# Python program to implement

# Webcam Motion Detector

# importing OpenCV, time and Pandas library

import cv2, time, pandas

# importing datetime class from datetime library

from datetime import datetime

# Assigning our static\_back to None

static\_back = None

# List when any moving object appear

motion\_list = [ None, None ]

# Time of movement

time = []

# Initializing DataFrame, one column is start

# time and other column is end time

df = pandas.DataFrame(columns = ["Start", "End"])

# Capturing video

video = cv2.VideoCapture(0)

# Infinite while loop to treat stack of image as video

while True:

# Reading frame(image) from video

check, frame = video.read()

# Initializing motion = 0(no motion)

motion = 0

# Converting color image to gray\_scale image

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

# Converting gray scale image to GaussianBlur

# so that change can be find easily

gray = cv2.GaussianBlur(gray, (21, 21), 0)

# In first iteration we assign the value

# of static\_back to our first frame

if static\_back is None:

static\_back = gray

continue

# Difference between static background

# and current frame(which is GaussianBlur)

diff\_frame = cv2.absdiff(static\_back, gray)

# If change in between static background and

# current frame is greater than 30 it will show white color(255)

thresh\_frame = cv2.threshold(diff\_frame, 30, 255, cv2.THRESH\_BINARY)[1]

thresh\_frame = cv2.dilate(thresh\_frame, None, iterations = 2)

# Finding contour of moving object

cnts,\_ = cv2.findContours(thresh\_frame.copy(),

cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

for contour in cnts:

if cv2.contourArea(contour) < 10000:

continue

motion = 1

(x, y, w, h) = cv2.boundingRect(contour)

# making green rectangle around the moving object

cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 3)

# Appending status of motion

motion\_list.append(motion)

motion\_list = motion\_list[-2:]

# Appending Start time of motion

if motion\_list[-1] == 1 and motion\_list[-2] == 0:

time.append(datetime.now())

# Appending End time of motion

if motion\_list[-1] == 0 and motion\_list[-2] == 1:

time.append(datetime.now())

# Displaying image in gray\_scale

cv2.imshow("Gray Frame", gray)

# Displaying the difference in currentframe to

# the staticframe(very first\_frame)

cv2.imshow("Difference Frame", diff\_frame)

# Displaying the black and white image in which if

# intensity difference greater than 30 it will appear white

cv2.imshow("Threshold Frame", thresh\_frame)

# Displaying color frame with contour of motion of object

cv2.imshow("Color Frame", frame)

key = cv2.waitKey(1)

# if q entered whole process will stop

if key == ord('q'):

# if something is movingthen it append the end time of movement

if motion == 1:

time.append(datetime.now())

break

# Appending time of motion in DataFrame

for i in range(0, len(time), 2):

df = df.append({"Start":time[i], "End":time[i + 1]}, ignore\_index = True)

# Creating a CSV file in which time of movements will be saved

df.to\_csv("Time\_of\_movements.csv")

video.release()

# Destroying all the windows

cv2.destroyAllWindows()