```
# -*- coding: utf-8 -*-
"""
Created on Fri Feb 14 13:49:34 2020

For processing video data of moving pendula marked with yellow paint,
and extracting information about this motion.

@author: Scott
"""

#Import scipy for graphing, Image for image processing, and cv2 for
video->frame
from PIL import Image
import cv2
import matplotlib.pyplot as plt
import csv

#Set parameters:
vidname = "C:\\Users\\Scott\\Desktop\\CIMG0359.avi" #input video file
tempimgname = "C:\\Users\\Scott\\Desktop\\TempIMG.jpg" #location to store
images during processing
csvname = "C:\\Users\\Scott\\Desktop\\PendLocs.csv" #output location for
CSV file
threshold = 380 #R+G-B value to set yellow pixels; suggest 300-400
depending on light conditions. Test with writeBW
nummasses = 5 #Currently number of masses is hardcoded, and any extra
masses/masses moving out of frame will mess things up
masswidth = 60 #Approximate width of painted stripes in pixels (only
adjust if camera distance changes dramatically)
reportframes = 0 #Gives printout when this many frames are complete. Set
to 0 to disable printout.
minheight = 600 #Pixel value above which masses will not move; set
properly it increses run time and removes errors. Set to 0 to use entire
image.

#The list of center points for each mass, in nested array from,
initialized with no values.
centerpoints=[]
for i in range(nummasses):
    centerpoints.append([])

#A simple function to check if a given pixel meets the threshold (defined
above) for being "yellow"
#Returns True if the pixel is yellow, false otherwise; criteria is r + g
- b
def isYellow(pixelvals):
    if pixelvals[0]+pixelvals[1]-pixelvals[2]>threshold:
        return True
    else:
        return False

# A function for testing, saving a processed image. Good for indicating
if the threshold for
# isYellow() is working properly.
```

```
def writeBW(outputlocation, inputlocation):
    imageToConvert = Image.open(inputlocation)
    bw = Image.new("1",imageToConvert.size,color=0)
    width, height = bw.size
    for i in range(width):
        for j in range(height):
            coordinate = i,j
            if isYellow(imageToConvert.getpixel(coordinate)):
               bw.putpixel(coordinate,1)

    bw.save(outputlocation)

# LEGACY: row with max yellow is no longer used; finding mass positions
uses the entire image.
# A function identifying the row of the image with the most yellow, to be
used for extracting
# mass positions
def findMaxRow(image):
    totyel = 0
    maxrow = 0
    width, height = image.size
    for i in range(height):
        tempyel = 0
        for j in range(width):
            coordinate = (j,i)
            if isYellow(image.getpixel(coordinate)):
                tempyel = tempyel + 1
        if tempyel>totyel:
            totyel = tempyel
            maxrow = i
    return maxrow

# LEGACY: Replaced with findPositions2 and findMassCenter.
# A function locating the center of each stretch of yellow pixels in a
given image, in a given
# row. This returns an array of pixel locations corresponding to these
centers.
def findPositions(image, row):
    startpoints = []
    endpoints = []
    centers = []
    width, height = image.size
    for i in range(width-1):
            coordinate1 = (i, row)
            coordinate2 = (i+1, row)
            if isYellow(image.getpixel(coordinate1))== False:
                if isYellow(image.getpixel(coordinate2)) == True:
                    startpoints.append(i+1)
            else:
                if isYellow(image.getpixel(coordinate2)) == False:
                    endpoints.append(i)
    for i in range(min(len(startpoints),nummasses)):
        centers.append(int((startpoints[i]+endpoints[i])/2))
    return centers
```

```
# Moves across the image searching for yellow pixels, then calls
findMassCenter to
# report the centers of each mass, returning a list of x-positions of
mass centers in the image
def findPositions2(image):
    centers = []
    width, height = image.size
    tempspot = -2*masswidth
    for i in range(width-1):
            for k in range(minheight, height-1):
                if (i-tempspot > 2* masswidth):
                    if isYellow(image.getpixel((i,k)))==True:
                        centers.append(findMassCenter(image,i))
                        tempspot = i
    return centers

#Iterates through locations near a mass, finding the average x-value of
yellow pixels in that area.
def findMassCenter(image, xstart):
    width, height = image.size
    total = 0
    count = 0
    for i in range(max(xstart-masswidth*2,0),
min(xstart+masswidth*2,width-1)):
        for j in range(minheight,height-1):
            if isYellow(image.getpixel((i,j))) == True:
                total += i
                count += 1
    if count == 0:
        return 0
    return total/count

# The main function: loads a video, and loops through each frame, finding
centers of masses.
# These pixel locations are stored to the centerpoints array, and then
plotted and saved to a CSV.
# Errors are reported if a frame does not provide the appropriate number
of mass locations.
try:
    video = cv2.VideoCapture(vidname)
    success, image = video.read()
    frame = 0
    while success:
        cv2.imwrite(tempimgname, image)
        image = Image.open(tempimgname)
        templocs = findPositions2(image)
        if len(templocs)> nummasses:
            print("Error: too many masses found. Frame:" + str(frame))
        if len(templocs) < nummasses:
            print("Error: too few masses found. Frame:" + str(frame))
        if len(templocs) == nummasses:
               for i in range(len(templocs)):
                   centerpoints[i].append(templocs[i])
        if (reportframes != 0):
            if (frame % reportframes == 0):
                print("Frame " + str(frame) + " complete")
        frame +=1
        success, image = video.read()
    for i in range(len(centerpoints)):
        plt.plot(centerpoints[i])
    with open(csvname, "w") as csvfile:
        writer = csv.writer(csvfile)
        writer.writerows(centerpoints)

except IOError:
    print("Video read error")
    pass
```