# USAGE

# python barcode\_scanner\_video.py

# import the necessary packages

from imutils.video import VideoStream

from pyzbar import pyzbar

import argparse

import datetime

import imutils

import time

import cv2

# construct the argument parser and parse the arguments

ap = argparse.ArgumentParser()

ap.add\_argument("-o", "--output", type=str, default="barcodes.csv",

help="path to output CSV file containing barcodes")

args = vars(ap.parse\_args())

# initialize the video stream and allow the camera sensor to warm up

print("[INFO] starting video stream...")

# vs = VideoStream(src=0).start()

vs = VideoStream(usepicamera=True).start()

time.sleep(2.0)

# open the output CSV file for writing and initialize the set of

# barcodes found thus far

csv = open(args["output"], "w")

found = set()

# loop over the frames from the video stream

while True:

# grab the frame from the threaded video stream and resize it to

# have a maximum width of 400 pixels

frame = vs.read()

frame = imutils.resize(frame, width=400)

# find the barcodes in the frame and decode each of the barcodes

barcodes = pyzbar.decode(frame)

# loop over the detected barcodes

for barcode in barcodes:

# extract the bounding box location of the barcode and draw

# the bounding box surrounding the barcode on the image

(x, y, w, h) = barcode.rect

cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 2)

# the barcode data is a bytes object so if we want to draw it

# on our output image we need to convert it to a string first

barcodeData = barcode.data.decode("utf-8")

barcodeType = barcode.type

# draw the barcode data and barcode type on the image

text = "({})".format(barcodeType)

cv2.putText(frame, text, (x, y - 10),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 0, 255), 2)

# if the barcode text is currently not in our CSV file, write

# the timestamp + barcode to disk and update the set

if barcodeData not in found:

csv.write("{},{}\n".format(datetime.datetime.now(),

#barcodeData))

csv.flush()

#found.add(barcodeData)

# show the output frame

cv2.imshow("Barcode Scanner", frame)

key = cv2.waitKey(1) & 0xFF

# if the `q` key was pressed, break from the loop

if key == ord("q"):

break

# close the output CSV file do a bit of cleanup

print("[INFO] cleaning up...")

csv.close()

cv2.destroyAllWindows()

vs.stop()