3.1.4 Tensorflow object recognition

TensorFlow is an open source software library that uses data flow graphs for numerical calculations.

Features

1. High flexibility
2. True Portability
3. Connect scientific research and products
4. Differentiate automatically
5. Support multi-language
6. Performance optimization

Code path:

/home/pi/Yahboom_Project/Raspbot/3.AI Vision course/05.Object recognition/Object recognition.ipynb

```
import numpy as np
import cv2
import os,time
import tensorflow as tf
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as vis_utils
import ipywidgets.widgets as widgets
```

```
# Init camera
cap = cv2.VideoCapture(0)
cap.set(3, 320) # set Width
cap.set(4, 240) # set Height
cap.set(5, 30)    # set Frame rate
cap.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter.fourcc('M', 'J', 'P', 'G'))
cap.set(cv2.CAP_PROP_BRIGHTNESS, 40) #Set brightness -64 - 64    0.0
        cap.set(cv2.CAP_PROP_CONTRAST, 50) #Set contrast -64 - 64    2.0
cap.set(cv2.CAP_PROP_EXPOSURE, 156)    #Set exposure 1.0 - 5000    156.0
```

```
image_widget = widgets.Image(format='jpg', width=320, height=240)
```

```
display(image_widget)
```

```
# Init tf model

MODEL_NAME = 'ssdlite_mobilenet_v2_coco_2018_05_09' #fast
PATH_TO_CKPT = MODEL_NAME + '/frozen_inference_graph.pb'
PATH_TO_LABELS = os.path.join('data', 'mscoco_label_map.pbtxt')
NUM_CLASSES = 90
IMAGE_SIZE = (12, 8)
fileAlreadyExists = os.path.isfile(PATH_TO_CKPT)

if not fileAlreadyExists:
    print('Model does not exsist !')
    exit
```

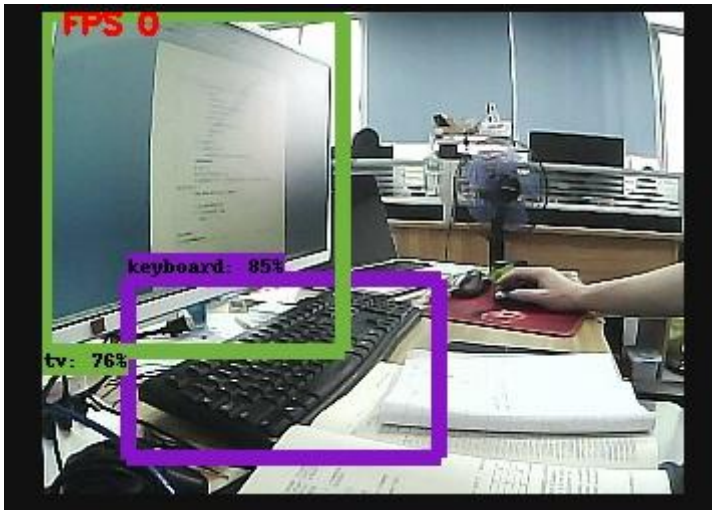
```
# LOAD GRAPH
print('Loading...')
detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.compat.v1.GraphDef()
    with tf.io.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')
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)
print('Finish Load Graph..')
```

```
print(type(category_index))
```

```
print("dict['Name']: ", category_index[1]['name'])
```

```
# Main Thread
t_start = time.time()
fps = 0

with detection_graph.as_default():
    with tf.compat.v1.Session(graph=detection_graph) as sess:
        while True:
            ret, frame = cap.read()
#               frame = cv2.flip(frame, -1) # Flip camera vertically
#                 frame = cv2.resize(frame,(320,240))
            ###############
            image_np_expanded = np.expand_dims(frame, axis=0)
            image_tensor                                                    =
detection_graph.get_tensor_by_name('image_tensor:0')
            detection_boxes =
```

```
detection_graph.get_tensor_by_name('detection_boxes:0')
                detection_scores =
detection_graph.get_tensor_by_name('detection_scores:0')
                detection_classes =
detection_graph.get_tensor_by_name('detection_classes:0')
                num_detections =
detection_graph.get_tensor_by_name('num_detections:0')

#                   print('Running detection..')
                (boxes, scores, classes, num) = sess.run(
                    [detection_boxes, detection_scores, detection_classes,
num_detections],
                    feed_dict={image_tensor: image_np_expanded})

#                   print('Done.    Visualizing..')
                vis_utils.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)

                for i in range(0, 10):
                    if scores[0][i] >= 0.5:
                        print(category_index[int(classes[0][i])]['name'])
                ###############
                fps = fps + 1
                mfps = fps / (time.time() - t_start)
                cv2.putText(frame, "FPS " + str(int(mfps)), (10,10),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,255), 2)
                image_widget.value = bgr8_to_jpeg(frame)

                k = cv2.waitKey(3000) & 0xff
                if k == 27: # press 'ESC' to quit
                    break

cap.release()
cv2.destroyAllWindows()
```




```
person
person
person
person
person
person
person
person
person
laptop
tv
tv
```

This program runs on JupyterLab with low frame rate, but you can try python programs on a Raspberry Pi desktop, effect will become better.