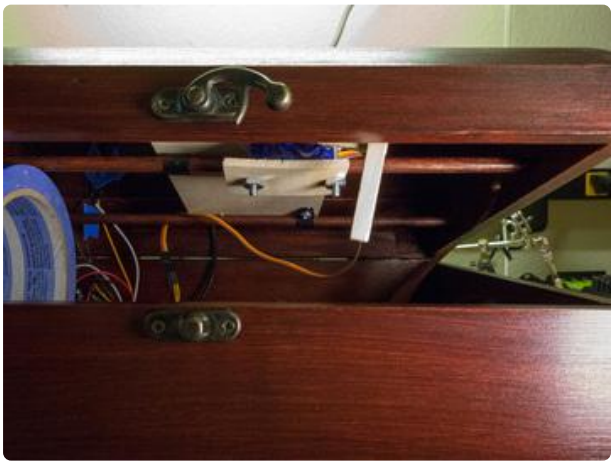
Now you can call the set_servo function to send pulse width values to the servo which will move it to different positions. The pulse width value you send can range from 1000 to 2000 microseconds, with a value of 1500 being the center position of the servo. If you're curious, you can find more information on servos in this tutorial (https://adafru.it/aWJ).

Make sure the box is open so you can see the servo and it's free to move, then execute this command to move the servo to the center position (pulse width of 1500 microseconds):

```
>>> servo.set_servo(18, 1500)
```

Note that if your servo is connected to a different GPIO port of the Raspberry Pi, change the first parameter from 18 to the appropriate value.

Try sending different pulse width values between 1000 and 2000 to find the values which move the latch to the locked and unlocked position. Don't forget you can adjust the position of the servo horn and latch on the servo if necessary. You can see the values I found for my hardware below:



Latch in the unlocked position with a servo pulse width of 2000 micro seconds.

Latch in the locked position with a servo pulse width of 1100 micro seconds.

Take note of the pulse width values you determined for the locked and unlocked position as they will be used later in the configuration. Execute the following command in the python shell to close it:

```
>>> quit()
```

Software Configuration

In the directory with code for this project, open the file config.py in a text editor. This file defines the configuration for the project and might require a few changes for your hardware. Specifically you can adjust the following values:

- LOCK_SERVO_PIN - this should be the GPIO number of the Raspberry Pi which is connected to the signal line of the lock servo. This project is built to use GPIO 18 by default.
- LOCK_SERVO_UNLOCKED - this should be the pulse width value (in microseconds) for the unlocked position of the latch. Use the value you determined in the previous step.
- LOCK_SERVO_LOCKED - this should be the pulse width value for the locked position of the latch.
- BUTTON_PIN - this should be the GPIO number which is connected to the push button on the box. This project is built to use GPIO 25 by default.

The rest of the values in the configuration can be ignored. The only other value of interest is the POSITIVE_THRESHOLD configuration, which gives the boundary for how confident the face recognition has to be before it's considered a positive match. For now the default value of 2000 should suffice, but it can be adjusted later if the box has too many false positive or false negative recognitions.

Continue on to run the box software and finish the project.

Usage

To run the box software, make sure the hardware is assembled and the training (https://adafru.it/d6v) is complete. First you'll run the box with only power to the Raspberry Pi and not the lock servo--this will allow you to test the face recognition without locking yourself out of the box. Apply power to just the Raspberry Pi (and not the servo), then connect to the Pi in a terminal session, navigate to the folder with the software, and execute the following command:

```
sudo python box.py
```

You should see the box software load and the message 'Loading training data...' appear. It should take a few minutes for the training data to load and the following message to appear:

```
Running box...
Press button to lock (if unlocked), or unlock if the correct face is detected.
```

Press Ctrl-C to quit.

The box will lock itself (however because no power is applied to the servo, the lock latch should not move) and be ready to try unlocking with face recognition. Aim the box camera at your face, like you did when training the face recognition model, and press the button on the box. You should see a message that the button was pressed, and after a moment if the face is recognized you will see a message like:

Predicted POSITIVE face with confidence 1321.35253959 (lower is more confident). R ecognized face!

If the face is not recognized you will see a message like:

Predicted NEGATIVE face with confidence 3987.76625152 (lower is more confident). D id not recognize face!

If a single face couldn't be detected (because none is visible, or there are multiple faces detected) an error message will be displayed like with the training script.

When a face is found, you'll notice the prediction message includes how the face was recognized (i.e. matching either the positive or negative training data), and the confidence of the recognition. The confidence is a value like 1321.35 or 3987.77 above and represents the distance of the captured face from the predicted face--a lower distance value means the captured face is more similar to the predicted face.

To unlock the box, a captured face must be a positive prediction with a confidence value below the POSITIVE_THRESHOLD configuration value. By default the POSITIVE _THRESHOLD is 2000, so the positive prediction with confidence 1321.35 was considered a match and would have unlocked the box.

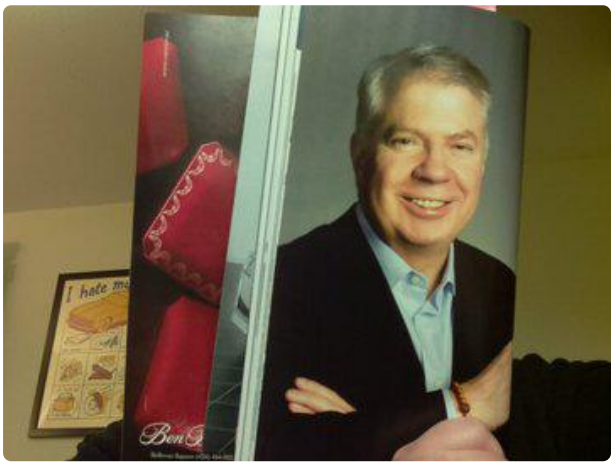
When the box is in an unlocked state, press the button to lock the box again. You do not need to have a face recognized to lock the box.

Play with unlocking and locking the box to see what prediction and confidence you get from your trained face recognition model. Ideally you want images of the positively trained face to be predicted as positive images with a low number for the confidence value. The eigenfaces recognition algorithm is sensitive to the lighting and orientation of your face, so try to have as similar conditions as when the positive training images were captured (or consider re-training your model to include more varied images).

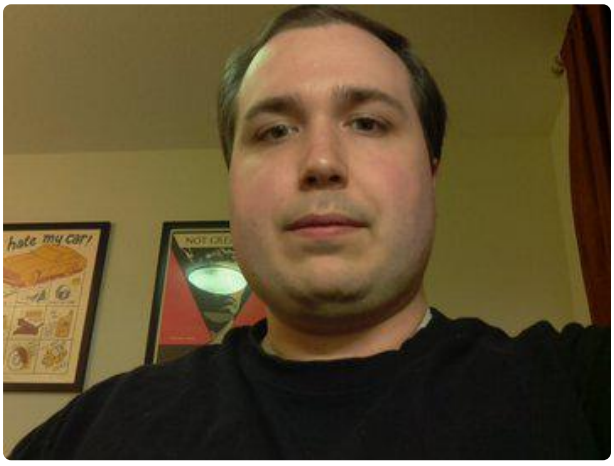
You can see examples of how my training data predicted faces below (remember you

can look at the capture.pgm image in the software directory after pressing the button to attempt a face recognition):

This image unlocked my box because it was a match for the positive training images, and had a confidence of 1321.35 which was below the threshold of 2000.

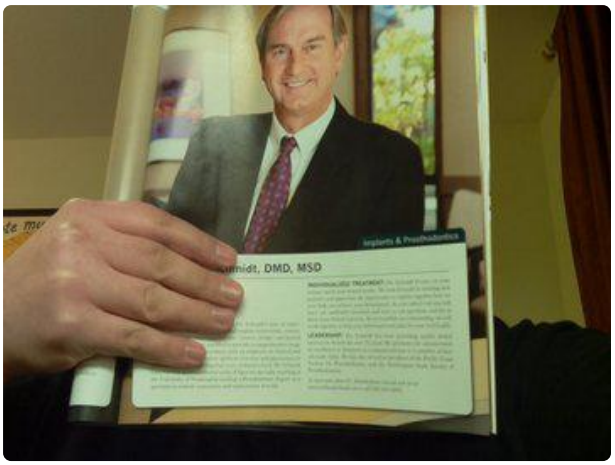


This image of a face from a magazine did not unlock my box because it was a match for the negative training data, with a confidence of 3987.77.



This image unfortunately did not unlock my box. Although it was predicted as a positive match, the confidence was 2269.04 which was slightly higher than the positive match threshold.

If you get a lot of false negative predictions like this, it's a sign you need better training data or should adjust the POSITIVE_THRESHOLD configuration value up.



This image matched the positive training data, but the confidence value of 4407.85 was too far above the threshold to unlock the box.

If you get a lot of false positive responses like this it's also a sign that you need better training data. Remember face recognition is far from perfect--there's always a chance someone who looks like the positively trained images will match close enough to unlock the box!

Based on how the face recognition responds, you might consider training it again or adjust the POSITIVE_THRESHOLD configuration to a higher/lower value.

When you're happy with how the recognition is working, apply power to the servo and try locking and unlocking the box for real. Congratulations on building the face recognition treasure box!

If for some reason your box is locked and will not recognize a face to unlock itself, you can manually set the servo position to an unlocked value by following the steps in the servo configuration (https://adafru.it/d6w). If you don't have access to the Raspberry Pi terminal or the servo isn't moving, unfortunately you'll need to disassemble your box (this is why it's a good idea to use a box with hinges on the outside!).

Future Work

This project is a great example of how to use the Raspberry Pi and Pi camera with OpenCV's computer vision algorithms. By compiling the latest version of OpenCV, you can get access to the latest and most interesting computer vision algorithms like face recognition. You can consider extending this project in other interesting ways, such as:

- Adding LED's to the box which flash when a face is detected and recognized. This feedback would help you train and use the box without having to connect to the Pi in a terminal session.
- Make the main box.py script start up automatically when the Raspberry Pi powers on. Look at this blog post (https://adafru.it/d6x) for more information on the various ways of automatically running a script at start up.

- Investigate using other face recognition algorithms in OpenCV. See the OpenCV tutorial on face recognition (https://adafru.it/d6l) for more information. Algorithms such as Fisherfaces might be more robust to varied lighting or other conditions.
- Have the box send you an email with a picture of whoever is attempting to open it.
- Add a microphone and look at adding speech recognition (https://adafru.it/d6y) as another means of unlocking the box.

Can you think of other ways to extend the project?