```
# AVANTI3D Bed Adhesion Failure Detection Module Code v2.3
# READ “SUMMARIES” IN COMMENTS TO GET GIST OF CODE

# Run on Raspberry Pi Zero W with OpenCV installed (can be tested on PC if gpio
sections are commented out)
# Make launcher script for Cron to launch python script at launch
import cv2  # will require OpenCV installation on given device
import time
import numpy as np
import math  # to allow more complex math in future
import smtplib  # for graphing
import RPi.GPIO as GPIO  # pin 35

cap = cv2.VideoCapture(0)  # Direct to camera port

# LED PWM SETUP GPIO on Raspberry Pi
GPIO.setmode(GPIO.BOARD)
GPIO.setup(20, GPIO.OUT)
p = GPIO.PWM(20, 1000)  # pin 35, frequency 100/s
dutyCycle = 0
p.start(dutyCycle);
lightingConstant = 100*640*480  # resolution times desired average Value (in HSV)

###LED CALIBRATION###
# Summary: find the value when the LEDs are 0% and 100% on to find the
d(Value)/d(dutyCycle)
# this is the slope used to adjust the lighting
# if the initial dutyZero calibration total is more than the initial Lighting
Constant, the Lighting Constant is changed accordingly

# Part 1
p.ChangeDutyCycle(0)  # Find Total Value when there's no LED
_, temp = cap.read()  # Take Picture
temp = cv2.cvtColor(temp, cv2.COLOR_BGR2HSV)  # Convert to HSV
dutyZero = 0
for x in range(0, 479, 1):  # for every row
   for y in range(0, 639, 1):  # for every column
       dutyZero = dutyZero + temp[x, y][2]  # Add values
if dutyZero > lightingConstant:  # change lightingConstant if necessary
   lightingConstant = dutyZero  # dutyZero is y-intercept
# Part 2
p.ChangeDutyCycle(100)
time.sleep(0.25)  # wait to adjust
_, temp = cap.read()  # Take Picture
temp = cv2.cvtColor(temp, cv2.COLOR_BGR2HSV)  # Convert to HSV
dutyHundred = 0
for x in range(0, 479, 1):  # for every row
   for y in range(0, 639, 1):  # for every column
       dutyHundred = dutyHundred + temp[x, y][2]  # Add values
lightingDerivative = (dutyHundred-dutyZero)/100  # calculating slope
```

```
###LED ADJUSTER###
# Summary: the function uses the slope from calibration to adjust accordingly

def LEDadjuster():
   global dutyCycle
   _, temp = cap.read()  # Take Picture
   temp = cv2.cvtColor(temp, cv2.COLOR_BGR2HSV)  # Convert to HSCV
   total = 0
   for x in range(0, 479,1):  # for every row
       for y in range(0, 639,1):  # for every column
           total = total + temp[x, y][2]  # Add values
   g = dutyCycle + ((lightingConstant-total)/lightingDerivative)  # math using
derivativeLighting to adjust
   if g > -100 & g < 100:  # ensuring dutyCycle is within feasible range
       dutyCycle = g
   else:
       if g > 100:
           dutyCycle = 100
       if g < -100:
           dutyCycle = -100
   p.ChangeDutyCycle(dutyCycle)  # execute changes
   time.sleep(0.2)

# Extruder Detection Functions
def xTop():
   pos = 0
   j = 0
   for j in range(0,479):
       for k in range(0,639):
           if current_extruder[j, k] == 255:
               pos = 1
       if pos:
           break
   return j

def nothing(x):
   pass
# VARIABLES
userName = "Avanti3D"
global current
global prev
iterations = 0
kernel1 = np.ones((3,3),np.uint8)

deltaT = 2  # time between shots
compareConstant = 0.97  # lowest percent of similarity before failure is assumed
Canny_X = 130
Canny_Y = 130
```

```
x_top = [0]  # list expands as database of extruder movement, can be used later to
detect print start, stop, and clogs
x_crop = [0,479]  # 0,479
y_crop = [0, 629]  # 0,639
extruderRange = np.array([[20, 80, 100], [180, 250, 240]])  # For HSV Thresholding

# Email Function
def email():
   gmail_user = 'info.avanti3d@gmail.com'  # Email information will have to be edited
with an app
   gmail_password = 'MITlaunch@team6'
   sent_from = gmail_user
   to = ['info.avanti3d@gmail.com']
   subject = 'PRINT FAILURE'
   body = userName + ", your print may have failed, we suggest checking on it."
   email_text = userName + ", your print may have failed, we suggest checking on it."

   sent_from, ", ".join(to), subject, body

   try:
       server = smtplib.SMTP_SSL('smtp.gmail.com', 465)
       server.ehlo()
       server.login(gmail_user, gmail_password)
       server.sendmail(sent_from, to, email_text)
       server.close()

       print 'Email sent!'
       return 1
   except:
       print 'Something went wrong...'
# Email commented out

# Pixel-by-Pixel Compare Function
# Summary: function compares mats pixel-by-pixel using loops and returns similarity
def compare(xtop, xbottom, ytop, ybottom):  # two images, and the places to check for
similarities
   totalPixels = 1  # to avoid divide by zero error, error is negligible
   matchPixels = 0.0

   for x in range(xtop, xbottom,3):  # for every row
       for y in range(ytop, ybottom,3):  # for every column
           value = prev[x, y]
           value2 = current[x, y]
           if np.all(value != value2):
               matchPixels = matchPixels + 1  # check for matching pixels
           totalPixels = totalPixels + 1
   print (matchPixels/totalPixels)*100, "% similar"
   return (matchPixels/totalPixels)  # returns float between zero and one

#START
```

```
_, current = cap.read()
prev = cv2.dilate(current, kernel1, 2)  # prev is a more dilated

# THRESHOLDING TOOL
"""cv2.namedWindow("window")
cv2.createTrackbar('Hmax', 'window', 1, 255, nothing) # noting() created to fill argument
cv2.createTrackbar('Hmin', 'window', 0, 255, nothing)
cv2.createTrackbar('Smax', 'window', 1, 255, nothing)
cv2.createTrackbar('Smin', 'window', 0, 255, nothing)
cv2.createTrackbar('Vmax', 'window', 1, 255, nothing)
cv2.createTrackbar('Vmin', 'window', 0, 255, nothing)
cv2.createTrackbar('CannyX', 'window', 10, 300, nothing)
cv2.createTrackbar('CannyY', 'window', 10, 300, nothing)"""

###MAIN LOOP###
# Summary: 1) camera adjusts LEDs to consistent average value
#2) Find edges in shot and dilate to increase mass
# 3) Threshold to find neon green sticker on extruder and use to find height of extruder in pixels
#4) Don't compare previous and current pictures above the height of extruder
# 5) The pixel-by-pixel comparison with return how much the edges have shifted since the previous frame,
# and the extruder detection helps ensure that the extruder movement while printing doesn't interfere with the comparison
# Conclusion: If the object on the print bed shifts relative to the camera module, then bed adhesion has failed the print, the shift should be detected with the edge comparison
while True:  # format for streaming video
    # LED LIGHT ADJUSTING
    LEDadjuster()
    # Uncomment to adjust Canny and Threshholding
    """extruderRange[1][0] = cv2.getTrackbarPos('Hmax', 'window')
    extruderRange[0][0] = cv2.getTrackbarPos('Hmin', 'window')
    extruderRange[1][1] = cv2.getTrackbarPos('Smax', 'window')
    extruderRange[0][1] = cv2.getTrackbarPos('Smin', 'window')
    extruderRange[1][2] = cv2.getTrackbarPos('Vmax', 'window')
    extruderRange[0][2] = cv2.getTrackbarPos('Vmin', 'window')
    Canny_X = cv2.getTrackbarPos('CannyX', 'window')
    Canny_Y = cv2.getTrackbarPos('CannyY', 'window')"""
    # START
    time.sleep(deltaT)  # delay t seconds
    iterations = iterations + 1
    #  Take Picture
    _, current = cap.read()
    # Canny and Edges Blurred
    current_edges = cv2.Canny(current, Canny_X, Canny_Y)  # Find edges
    current_edges = cv2.dilate(current_edges, kernel1)
    cv2.imshow('edges2', current_edges)
    # Extruder Detection
    current_extruder = cv2.cvtColor(current, cv2.COLOR_BGR2HSV)
# Converting Image Type
    current_extruder = cv2.inRange(current, extruderRange[0], extruderRange[1])
# Threshhold uses numpy arrays as arguments, see above
    current_extruder = cv2.erode(current_extruder, kernel1)
# Erodes smaller pixel masses to avoid false height readings
    cv2.imshow('extruder', current_extruder)
    # Find the x_top from the extruder mat
    x_top.append(xTop()) # Add next x_top value
    print x_top[iterations], "= extruder location"

    # Compare current and prev mats
    if compare(x_crop[0], x_top[iterations-1],y_crop[0],y_crop[1]) < compareConstant:
# Compare() use extruder height in x_top[] as cropping argument
        email()  # email user about bed adhesion problem
    # RESET PREV
    prev = current_edges
    # Esc key to quit
    if cv2.waitKey(1) == 27:
        break

cap.release()  # Release camera when program exits loop
```