TensorFlow OpenCV Slack Flack

TensorFlow OpenCV Slack Flack

main.py : 웹 메인 서버

```
#!/usr/bin/env python
#
# Project: Streaming Tensorflow image with Flask
# Author: jframework@gmail.com
# Date: 2020/05/11
# Website: http://www.joang.com
# Description:
# Publishing a video from room , Tensorflow and OpenCV and GPIO, Flask
# Usage:
# 1. Install Python dependencies: tensorflow, opencv, gpio, cv2, flask. (wish that pip
install works like a charm)
# 2. Run "python3 main.py".
# 3. Navigate the browser to the local webpage. http://web.joang.com:8081/
#
#
from flask import Flask, render_template, Response, request
from camera import VideoCamera
from gpioControl import GpioControl
import os

app = Flask(__name__)

def shutdown_server():
    func = request.environ.get('werkzeug.server.shutdown')
    if func is None:
        raise RuntimeError('Not running with the kkoRack Server')
    func()

ONAIR = True
gpio = GpioControl()


@app.route('/')
def index():
    temp = os.popen("vcgencmd measure_temp").readline()
    return render_template('index.html', temperature=(temp.replace("temp=","")))
```

```
## Video ##
def gen(camera):
    global ONAIR
    while(ONAIR):
        frame = camera.get_frame()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n')

@app.route('/video_feed')
def video_feed():
    return Response(gen(VideoCamera()),
                    mimetype='multipart/x-mixed-replace; boundary=frame')



## Init ##
@app.route('/init')
def video_init():
    print("video_init")
    reString = gpio.initMotorPosition()
    return "Camera position init : " + reString




## camera movement ##
@app.route('/right')
def video_right():
    print("video_right")
    reString = gpio.moveRight()
    return "Camera Move right : " + reString

@app.route('/left')
def video_left():
    print("video_left")
    reString = gpio.moveLeft()
    return "Camera Move left : " + reString

@app.route('/up')
def video_up():
    print("video_up")
    reString = gpio.moveUp()
    return "Camera Move up : " + reString

@app.route('/down')
def video_down():
    print("video_down")
    reString = gpio.moveDown()
    return "Camera Move down : " + reString
```

```
## gpio on/off ##
@app.route('/clicklight')
def clicklight():
    reString = gpio.click()
    return "Light Click"


## System ##
@app.route('/stop')
def video_stop():
    global ONAIR
    ONAIR = False
    print("video_stop")
    return render_template('index.html')

@app.route('/start')
def video_start():
    global ONAIR
    ONAIR = True
    return render_template('index.html')

@app.route('/shutdown')
def shutdown():
    gpio.cleanUp()
    shutdown_server()
    return 'Server shutting down...'

@app.route('/temperature')
def temperature():
    temp = os.popen("vcgencmd measure_temp").readline()
    return (temp.replace("temp=",""))


if __name__ == '__main__':
    app.run(host='0.0.0.0', port='8081', debug=True)
```

camera.py : 카메라 기능

```
import cv2
from objectDetector import ObjectDetector

# Set up camera constants
IM_WIDTH = 640
IM_HEIGHT = 480

# Initialize frame rate calculation
frame_rate_calc = 1
```

```
freq = cv2.getTickFrequency()
font = cv2.FONT_HERSHEY_SIMPLEX

objDetector = ObjectDetector()


class VideoCamera(object):
    def __init__(self):
        # Using OpenCV to capture from device 0. If you have trouble capturing
        # from a webcam, comment the line below out and use a video file
        # instead.
        self.video = cv2.VideoCapture(0)
        # If you decide to use video.mp4, you must have this file in the folder
        # as the main.py.
        # self.video = cv2.VideoCapture('video.mp4')
        print( "width: {}, height : {}".format(self.video.get(3),
self.video.get(4) ) )
        ret = self.video.set(3,IM_WIDTH)
        ret = self.video.set(4,IM_HEIGHT)

    def __del__(self):
        self.video.release()

    def get_frame(self):
        global frame_rate_calc
        t1 = cv2.getTickCount()
        success, frame = self.video.read()
        # We are using Motion JPEG, but OpenCV defaults to capture raw images,
        # so we must encode it into JPEG in order to correctly display the
        # video stream.
        frame = objDetector.objectDetector(frame)

        # Draw FPS
        cv2.putText(frame,"FPS: {0:.2f}".format(frame_rate_calc),(30,50),font,1,
(255,255,0),2,cv2.LINE_AA)

        # FPS calculation
        t2 = cv2.getTickCount()
        time1 = (t2-t1)/freq
        frame_rate_calc = 1/time1

        ret, jpeg = cv2.imencode('.jpg', frame)
        return jpeg.tobytes()
```

objectDetector.py : 이미지 식별

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

# Import packages
```

```
import os
import cv2
import numpy as np
import tensorflow as tf
from time import sleep
import sys
from gpioControl import GpioControl
from slackmsg import SlackMsg

# Set up camera constants
IM_WIDTH = 640
IM_HEIGHT = 480

# This is needed since the working directory is the object_detection folder.
sys.path.append('/home/pi/tensorflow1/models/research')
sys.path.append('/home/pi/tensorflow1/models/research/object_detection')

# Import utilites
from utils import label_map_util
from utils import visualization_utils as vis_util

# Grab path to current working directory
#CWD_PATH = os.getcwd() + "/tensorflow1/models/research/object_detection"
CWD_PATH = "/home/pi/tensorflow1/models/research/object_detection"
#IMG_PATH = os.getcwd() + "/Pictures"
IMG_PATH = "/home/pi/Pictures"
print(" \n ###################################")
print("  Base Path %s" % CWD_PATH)
print("  Image Path %s" % IMG_PATH)
print(" ###################################")


#### Initialize TensorFlow model ####

# Name of the directory containing the object detection module we're using
MODEL_NAME = 'ssdlite_mobilenet_v2_coco_2018_05_09'

# Path to frozen detection graph .pb file, which contains the model that is used
# for object detection.
PATH_TO_CKPT = os.path.join(CWD_PATH,MODEL_NAME,'frozen_inference_graph.pb')

# Path to label map file
PATH_TO_LABELS = os.path.join(CWD_PATH,'data','mscoco_label_map.pbtxt')

# Number of classes the object detector can identify
NUM_CLASSES = 80

print(" \n ###################################")
print("  Load the label map ")
print(" ###################################")
```

```
## Load the label map.
# Label maps map indices to category names, so that when the convolution
# network predicts `5`, we know that this corresponds to `airplane`.
# Here we use internal utility functions, but anything that returns a
# dictionary mapping integers to appropriate string labels would be fine
label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories = label_map_util.convert_label_map_to_categories(label_map,
max_num_classes=NUM_CLASSES, use_display_name=True)
category_index = label_map_util.create_category_index(categories)

# Load the Tensorflow model into memory.
detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.compat.v1.GraphDef()
    with tf.io.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')

    sess = tf.compat.v1.Session(graph=detection_graph)

print(" \n ###################################")
print("  Define input and output tensors (i.e. data) for the object detection
classifier ")
print(" ###################################")

# Input tensor is the image
image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')

# Output tensors are the detection boxes, scores, and classes
# Each box represents a part of the image where a particular object was detected
detection_boxes = detection_graph.get_tensor_by_name('detection_boxes:0')

# Each score represents level of confidence for each of the objects.
# The score is shown on the result image, together with the class label.
detection_scores = detection_graph.get_tensor_by_name('detection_scores:0')
detection_classes = detection_graph.get_tensor_by_name('detection_classes:0')

# Number of objects detected
num_detections = detection_graph.get_tensor_by_name('num_detections:0')

print(" \n ###################################")
print("  Initialize other parameters ")
print(" ###################################")
# Initialize frame rate calculation
frame_rate_calc = 1
freq = cv2.getTickFrequency()
font = cv2.FONT_HERSHEY_SIMPLEX

# Initialize control variables used for pet detector
```

```
detected_inside = False
detected_outside = False

inside_counter = 0
outside_counter = 0

ObjX = 0
ObjY = 0

pause = 0
pause_counter = 0



gpio = GpioControl()
slackmsg = SlackMsg()


class ObjectDetector(object):

    def objectDetector(self, frame):
        # Use globals for the control variables so they retain their value after
function exits
        global detected_inside, detected_outside
        global inside_counter, outside_counter
        global pause, pause_counter

        frame_expanded = np.expand_dims(frame, axis=0)

        # Perform the actual detection by running the model with the image as input
        (boxes, scores, classes, num) = sess.run(
            [detection_boxes, detection_scores, detection_classes, num_detections],
            feed_dict={image_tensor: frame_expanded})

        # Draw the results of the detection (aka 'visulaize the results')
        vis_util.visualize_boxes_and_labels_on_image_array(
            frame,
            np.squeeze(boxes),
            np.squeeze(classes).astype(np.int32),
            np.squeeze(scores),
            category_index,
            use_normalized_coordinates=True,
            line_thickness=8,
            min_score_thresh=0.40)

        # Draw boxes defining "outside" locations.
        TL_outside = (int(IM_WIDTH*0.6),int(IM_HEIGHT*0.25))
        BR_outside = (int(IM_WIDTH*0.85),int(IM_HEIGHT*.85))
        cv2.rectangle(frame,TL_outside,BR_outside,(255,20,20),3)
        cv2.putText(frame,"Outside room",(TL_outside[0]+10,TL_outside[1]-10),font,1,
```

```
(255,20,255),3,cv2.LINE_AA)

        # Check the class of the top detected object by looking at classes[0][0].
        # If the top detected object is a person (73)
        # boxes ( xmin, xmax, ymin, ymax )
        threshold = 0.8
        for index, value in enumerate(classes[0]):
            ymin = boxes[0][index][0] * IM_HEIGHT
            xmin = boxes[0][index][1] * IM_WIDTH
            ymax = boxes[0][index][2] * IM_HEIGHT
            xmax = boxes[0][index][3] * IM_WIDTH
            #print(' =1= %s' % category_index.get(value) )
            if category_index.get(value) != None:
                personclassname = (category_index.get(value)).get('name')
                widthvalue = int((xmax - xmin) / 2)  # width 길이
                heightvalue = int((ymax - ymin) / 2)  # height 길이
                if( scores[0, index] != 0.0 ): print('> Score = %s, Object = %s ,
pause = %s' % (scores[0, index], personclassname, pause) )
                if scores[0, index] > threshold and personclassname == 'person' and
pause == 0:
                    print('> Detected %s' % personclassname )

                    ObjX = int(((boxes[0][0][1]+boxes[0][0][3])/2)*IM_WIDTH)
                    ObjY = int(((boxes[0][0][0]+boxes[0][0][2])/2)*IM_HEIGHT)

                    # Draw a circle at center of object
                    cv2.circle(frame,(ObjX,ObjY), 5, (75,13,180), -1)

                    # If object is in outside box, increment outside counter variable
                    if ((ObjX > TL_outside[0]) and (ObjX < BR_outside[0]) and (ObjY >
TL_outside[1]) and (ObjY < BR_outside[1])):
                        outside_counter = outside_counter + 1
                    else :
                        inside_counter = inside_counter + 1


        # If pet has been detected inside for more than 10 frames, set detected_inside
flag
        # and send a text to the phone.
        if inside_counter > 10:
            detected_inside = True
            cv2.imwrite(IMG_PATH+'/inside_' + str(ObjX) + '_' + str(ObjY) +
'_counter.jpg', frame)
            captureImg=os.path.join(IMG_PATH, 'inside_' + str(ObjX) + '_' + str(ObjY)
+ '_counter.jpg')
            response = slackmsg.uploadImage("Inside Photo", captureImg)
            captureImg=response['file']['permalink']
            slackmsg.sendMsg("Inside", captureImg)

            # Set move to detected object
```

```
            gpio.move_to_position(ObjX, ObjY)

            inside_counter = 0
            outside_counter = 0
            ObjX = 0
            ObjY = 0
            # Pause pet detection by setting "pause" flag
            pause = 1

        # If pet has been detected outside for more than 10 frames, set
detected_outside flag
        # and send a text to the phone.
        if outside_counter > 10:
            detected_outside = True
            cv2.imwrite(IMG_PATH+'/outside_' + str(ObjX) + '_' + str(ObjY) +
'_counter.jpg', frame)
            captureImg=os.path.join(IMG_PATH, 'outside_' + str(ObjX) + '_' + str(ObjY)
+ '_counter.jpg')
            response = slackmsg.uploadImage("Outside Photo", captureImg)
            captureImg=response['file']['permalink']
            slackmsg.sendMsg("Outside", captureImg)

            # Set move to detected object
            gpio.move_to_position(ObjX, ObjY)

            inside_counter = 0
            outside_counter = 0
            ObjX = 0
            ObjY = 0
            # Pause pet detection by setting "pause" flag
            pause = 1

        # If pause flag is set, draw message on screen.
        if pause == 1:
            if detected_inside == True:
                cv2.putText(frame,'Inside!',
(int(IM_WIDTH*.1),int(IM_HEIGHT*.5)),font,1,(0,0,0),7,cv2.LINE_AA)
                cv2.putText(frame,'Inside!',
(int(IM_WIDTH*.1),int(IM_HEIGHT*.5)),font,1,(95,176,23),5,cv2.LINE_AA)

            if detected_outside == True:
                cv2.putText(frame,'Outside!',
(int(IM_WIDTH*.1),int(IM_HEIGHT*.5)),font,1,(0,0,0),7,cv2.LINE_AA)
                cv2.putText(frame,'Outside!',
(int(IM_WIDTH*.1),int(IM_HEIGHT*.5)),font,1,(95,176,23),5,cv2.LINE_AA)

            # Increment pause counter until it reaches 30 (for a framerate of 1.5 FPS,
this is about 20 seconds),
            # then unpause the application (set pause flag to 0).
            pause_counter = pause_counter + 1
```

```
            if pause_counter > 30:
                pause = 0
                pause_counter = 0
                detected_inside = False
                detected_outside = False

        # Draw counter info
        cv2.putText(frame,'Detection counter: ' +
str(max(inside_counter,outside_counter)),(10,100),font,0.5,(255,255,0),1,cv2.LINE_AA)
        cv2.putText(frame,'Pause counter: ' + str(pause_counter),(10,150),font,0.5,
(255,255,0),1,cv2.LINE_AA)

        return frame
```

gpioControl.py : 등 켜기 등 릴ㄹ레이 모듈 , 카메라 위치 조정 스텝 모터 조정

```
import RPi.GPIO as GPIO
from time import sleep


GPIO.setmode(GPIO.BOARD)
GPIO.setup(12, GPIO.OUT, initial=1)
GPIO.setup(18, GPIO.OUT, initial=1)
GPIO.setup(11, GPIO.OUT, initial=1) # light
p1 = GPIO.PWM(12, 50)  # 50 Hz
p2 = GPIO.PWM(18, 50)  # 50
p1.start(0)
p2.start(0)
p1.ChangeDutyCycle(0)
p2.ChangeDutyCycle(0)

verticalVal = 6.5
horizontalVal = 6.5

cameraPositionX = 6.5
cameraPositionY = 6.5

# Set up camera constants
IM_WIDTH = 640
IM_HEIGHT = 480


class GpioControl(object):

    def __init__(self):
        global verticalVal
        global horizontalVal
        global p1
```

```
        global p2
        p1.ChangeDutyCycle(6.5)
        p2.ChangeDutyCycle(6.5)
        sleep(0.1)
        p1.ChangeDutyCycle(0)
        p2.ChangeDutyCycle(0)
        verticalVal = 6.5
        horizontalVal = 6.5
        print("> Init Vert=" + str(verticalVal) + ",Hort=" + str(horizontalVal))

    def click(self):
        GPIO.output(11, GPIO.LOW)
        sleep(0.5)
        GPIO.output(11, GPIO.HIGH)
        sleep(1)

    def __del__(self):
        global p1
        global p2
        p1.stop()
        p2.stop()
        print(" GPIO.__del__() ")
        GPIO.cleanup()

    def cleanUp(self):
        global p1
        global p2
        p1.stop()
        p2.stop()
        print(" GPIO.cleanUp() ")
        GPIO.cleanup()
        sleep(2)

    def initMotorPosition(self):
        # Init
        global verticalVal
        global horizontalVal
        global p1
        global p2
        p1.ChangeDutyCycle(6.5)
        p2.ChangeDutyCycle(6.5)
        sleep(0.1)
        p1.ChangeDutyCycle(0)
        p2.ChangeDutyCycle(0)
        verticalVal = 6.5
        horizontalVal = 6.5
        print("> Init Vert=" + str(verticalVal) + ",Hort=" + str(horizontalVal))
        return "Vert=" + str(verticalVal) + ",Hort=" + str(horizontalVal)

    def moveUp(self):
```

```
        global verticalVal
        global horizontalVal
        global p2
        verticalVal = round(verticalVal+0.2, 1)
        p2.ChangeDutyCycle(verticalVal)
        print("> UP Vert=" + str(verticalVal) + ",Hort=" + str(horizontalVal))
        sleep(0.1)
        p2.ChangeDutyCycle(0)
        return "Vert=" + str(verticalVal) + ",Hort=" + str(horizontalVal)

    def moveDown(self):
        global verticalVal
        global horizontalVal
        global p2
        verticalVal = round(verticalVal-0.2, 1)
        p2.ChangeDutyCycle(verticalVal)
        print("> Down Vert=" + str(verticalVal) + ",Hort=" + str(horizontalVal))
        sleep(0.1)
        p2.ChangeDutyCycle(0)
        return "Vert=" + str(verticalVal) + ",Hort=" + str(horizontalVal)

    def moveRight(self):
        global verticalVal
        global horizontalVal
        global p1
        horizontalVal = round(horizontalVal+0.2, 1)
        p1.ChangeDutyCycle(horizontalVal)
        print("> Right Vert=" + str(verticalVal) + ",Hort=" + str(horizontalVal))
        sleep(0.1)
        p1.ChangeDutyCycle(0)
        return "Vert=" + str(verticalVal) + ",Hort=" + str(horizontalVal)

    def moveLeft(self):
        global verticalVal
        global horizontalVal
        global p1
        horizontalVal = round(horizontalVal-0.2, 1)
        p1.ChangeDutyCycle(horizontalVal)
        print("> Left Vert=" + str(verticalVal) + ",Hort=" + str(horizontalVal))
        sleep(0.1)
        p1.ChangeDutyCycle(0)
        return "Vert=" + str(verticalVal) + ",Hort=" + str(horizontalVal)


    def move_to_position(self,ObjX, ObjY):
        global cameraPositionX
        global cameraPositionY
        global p1
        global p2
        print(" >> Move to location x=%s, y=%s" % (ObjX, ObjY))
```

```
        moveLoop = True
        movX = int(IM_WIDTH/2)-ObjX
        movY = int(IM_HEIGHT/2)-ObjY
        print(" >> Center location movX=%s, movY=%s" % (movX, movY))

        xx = 1
        xy = 0
        yx = 1
        yy = 0
        while(moveLoop):
            if( xy < abs(movX) ):
                p1.ChangeDutyCycle(cameraPositionX)
                sleep(0.1)
                p1.ChangeDutyCycle(0)
                print("xx=" + str(xx) + ", xy="+ str(xy) +" cameraPositionX="
+str(round(cameraPositionX,1)))
                xy = xx*xx * 12
                xx = xx + 1
                if(movX > 0): cameraPositionX = cameraPositionX - 0.2
                else: cameraPositionX = cameraPositionX + 0.2
            if( yy < abs(movY) ):
                p2.ChangeDutyCycle(cameraPositionY)
                sleep(0.1)
                p2.ChangeDutyCycle(0)
                print("yx=" + str(yx) + ", yy="+ str(yy) +" cameraPositionY="
+str(round(cameraPositionY,1)))
                yy = yx*yx * 12
                yx = yx + 1
                if(movY > 0): cameraPositionY = cameraPositionY + 0.2
                else: cameraPositionY = cameraPositionY - 0.2
            elif( xy >= movX and yy >= movY):
                print(" >> Center location movX="+str(movX)+", movY=" + str(movY))
                print(" >> Position location xy="+str(xy)
                      +", yy="+str(yy)
                      +" cameraPositionX="+str(round(cameraPositionX,1))+
                       " cameraPositionY="+str(round(cameraPositionY,1))+" .. ")
                moveLoop = False
```

Revision #2
★Created 2023-06-02 12:38:10 UTC by Hyeon Su Ryu
Updated 2024-03-31 11:26:03 UTC by Hyeon Su Ryu