vOS Source Code

Builddate: Sunday, December 11, 2023

Heeyeon lee

2023

vOS.py

vOS 메인코드

# 임포트 라이브러리

import os

import argparse

import cv2

import numpy as np

import time

import math

from threading import Thread

import importlib.util

import RPi.GPIO as GPIO

# 디버그 모드 On / Off

debug = True

class Color:

BLACK = (0, 0, 0)

WHITE = (255, 255, 255)

GREEN = (0, 255, 0)

RED = (0, 0, 255)

BLUE = (255, 0, 0)

YELLOW = (0, 255, 255)

# 비디오 스트림 정의

class VideoStream:

"""Camera object that controls video streaming from the Picamera"""

def \_\_init\_\_(self, resolution=(800, 420), framerate=30):

# 파이 카메라 초기화

self.stream = cv2.VideoCapture(0)

ret = self.stream.set(cv2.CAP\_PROP\_FOURCC, cv2.VideoWriter\_fourcc(\*'MJPG'))

ret = self.stream.set(3, resolution[0])

ret = self.stream.set(4, resolution[1])

# 첫 프레임 읽기

(self.grabbed, self.frame) = self.stream.read()

self.stopped = False

def start(self):

Thread(target=self.update, args=()).start()

return self

def update(self):

while True:

if self.stopped:

self.stream.release()

return

(self.grabbed, self.frame) = self.stream.read()

def read(self):

return self.frame

def stop(self):

self.stopped = True

# 입력 인자 정의

parser = argparse.ArgumentParser()

parser.add\_argument('--modeldir', help='Folder the .tflite file is located in',

required=True)

parser.add\_argument('--graph', help='Name of the .tflite file, if different than detect.tflite',

default='detect.tflite')

parser.add\_argument('--labels', help='Name of the labelmap file, if different than labelmap.txt',

default='labelmap.txt')

parser.add\_argument('--threshold', help='Minimum confidence threshold for displaying detected objects',

default=0.5)

parser.add\_argument('--resolution',

help='Desired webcam resolution in WxH. If the webcam does not support the resolution entered, errors may occur.',

default='800x420')

parser.add\_argument('--edgetpu', help='Use Coral Edge TPU Accelerator to speed up detection',

action='store\_true')

args = parser.parse\_args()

MODEL\_NAME = args.modeldir

GRAPH\_NAME = args.graph

LABELMAP\_NAME = args.labels

minConfidenceThreshold = float(args.threshold)

resW, resH = args.resolution.split('x')

imW, imH = int(resW), int(resH)

use\_TPU = args.edgetpu

# 텐서플로우 임포트

pkg = importlib.util.find\_spec('tflite\_runtime')

if pkg:

from tflite\_runtime.interpreter import Interpreter

if use\_TPU:

from tflite\_runtime.interpreter import load\_delegate

if use\_TPU:

if (GRAPH\_NAME == 'detect.tflite'):

GRAPH\_NAME = 'edgetpu.tflite'

# 디렉토리 패치

CWD\_PATH = os.getcwd()

# 모델 파일 패치

PATH\_TO\_CKPT = os.path.join(CWD\_PATH, MODEL\_NAME, GRAPH\_NAME)

# 라벨 맵 파일 패치

PATH\_TO\_LABELS = os.path.join(CWD\_PATH, MODEL\_NAME, LABELMAP\_NAME)

# 라벨 맵 가져오기

with open(PATH\_TO\_LABELS, 'r') as f:

labels = [line.strip() for line in f.readlines()]

if labels[0] == '???':

del (labels[0])

if use\_TPU:

interpreter = Interpreter(model\_path=PATH\_TO\_CKPT,

experimental\_delegates=[load\_delegate('libedgetpu.so.1.0')])

print(PATH\_TO\_CKPT)

else:

interpreter = Interpreter(model\_path=PATH\_TO\_CKPT)

interpreter.allocate\_tensors()

# 모델 불러오기

input\_details = interpreter.get\_input\_details()

output\_details = interpreter.get\_output\_details()

height = input\_details[0]['shape'][1]

width = input\_details[0]['shape'][2]

floating\_model = (input\_details[0]['dtype'] == np.float32)

input\_mean = 127.5

input\_std = 127.5

outname = output\_details[0]['name']

if ('StatefulPartitionedCall' in outname):

boxes\_idx, classes\_idx, scores\_idx = 1, 3, 0

else:

boxes\_idx, classes\_idx, scores\_idx = 0, 1, 2

# 프레임 초기화

framerateCalculate = 1

frequency = cv2.getTickFrequency()

# 비디오 스트림 초기화

videostream = VideoStream(resolution=(imW, imH), framerate=30).start()

time.sleep(1)

# 창 사이즈 지정

xFrame = 800

yFrame = 420

objectCount = 0

GPIO.setwarnings(False)

GPIO.setmode(GPIO.BCM)

GPIO.setup([6, 13, 16, 19, 20, 21, 26], GPIO.OUT)

# 카메라 캡쳐

while True:

# 타이머 시작 (프레임 계산)

t1 = cv2.getTickCount()

# 비디오 스트림

frame1 = videostream.read()

# 창 사이즈 지정

if not frame1 is None:

frame = frame1.copy()

frame\_rgb = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

frame\_resized = cv2.resize(frame\_rgb, (width, height))

input\_data = np.expand\_dims(frame\_resized, axis=0)

if floating\_model:

input\_data = (np.float32(input\_data) - input\_mean) / input\_std

interpreter.set\_tensor(input\_details[0]['index'], input\_data)

interpreter.invoke()

# 인식 결과 가져오기

boxes = interpreter.get\_tensor(output\_details[boxes\_idx]['index'])[0]

classes = interpreter.get\_tensor(output\_details[classes\_idx]['index'])[0]

scores = interpreter.get\_tensor(output\_details[scores\_idx]['index'])[0]

for i in range(len(scores)):

if ((scores[i] > minConfidenceThreshold) and (scores[i] <= 1.0)):

isFire = False

objectName = labels[int(classes[i])]

if (objectName == 'person'):

yMin = int(max(1, (boxes[i][0] \* imH)))

xMin = int(max(1, (boxes[i][1] \* imW)))

yMax = int(min(imH, (boxes[i][2] \* imH)))

xMax = int(min(imW, (boxes[i][3] \* imW)))

# 이름과 정확도 표시

label = '%s: %d%%' % (objectName, int(scores[i] \* 100))

labelSize, baseLine = cv2.getTextSize(label, cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, 2)

label\_yMin = max(yMin, labelSize[1] + 10)

cv2.rectangle(frame, (xMin, label\_yMin - labelSize[1] - 10),

(xMin + labelSize[0], label\_yMin + baseLine - 10), Color.WHITE, cv2.FILLED)

# 라벨 글씨 쓰기

cv2.putText(frame, label, (xMin, label\_yMin - 7), cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, Color.BLACK, 2)

# 박스 사이즈 계산

xBox = xMax - xMin

yBox = yMax - yMin

# 창 기준 좌표

xTarget = xMin + xBox / 2

yTarget = yMin + yBox / 2

# 정 중앙 기준 좌표

xPoint = xTarget - xFrame / 2

yPoint = (yTarget - yFrame / 2) \* (-1)

# 목표 좌표 표시

if debug:

cv2.putText(frame, '({: .1f},'.format(xPoint), (int(xTarget - 80), int(yTarget + 50)),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.8, Color.BLUE, 1, cv2.LINE\_AA)

cv2.putText(frame, '{: .1f})'.format(yPoint), (int(xTarget + 30), int(yTarget + 50)),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.8, Color.BLUE, 1, cv2.LINE\_AA)

# 발사 범위 반지름 값 지정

targetCircleRadius = int((xBox \* 2 + yBox \* 2) / 16)

# 목표 원 그리기

cv2.circle(frame, (int(xTarget), int(yTarget)), 10, Color.BLUE, 2)

cv2.circle(frame, (int(xTarget), int(yTarget)), targetCircleRadius, Color.YELLOW, 1)

# 인식된 객체 박스 그리기

cv2.rectangle(frame, (xMin, yMin), (xMax, yMax), Color.GREEN, 1)

# 거리 계산

distance = math.sqrt((int(xFrame / 2) - int(xTarget)) \*\* 2 + (int(yFrame / 2) - int(yTarget)) \*\* 2)

# 목표까지 선 그리기

cv2.line(frame, (int(xTarget), int(yTarget)), (int(xFrame / 2), int(yFrame / 2)), (0, 255, 0),1, cv2.LINE\_AA)

# 조준 되었을 때 발사

if distance <= targetCircleRadius:

cv2.putText(frame, 'Fire', (30, 120), cv2.FONT\_HERSHEY\_SIMPLEX, 1, Color.RED, 2, cv2.LINE\_AA)

isFire = True

objectCount += 1

# 인식된 객체의 개수

cv2.putText(frame, 'Object Count: {}'.format(objectCount), (30, 80), cv2.FONT\_HERSHEY\_SIMPLEX, 1, Color.BLACK, 2, cv2.LINE\_AA)

objectCount = 0

# 프레임 표시

cv2.putText(frame, 'FPS: {0:.2f}'.format(framerateCalculate), (30, 50), cv2.FONT\_HERSHEY\_SIMPLEX, 1, Color.BLACK, 2, cv2.LINE\_AA)

# 격자 표시

cv2.line(frame, (0, int(yFrame / 2)), (xFrame, int(yFrame / 2)), Color.WHITE, 1, cv2.LINE\_AA)

cv2.line(frame, (int(xFrame / 2), 0), (int(xFrame / 2), yFrame), Color.WHITE, 1, cv2.LINE\_AA)

# 표시

cv2.imshow('Auto Mode', frame)

# 프레임 계산

t2 = cv2.getTickCount()

time1 = (t2 - t1) / frequency

framerateCalculate = 1 / time1

# q 를 누르면 종료

if cv2.waitKey(1) == ord('q'):

break

# 창 닫기

#GPIO.clenup()

cv2.destroyAllWindows()

videostream.stop()