source code : https://github.com/Aeocp/Traffic_speed_detection

รันผ่าน : https://colab.research.google.com/drive/1DA47n8S2b5Yc7tsJbMjgRkn49eZ1pyFL?usp=sharing

วิธีการรันโค้ด Traffic Counting

- เข้า https://colab.research.google.com/drive/1DA47n8S2b5Yc7tsJbMjgRkn49eZ1pyFL?usp=sharing

Step1

1. รัน 2 โค้ดดังภาพด้านล่างเพื่อ clone code จาก github และดาวน์โหลด requirements

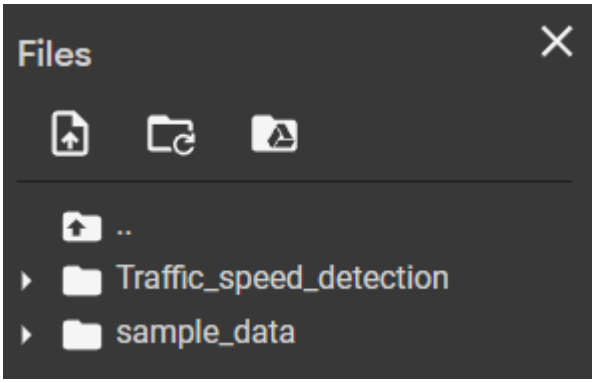
```
!git clone https://github.com/Aeocp/Traffic_speed_detection
```

```
Cloning into 'Traffic_speed_detection'...
remote: Enumerating objects: 1113, done.
remote: Counting objects: 100% (28/28), done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 1113 (delta 13), reused 27 (delta 12), pack-reused 1085
Receiving objects: 100% (1113/1113), 333.08 MiB | 38.32 MiB/s, done.
Resolving deltas: 100% (612/612), done.
```

```
%cd /content/Traffic_speed_detection/
!pip install -r requirements.txt
```

```
/content/Traffic_speed_detection
Requirement already satisfied: imutils in /usr/local/lib/python3.7/dist-packages (from -r requirements.txt (line 4)) (0.5.4)
Requirement already satisfied: keras in /usr/local/lib/python3.7/dist-packages (from -r requirements.txt (line 5)) (2.7.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (from -r requirements.txt (line 6)) (3.2.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from -r requirements.txt (line 7)) (1.19.5)
Requirement already satisfied: opencv-python in /usr/local/lib/python3.7/dist-packages (from -r requirements.txt (line 8)) (4.1.2.30)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (from -r requirements.txt (line 11)) (1.0.1)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from -r requirements.txt (line 12)) (1.4.1)
Requirement already satisfied: tensorboard in /usr/local/lib/python3.7/dist-packages (from -r requirements.txt (line 13)) (2.7.0)
Requirement already satisfied: tensorflow in /usr/local/lib/python3.7/dist-packages (from -r requirements.txt (line 14)) (2.7.0)
Requirement already satisfied: tensorflow-estimator in /usr/local/lib/python3.7/dist-packages (from -r requirements.txt (line 15)) (2.7.0)
Collecting tensorflow-gpu
  Downloading tensorflow_gpu-2.7.0-cp37-cp37m-manylinux2010_x86_64.whl (489.6 MB)
     |                                | 202.2 MB 1.3 MB/s eta 0:03:36
```

จะได้ floder ที่มีชื่อว่า “Traffic_speed_detection” ดังภาพ



2. นำเข้าข้อมูล (วีดีโอและข้อมูลที่ได้จาก Yolo)

เนื่องจากใน github ลงคลิปวีดีโอในขนาดที่จำกัดจึงต้องดึงข้อมูลจาก google drive โดยสามารถโหลดตัวอย่างได้จาก

https://www.dropbox.com/sh/26yoftfgxx5hcbr/AADCoZmdosS1-ZgMHB_ac6q3a?dl=0

```
%cd /content/
from google.colab import drive
drive.mount('/content/drive')
```

Step2A และ Step2B

เป็นการหา input เพื่อกำหนดตำแหน่งของเส้นวัดความเร็ว โดย

Step2A : เมื่อใช้ตัวอย่างวีดีโอที่มีอยู่หรือมุมกล้องตัวเดียวกับที่มีอยู่

เช่น หากต้องการ test ไฟล์ hlp-01-20210610-170005-cut.mp4

จะมี input ให้เลือกได้แก่

0.525,0.495,0.7,0.531,0.33,0.7,0.58,0.83	=	32 เมตร
0.525,0.475,0.72,0.515,0.33,0.7,0.58,0.83	=	36 เมตร
0.525,0.495,0.7,0.531,0.3,0.755,0.55,0.91	=	36 เมตร
0.525,0.475,0.72,0.515,0.3,0.755,0.55,0.91	=	40 เมตร

ซึ่งจะนำ input ชุดใดชุดหนึ่งและ ระยะห่างเส้นด้านหลัง นำไปใช้ใน Step 3

Step2B : เมื่อใช้มุมกล้องอื่นๆนอกเหนือจากตัวอย่าง

1. แก้ไขบรรทัดบริเวณสีเขียว

   vidcap = cv2.VideoCapture ("/content/Traffic_counting/video/hlp/080841-03.mp4")

   ให้เป็น path ของวีดีโอที่ต้องการหาตำแหน่งเส้น

2. รันโค้ดด้านล่าง

```
%cd /content/
import cv2
from google.colab.patches import cv2_imshow
vidcap = cv2.VideoCapture("/content/drive/MyDrive/test/hlp-01-20210610-170005-cut.mp4")
success,image = vidcap.read()
frameY = image.shape[0]
frameX = image.shape[1]
x1 = []
y1 = []
x2 = []
y2 = []
print("Enter the beginning and ending of two line (0-1) (xb1,yb1,xe1,ye1,xb2,yb2,xe2,ye2)")
Bp = input("Enter line position :").split(",")
x1.append((Bp[0]))
y1.append((Bp[1]))
x1.append((Bp[2]))
y1.append((Bp[3]))
x2.append((Bp[4]))
y2.append((Bp[5]))
x2.append((Bp[6]))
y2.append((Bp[7]))
line = [(int(float(x1[0]) * frameX), int(float(y1[0]) * frameY)), (int(float(x1[1]) * frameX), int(float(y1[1]) * frameY))]
line2 = [(int(float(x2[0]) * frameX), int(float(y2[0]) * frameY)), (int(float(x2[1]) * frameX), int(float(y2[1]) * frameY))]
cv2.line(image, line[0], line[1], (0, 0, 255), 3)
cv2.line(image, line2[0], line2[1], (0, 0, 255), 3)
cv2.imwrite("pic_output.jpg", image)
img = cv2.imread("/content/pic_output.jpg")
cv2_imshow(img)
cv2.waitKey(0)
```

3. เมื่อรันแล้วจะต้องใส่ input ซึ่งจะเป็นตำแหน่งของเส้นวัดความเร็ว

* โดย input ตำแหน่งเส้นจะอยู่ในรูปแบบของ (xs1,ys1,xe1,ye1,xs2,ys2,xe2,ye2)

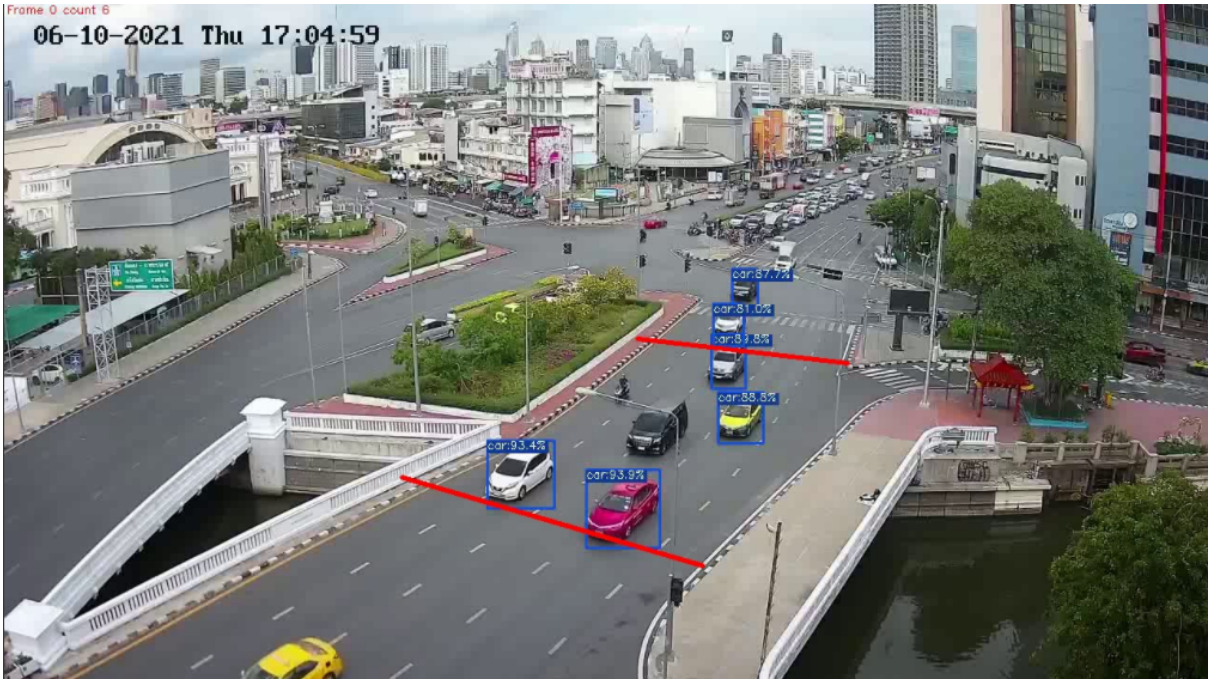
* โดยที่ x,y จะมีค่าระหว่าง 0-1

```
/content
Enter the beginning and ending of two line (0-1) (xb1,yb1,xe1,ye1,xb2,yb2,xe2,ye2)
Enter line position :0.525,0.495,0.7,0.531,0.33,0.7,0.58,0.83
```

จะได้ผลเป็นภาพที่มีเส้นสีแดงปรากฏ(ดังภาพตัวอย่าง) เราจะต้องปรับเปลี่ยนตัวเลขเพื่อให้ได้ตำแหน่งของเส้นวัดความเร็วที่เหมาะสม และจดบันทึกแยกเอาไว้

รวมถึงคำนวณระยะห่างของเส้นตามความเป็นจริงโดยสามารถใช้หลักการดังนี้

https://docs.google.com/document/d/1uIpyYGBzCeJkScWv91m5tHODf-5Pcrtw7Q47AA753Dw/edit?usp=sharing



Step3 การรันโปรแกรม

1. คล้ายกับวิธีการใน **Step2B** ในคำสั่ง

!python main.py

--video /content/drive/MyDrive/test/hlp-01-20210610-170005-cut.mp4

--text /content/drive/MyDrive/test/hlp-01-20210610-170005-cut.mp4.txt

--output /content/hlp-01.avi

เราจะต้องเปลี่ยน path ของไฟล์วีดีโอและtxtที่ได้จาก yolo ให้เป็น path ที่ต้องการ รวมถึงสามารถเปลี่ยนตำแหน่งที่จะเก็บวีดีโอ output และชื่อได้อีกด้วย

!python main.py

--video **โฟลเดอร์ที่ไฟล์วิดีโออยู่**

--text **โฟลเดอร์ที่ไฟล์outputที่ได้จากYolo(.txt)**

--output **โฟลเดอร์ที่ต้องการให้แสดงวีดีโอ**

ตัวอย่าง

!python main.py --video **/content/Traffic_counting/video/hlp/080841-03.mp4** --text

/content/Traffic_counting/input_txt/hlp/080841-03.txt --output **/content/Traffic_counting/080841-03-s1.avi**

2. เมื่อรันแล้วจะต้องใส่ input ซึ่งจะเป็นตำแหน่งของเส้นวัดความเร็ว ซึ่งได้มาจาก Step2A หรือ Step2B และ ระยะห่างระหว่างเส้น (หน่วยเมตร)

* โดย input ตำแหน่งเส้นจะอยู่ในรูปแบบของ (xs1,ys1,xe1,ye1,xs2,ys2,xe2,ye2)

* โดยที่ x,y จะมีค่าระหว่าง 0-1

```
Enter the beginning and ending of two line (0-1) (xb1,yb1,xe1,ye1,xb2,yb2,xe2,ye2)
Enter line position :0.525,0.495,0.7,0.531,0.33,0.7,0.58,0.83
Enter distance between lines(m) :32
```

3. รอจนรันผลเสร็จจะได้ดังนี้

```
Frame: 103  ID: 6  speed: 41.63855421686747
Frame: 126  ID: 5  speed: 43.74683544303798
Frame: 147  ID: 7  speed: 49.37142857142857
Frame: 167  ID: 17  speed: 51.582089552238806
Frame: 262  ID: 22  speed: 40.65882352941176
Frame: 297  ID: 43  speed: 54.857142857142854
Frame: 299  ID: 29  speed: 56.65573770491804
Frame: 310  ID: 32  speed: 43.74683544303798
Frame: 343  ID: 37  speed: 51.582089552238806
Frame: 413  ID: 46  speed: 48.67605633802817
Frame: 473  ID: 65  speed: 57.6
Frame: 518  ID: 59  speed: 54.857142857142854
Frame: 751  ID: 147  speed: 104.72727272727272
Frame: 752  ID: 142  speed: 50.8235294117647
Frame: 776  ID: 152  speed: 132.9230769230769
Frame: 782  ID: 156  speed: 58.57627118644068
Frame: 959  ID: 182  speed: 51.582089552238806
Frame: 1115  ID: 200  speed: 73.53191489361703
imutils FPS: 3.840095815714019
speed_avg : 74.83333333333333
จำนวนรถที่วัดความเร็วได้ 18
จำนวนรถทั้งหมด [[19, 40]]
```

ซึ่งจะแสดง frame ID และความเร็วที่วัดได้ของรถเมื่อผ่านเส้นที่ 2

ความเร็วเฉลี่ย จำนวนรถที่วัดความเร็วได้และจำนวนรถทั้งหมดที่ผ่านแต่ละเส้น

หมายเหตุ : รถที่จะถูกคำนวณความเร็วจะต้องผ่านทั้งสองเส้นเท่านั้น

Step4 Show Video

1. แก้ไขโค้ด %cd /content/ ให้เป็นตำแหน่ง path ที่ iutput อยู่
2. แก้ชื่อใน !ffmpeg -i hlp-01.avi output.mp4 ให้เป็นชื่อไฟล์ iutput ที่ตั้งไว้

จะแสดงวีดีโอผลลัพท์ที่ได้ขึ้นมา และสามารถดาวน์โหลดผลลัพท์ได้จากไฟล์ output.mp4

เพิ่มเติม การปรับค่า theshold สำหรับการเว้นเฟรม

**โค้ดปัจจุบันถูกตั้งค่า theshold เป็นแบบ เว้นสองเฟรม (10 frame/sec) และอ่านค่าแบบเว้นสองเฟรม หากต้องการแก้ไขสามารถแก้ไขโดย*

1. การเว้นเฟรม : /content/Traffic_counting/main.py

    * บรรทัดที่ 118 : if(frame_index%3 == 1):
        - if(frame_index%1 == 0) : ไม่เว้นเฟรม
        - if(frame_index%2 == 1) : เว้นเฟรมหนึ่งเฟรม
        - if(frame_index%3 == 0) : เว้นเฟรมสองเฟรม
        - if(frame_index%4 == 0) : เว้นเฟรมสามเฟรม

2. ปรับค่า theshold :

    * กรณีไม่เว้นเฟรม แก้ไขดังนี้

    /content/Traffic_counting/main.py

    - บรรทัดที่ 40 : max_cosine_distance = 0.3
    - บรรทัดที่ 42 : nms_max_overlap = 1.0

    /content/Traffic_counting/deep_sort/tracker.py

    - บรรทัดที่ 37 : แก้

      max_iou_distance= 0.7

      max_age = 30

      n_init = 3

      adc_threshold = 0.5

    * กรณี เว้นเฟรม**หนึ่ง**เฟรม แก้ไขดังนี้

    /content/Traffic_counting/main.py

    - บรรทัดที่ 40 : max_cosine_distance = 0.3
    - บรรทัดที่ 42 : nms_max_overlap = 1.0

    /content/Traffic_counting/deep_sort/tracker.py

    - บรรทัดที่ 37 : แก้

      max_iou_distance= 0.95

max_age = 50

n_init = 3

adc_threshold = 0.5

* กรณี เว้นเฟรม**สอง**เฟรม แก้ไขดังนี้

/content/Traffic_counting/main.py

- บรรทัดที่ 40 : max_cosine_distance = 0.75
- บรรทัดที่ 42 : nms_max_overlap = 1.0

/content/Traffic_counting/deep_sort/tracker.py

- บรรทัดที่ 37 : แก้

  max_iou_distance=0.95

  max_age = 50

  n_init = 1

  adc_threshold = 0.5

* กรณี เว้นเฟรม**สาม**เฟรม แก้ไขดังนี้

/content/Traffic_counting/main.py

- บรรทัดที่ 40 : max_cosine_distance = 1.3
- บรรทัดที่ 42 : nms_max_overlap = 1.0

/content/Traffic_counting/deep_sort/tracker.py

- บรรทัดที่ 37 : แก้

  max_iou_distance= 0.95

  max_age = 50

  n_init = 1

  adc_threshold = 0.5