

Maskit: COVID-19 Resiliency through Computer Vision and Robotics

Allen Mao
University of California, Berkeley
Berkeley, California
allenmao@berkeley.edu

Hans Gundlach
University of California, Berkeley
Berkeley, California
11235hans@berkeley.edu

CONTENTS

I	Introduction	2
I-A	Code Source	2
II	Purpose & Motivation	2
II-A	Meet the Team	2
II-B	Previous Work on Mask Identification with Computer Vision	3
II-C	Our Solution	3
III	Step by Step Usage Instructions	3
III-A	Raspberry Pi Prerequisites	3
III-A1	Materials	3
III-A2	GPIO configuration	4
III-B	Setting up the server	4
III-C	Running it on R Pi	4
III-C1	Software Setup	4
IV	Difficulties and Challenges	5
V	Market Evaluation	5
VI	Suggested Improvements	6
VII	Conclusion	6
	References	6

I. INTRODUCTION

The COVID-19 pandemic has led to, at the time of this writing, almost half a million deaths, the worst global recession since the Great Depression, and school closures that have affected nearly all of the world's student population. Although reopening efforts have been underway in many parts of the world, their results have been mixed and it is often instead in citizens' own hands to protect themselves from this virus through means such as social distancing, frequent handwashing, and mask wearing. Although the United States Centers for Disease Control and Prevention (CDC) has issued recommendations for citizens to wear masks to reduce exposure to the virus, the choice to wear a mask has unfortunately evolved to become a political question too [1], [2]. Apart from CDC and World Health Organization (WHO) guidelines to wear facial coverings, i.e. masks, in this article we will not discuss the merits of face coverings; this knowledge is assumed. Instead, we propose, describe, and demonstrate *Maskit*, a computer vision and robotics system that keeps business owners and other ordinary citizens safe by blocking people who fail to wear a mask while letting people who do through entrances.

A. Code Source

Our work is found at <https://github.com/hansgundlach/FaceMaskDetectionRasPi>. It is based on earlier an earlier repository by AIZoo Tech¹. We thank AIZoo Tech for releasing their models to the public with the MIT License. Throughout this paper, we detail our original contributions on top of AIZoo Tech's technology to enhance public safety during a worldwide pandemic using computer vision and robotics.

II. PURPOSE & MOTIVATION

A. Meet the Team



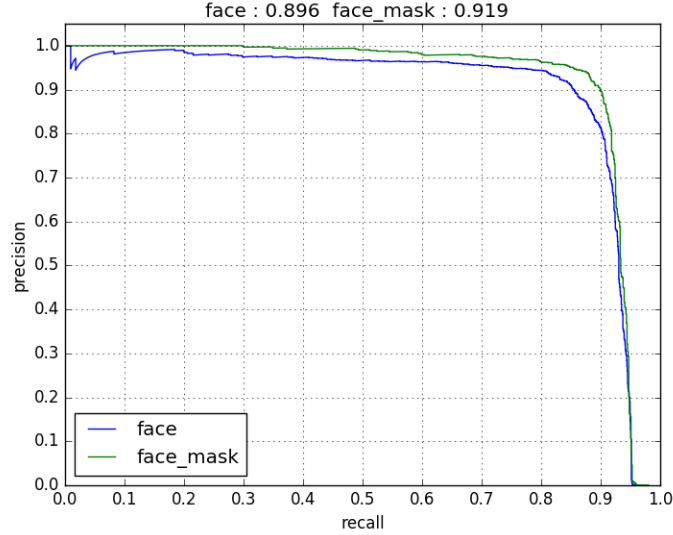
The team: Allen to the left, Hans to the right

We are both undergraduate students at the University of California, Berkeley. Allen is a Computer Science major who is primarily interested in Natural Language Processing

¹<https://github.com/AIZOOTech/FaceMaskDetection>

B. Previous Work on Mask Identification with Computer Vision

In the past several months, several organizations, for example Didi Chuxing² and AI Zoo Tech³ have released open-source pre-trained neural network models on multiple platforms (Keras, Tensorflow, and Caffe) that classify faces within a photo or video as mask-wearing or not. Both models perform spectacularly as the Didi model was trained on a dataset of over 200,000 faces while the AIZOO model consists of 24 convolutional layers trained on approximately 8000 images from WIDER Face and MAFA datasets. Accordingly, the Didi model achieves at least 98% accuracy and the AIZoo model has a 0.896 ROC AUC (*receiver operating characteristic area under the curve*) for face detection and 0.919 ROC AUC for face mask detection, as seen in their curve below.



C. Our Solution

Maskit consists of two parts: a Raspberry Pi and a server. We connect a Pi Camera onto the Raspberry Pi such that the Raspberry Pi would be able to take periodic photos. The Raspberry Pi then sends these images to the server via a POST request. The server then runs the mask detection model on the incoming image and for each face that it identifies, classifies the face as with mask or without mask. These classifications are sent back to the Raspberry Pi as the response to the POST request. After the Raspberry Pi receives these classifications, if and only if all faces identified wear a mask, then it will activate the servo to open access. In our prototype, we model this situation by raising a welcome flag, though this can be generalized to opening store front doors, for example.

III. STEP BY STEP USAGE INSTRUCTIONS

A. Raspberry Pi Prerequisites

1) *Materials:* We used a Raspberry Pi Model B (700-MHz processor, 512 MB RAM). The physical limitations of this Raspberry Pi version are discussed later. In addition to the Raspberry Pi and its regular accessories, i.e. keyboard, power input, etc., we used:

- 1) HiTEC HS-5625MG servo ⁴
- 2) Pi Camera module (for setup instructions, see ⁵

²<https://github.com/didi/maskdetection>

³<https://github.com/AIZOOTech/FaceMaskDetection>

⁴<https://hitecrd.com/products/servos/sport-servos/digital-sport-servos/hs-5625mg/product>

⁵<https://www.raspberrypi.org/documentation/usage/camera/>

- 3) 6 Volt DC input source. In our case, we put 4 AA batteries into a 9 volt battery case and used alligator clips to join the two empty battery terminals.
- 4) a breadboard (optional but makes setup nicer)

Our setup should work on the latest version of Raspberry Pi OS (formerly known as *Raspbian*). We will not go in depth on the Raspberry Pi settings on how to enable the Pi Camera or GPIO interfacing (i.e. `sudo raspi-config`, then enable both GPIO access and camera module), but links are provided in footnotes⁶⁷.

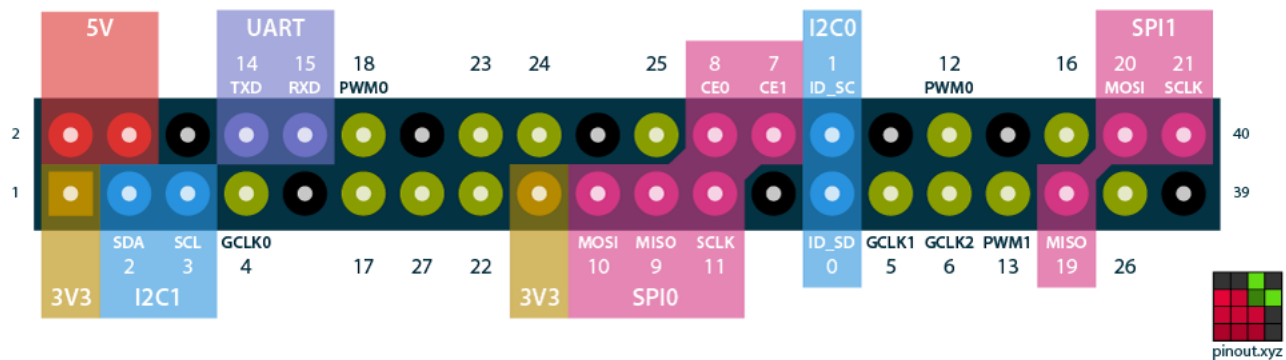
We installed the `picamera` and `wiringpi` with `python pip` for camera and GPIO pin interfacing, respectively, in Python.

In the next section, we describe how to connect the servo to the Raspberry Pi.

2) *GPIO configuration*: We use the Raspberry Pi's GPIO pins to control the servo. However, the servo cannot draw too much current from the Raspberry Pi without breaking it. As a result, the red servo wire must be connected directly to the 6 volt V_{in} . The black servo wire is connected to the ground. Make sure that the Raspberry Pi ground pin is also connected to this ground. The yellow wire is connected to PWM0 (*pulse-width modulation 0*), or physical pin 12⁸.

See below for the Raspberry Pi pinout⁹:

Raspberry Pi GPIO BCM numbering



B. Setting up the server

- 1) Install the dependencies found in the `environment.yml` file¹⁰.
- 2) On terminal, run `conda activate maskdetect` to load the environment.
- 3) Run `python simple_server.py`. The server will be hosted on port 8000.
- 4) The server will now be listening for POST requests.
- 5) Run `hostname -I` to get host IP Address. We will need this value in the next step.

C. Running it on R Pi

1) Software Setup:

```
1 # upload_img_post.py
2 import os
3 import requests
4 import json
5 import wiringpi
```

⁶<https://www.raspberrypi.org/documentation/configuration/camera.md>

⁷<https://pimylifeup.com/raspberry-pi-gpio/>

⁸See <https://pinout.xyz/> for Raspberry Pi pin information

⁹Diagram by <https://pinout.xyz/>

¹⁰<https://github.com/hansgundlach/FaceMaskDetectionRasPi/blob/master/environment.yml>

```

6 from io import BytesIO
7 from time import sleep
8 from picamera import PiCamera
9 url = 'http://192.168.0.31:8000' # replace with actual IP address
10 wiringpi.wiringPiSetupGpio()
11 wiringpi.pinMode(18, wiringpi.GPIO.PWM_OUTPUT)
12 wiringpi.pwmSetMode(wiringpi.GPIO.PWM_MODE_MS)
13 wiringpi.pwmSetClock(192)
14 wiringpi.pwmSetRange(2000)
15 delay_period= .001
16
17
18 stream = BytesIO()
19 camera = PiCamera()
20 camera.start_preview()
21 sleep(2)
22 for i in list(range(2)):
23     camera.capture(stream, 'jpeg')
24     stream.seek(0)
25     data = stream.read()
26     r = requests.post(url, data=data)
27     lst_str = r.content.decode('utf-8')
28     lst = json.loads(lst_str)
29     all_masked = sum([x[0] for x in lst])
30     if (all_masked == 0):
31         for pulse in range(50, 250, 1):
32             wiringpi.pwmWrite(18, pulse)
33             sleep(delay_period)
34         sleep(3)
35         for pulse in range(250, 50, -1):
36             wiringpi.pwmWrite(18, pulse)
37             sleep(delay_period)
38
39     stream = BytesIO()
40     sleep(5)

```

The raspberry pi is

IV. DIFFICULTIES AND CHALLENGES

We faced many challenges during the completion of the project. We tried to do all the image processing on the raspberry pi however the pi, unfortunately, did not have either the necessary speed or space to handle the complicated image processing in OpenCV and Tensorflow. Therefore, we had to design an HTTP server using python to send the image to a Linux computer where the image is processed. Finding the right model for mask detection was also difficult. We tried several different frameworks before finding an efficient Tensorflow mask detection model from AIZOO. Remote work also proved difficult especially for a hardware project where only one of us could see the immediate results.

V. MARKET EVALUATION

MaskIt has several market application. MaskIt could be used as an artificial bouncing mechanism to ensure that people use masks in public places. Large scale mask detection could also be useful in epidemiological and policy research.

VI. SUGGESTED IMPROVEMENTS

MaskIt needs to be fully evaluated in larger public situations. If it is going to be useful in public places

VII. CONCLUSION

In conclusion, MaskIT has significant potential in research and commercial settings during the COVID pandemic. Given the lack of current data on mask usage in practice and the importance of mask in the current public health it is important to create automated systems to track account of mask usage. Some building may need automated gate systems to scan mask usage while universities will need to monitor the percentage of mask compliance on campuses. Systems like MaskIT will be of increasing public importance in the coming months.

REFERENCES

- [1] John Brandon. A doctor explains why 45 protective mask, May 2020.
- [2] Neil Vigdor. American airlines bans conservative activist who refused to wear a mask, Jun 2020.