視訊串流與追蹤HW3

參考網址:
(1) https://github.com/yehengchen/Object-Detection-and-Tracking/tree/master/OneStage/yolo/yolov3_sort
(2) https://tw.live/

1. Experiment setup:

我參考網址(1)中的專案yolov3_sort進行實作。首先從網址(2)中的台灣即時攝影機中擷取一段時間的影像，依照專案指示的方式匯入影片以及weight檔就可以使用。

專案使用的model為yolov3 + SORT，weight檔案為pretrained。

2. Briefly explain your code:

參考網址(1)的程式碼進行實作，yolov3與sort都直接使用他準備好的部分，後處理部分如下:

```
labelsPath = os.path.sep.join([args["yolo"], "coco.names"])
LABELS = open(labelsPath).read().strip().split("\n")

# initialize a list of colors to represent each possible class label
np.random.seed(42)
COLORS = np.random.randint(0, 255, size=(200, 3),
    dtype="uint8")

# derive the paths to the YOLO weights and model configuration
weightsPath = os.path.sep.join([args["yolo"], "yolov3.weights"])
configPath = os.path.sep.join([args["yolo"], "yolov3.cfg"])

# load our YOLO object detector trained on COCO dataset (80 classes)
# and determine only the *output* layer names that we need from YOLO
print("[INFO] loading YOLO from disk...")
net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)
ln = net.getLayerNames()
ln = [ln[i[0] - 1] for i in net.getUnconnectedOutLayers()]

# initialize the video stream, pointer to output video file, and
# frame dimensions
vs = cv2.VideoCapture(args["input"])
writer = None
(W, H) = (None, None)

frameIndex = 0

# try to determine the total number of frames in the video file
try:
    prop = cv2.cv.CV_CAP_PROP_FRAME_COUNT if imutils.is_cv2() \
        else cv2.CAP_PROP_FRAME_COUNT
    total = int(vs.get(prop))
    print("[INFO] {} total frames in video".format(total))

# an error occurred while trying to determine the total
# number of frames in the video file
except:
    print("[INFO] could not determine # of frames in video")
    print("[INFO] no approx. completion time can be provided")
    total = -1
```

讀取labels, weights和config， 並設定好連接model以及擷取影片的基礎設定。

```
while True:

    # read the next frame from the file
    (grabbed, frame) = vs.read()

    # if the frame was not grabbed, then we have reached the end
    # of the stream
    if not grabbed:
        break

    # if the frame dimensions are empty, grab them
    if W is None or H is None:
        (H, W) = frame.shape[:2]

    # construct a blob from the input frame and then perform a forward
    # pass of the YOLO object detector, giving us our bounding boxes
    # and associated probabilities
    blob = cv2.dnn.blobFromImage(frame, 1 / 255.0, (416, 416),
        swapRB=True, crop=False)
    net.setInput(blob)
    start = time.time()
    layerOutputs = net.forward(ln)
    end = time.time()

    # initialize our lists of detected bounding boxes, confidences,
    # and class IDs, respectively
    boxes = []
    confidences = []
    classIDs = []
```

讀取影片frame，將讀取的資料重設為yolov3的input size，再接進model得到output。

```
for output in layerOutputs:
    # loop over each of the detections
    for detection in output:
        # extract the class ID and confidence (i.e., probability)
        # of the current object detection
        scores = detection[5:]
        classID = np.argmax(scores)
        confidence = scores[classID]


        # scale the bounding box coordinates back relative to
        # the size of the image, keeping in mind that YOLO
        # actually returns the center (x, y)-coordinates of
        # the bounding box followed by the boxes' width and
        # height
        box = detection[0:4] * np.array([W, H, W, H])
        (centerX, centerY, width, height) = box.astype("int")

        # use the center (x, y)-coordinates to derive the top
        # and and left corner of the bounding box
        x = int(centerX - (width / 2))
        y = int(centerY - (height / 2))
```

```
if mouse_x > x and mouse_x < x + width and mouse_y > y and mouse_y < y + height
and mouse_x != -1 and mouse_y != -1 and confidence > args["confidence"]:
    print(classID)
    mouse_y = -1
    mouse_x = -1
    chosen_classID.append(classID)
# filter out weak predictions by ensuring the detected
# probability is greater than the minimum probability
if confidence > args["confidence"] and (classID in chosen_classID or len(chosen_classID) == 0):


    # update our list of bounding box coordinates,
    # confidences, and class IDs
    boxes.append([x, y, int(width), int(height)])
    confidences.append(float(confidence))
    classIDs.append(classID)
```

```
def onmouse(event, x, y, flags, param):
    global mouse_x
    global mouse_y
    global chosen_classID
    if event==cv2.EVENT_LBUTTONDBLCLK:
        if(len(chosen_classID) != 0):
            chosen_classID.clear()
            mouse_x = -1;
            mouse_y = -1;
        else:
            mouse_x = x;
            mouse_y = y;

cv2.namedWindow("frame")
cv2.setMouseCallback("frame",onmouse)
```

根據output，我們可以得到每一個偵測到的物件以及其外框資訊與信心指數。此時我定義了滑鼠的callback function(以雙擊左鍵作為觸發條件)。檢查所點擊的座標是否在某個信心值足夠的外框範圍中，紀錄該範圍代表的物件並根據這個紀錄來決定哪些物件要被畫出框。

3. video result:

影片網址: https://www.youtube.com/watch?v=liRndnNoGmE

影片流程: 一開始用框標記人與車 -> 雙擊行人 -> 只標記行人 -> 雙擊任意處 -> 變回標記人與車 -> 雙擊車 -> 只標記車 -> 雙擊任意處 -> 變回標記人與車

4. discussion:

這次作業中我比較沒有遇到太多的困難，一開始雖然找資料花了一點時間，但仍然找到一個不錯的專案供我參考。因此我所需要做的只是找出他的各個物件所代表的ID，並根據這個ID進行畫框的建立。