pysource (https://pysource.com/)

Object Tracking with Opencv and Python

(https://pysource.com/2021/01/28/object-tracking-with-opencv-and-python/)

by Sergio Canu (https://pysource.com/author/admin/)

TUTORIALS (HTTPS://PYSOURCE.COM/CATEGORY/TUTORIALS/)

object_tracking.zip (https://pysource.com/wp-content/uploads/2021/01/object_tracking.zip) Download (https://pysource.com/wp-content/uploads/2021/01/object_tracking.zip)

In this tutorial we will learn how to use Object Tracking with Opencv and Python.

First of all it must be clear that what is the difference between object detection and object tracking:

Object detection is the detection on every single frame and frame after frame. **Object tracking** does frame-by-frame tracking but keeps the history of where the object is at a time after time

We will talk first about object detection and then about how to apply object tracking to the detection.

What are the possible applications?

The possible applications are different for example, counting how many people are in a certain area, checking how many objects pass on a conveyor belt, or counting the vehicles on a highway.

Surely where having seen the tutorial you will easily think of thousands of ideas applied to real-life or potentially to industry. Certainly, if you need to design a tri-section of objects this is the tool you need.

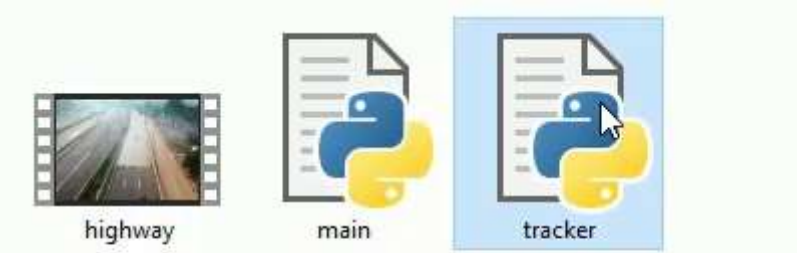
What do we need?

In this tutorial we will use 3 files.:

1 The video of the highway we will use to count the vehicles

2 tracker files. This has already been written and you can simply download it

3 main file. Write me in real-time and we will proceed step by step with the integration of the libraries



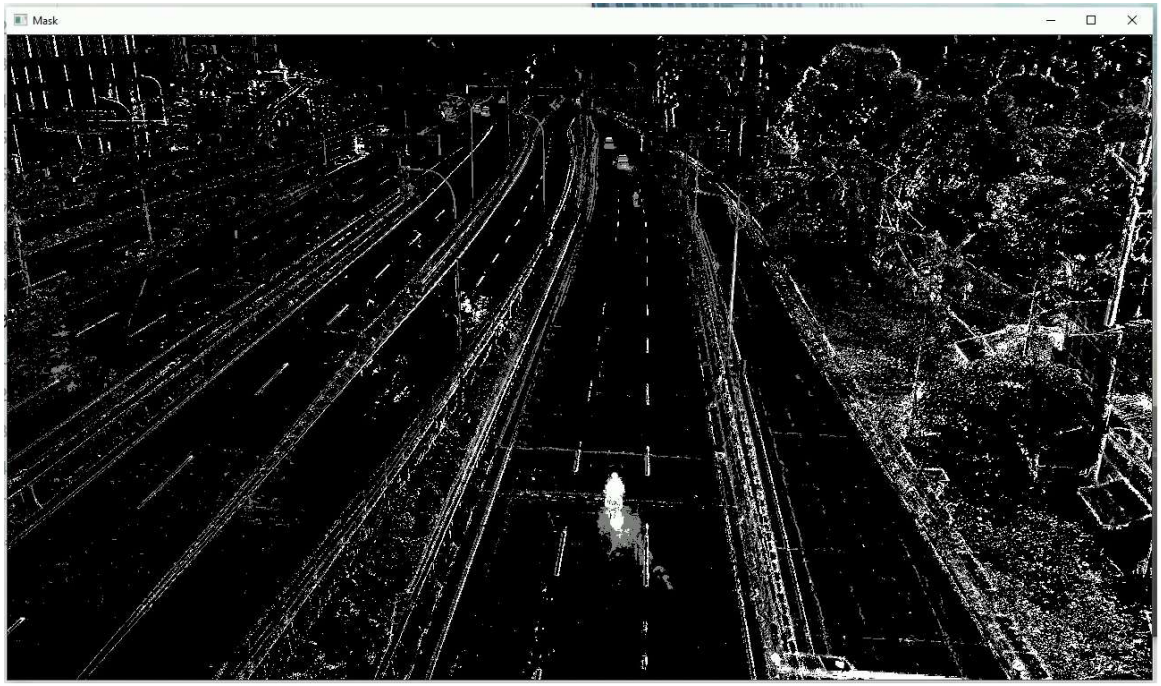
Object detection

First we need to call the **highway.mp4** file and create a mask

```
cap = cv2.VideoCapture("highway.mp4")

# Object detection from Stable camera
object_detector = cv2.createBackgroundSubtractorMOG2()

while True:
    ret, frame = cap.read()

    # 1. Object Detection
    mask = object_detector.apply(frame)
```

As you can see in the example code we also used the createBackgroundSubtractorMOG2 function which Returns the “background ratio” parameter of the algorithm and then create the mask.

This is a first result:

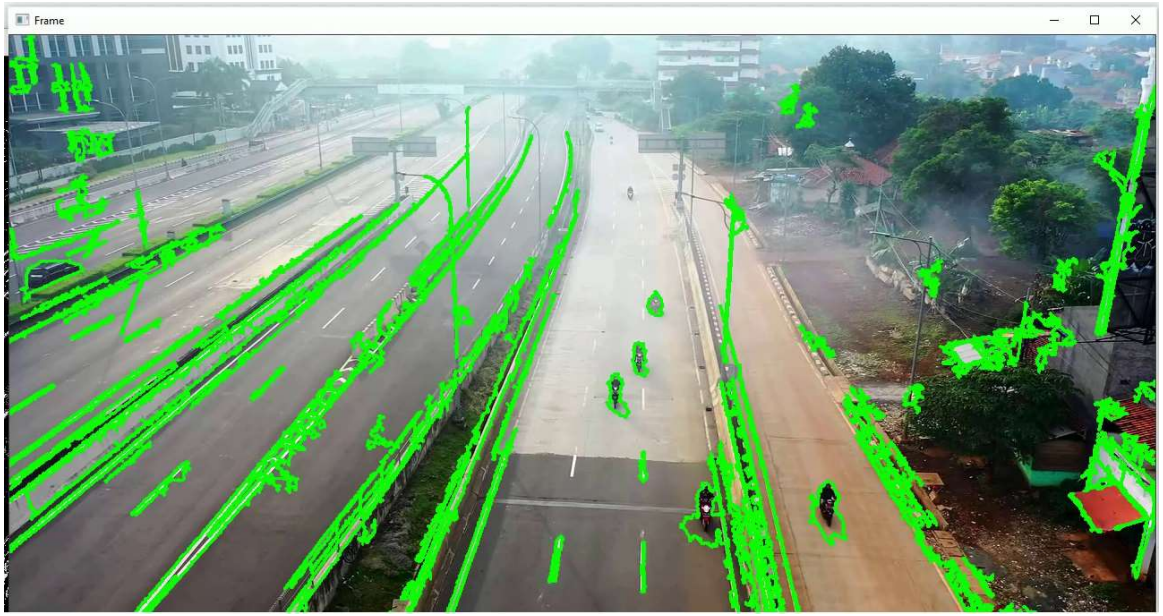


As you can see, however, there is a lot of noise in the image. So let’s improve the extraction by removing all the smaller elements and focus our attention on objects that are larger than a certain area.

```
        _, mask = cv2.threshold(mask, 254, 255, cv2.THRESH_BINARY)
        contours, _ = cv2.findContours(mask, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
        for cnt in contours:
            # Calculate area and remove small elements
```

```
            area = cv2.contourArea(cnt)
            if area > 100:
            #Show image
```

Drawing the contours with OpenCV's **cv2.drawContours** function we obtain this result. You won't need to use this function, consider it as a debug of a first result



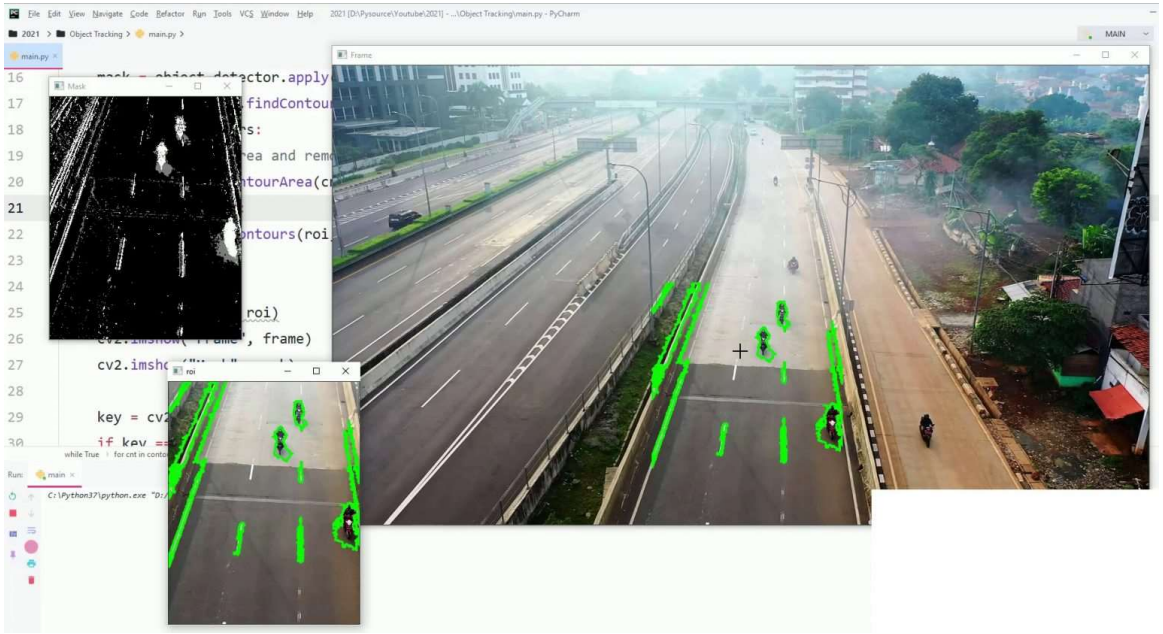
We define a Region of interest

For the purpose of this tutorial, it is not important to analyze the entire window. We are only interested in counting all the vehicles that pass at a certain point, for this reason, we must define a region of interest ROI and apply the mask only in this area.

```
while True:
```

```
    ret, frame = cap.read()
    height, width, _ = frame.shape

    # Extract Region of interest
    roi = frame[340: 720,500: 800]

    # 1. Object Detection
    mask = object_detector.apply(roi)
    _, mask = cv2.threshold(mask, 254, 255, cv2.THRESH_BINARY)
    contours, _ = cv2.findContours(mask, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

    for cnt in contours:
        # Calculate area and remove small elements
        area = cv2.contourArea(cnt)
        if area > 100:
            cv2.drawContours(roi, [cnt], -1, (0, 255, 0), 2)
```

Already in the image you can see a good first result.



The function cv2.createBackgroundSubtractorMOG2 was added at the beginning without defining parameters, now let's see how to further improve our result. **history** is the first parameter, in this case, it is set to 100 because the camera is fixed. var **Threshold** instead is 40 because the lower the value the greater the possibility of making false positives. In this case, we are only interested in the larger objects.

```
1.  # Object detection from Stable camera
2.  object_detector = cv2.createBackgroundSubtractorMOG2(history=100,
varThreshold=40)
```

Draw the box around the object

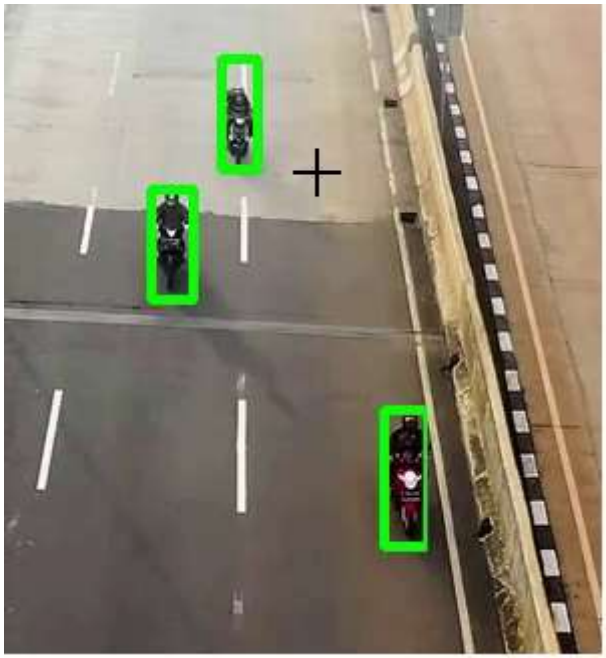
Before proceeding with the rectangle we do a further cleaning of the image. To do this, the **threshold** function comes in handy. Starting from our mask we tell it that we want to show only the white or black values so by writing "254, 255" only the values between 254 and 255 will be considered.

```
1.  _, mask = cv2.threshold(mask, 254, 255, cv2.THRESH_BINARY)
```

We then insert the coordinates of the found object into the if condition and draw the rectangle

```
1.      x, y, w, h = cv2.boundingRect(cnt)
2.      cv2.rectangle(roi, (x, y), (x + w, y + h), (0, 255, 0), 3)
```

This is the final result



As you can see, we have everything you need to proceed with object tracking.

Object Tracking

We now simply have to import and integrate the tracking functions.

```
from tracker import *

# Create tracker object
tracker = EuclideanDistTracker()
```

Once the object has been created, we must therefore take each position of the bounding box and insert them in a single array.

```
detections.append([x, y, w, h])
```

By showing the result on the screen you can see how all the lanes that pass through our ROI are identified and their positions inserted in a specific array. Obviously, the more motorcycles identified the larger our array will be

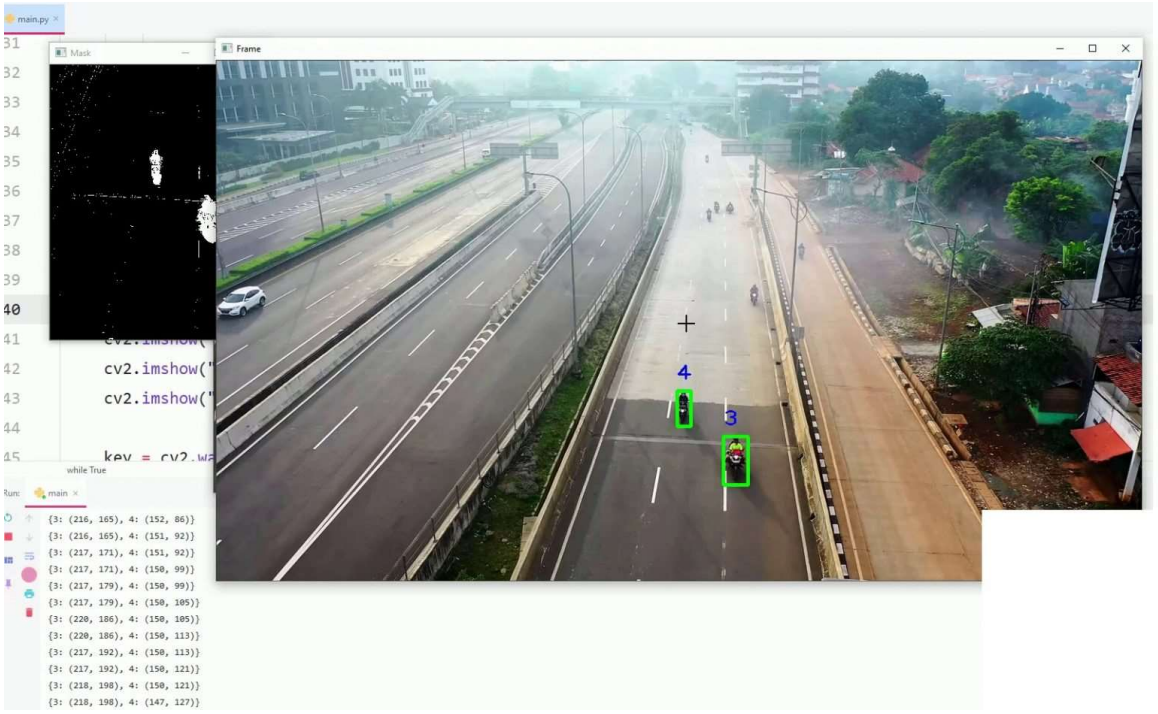
Associate unique ID to the object

Let's now pass our array with positions to **tracker.update()**. We will again get an array with the potions but in addition, a unique id will be assigned for each object.

As you can see from the code we can analyze everything with a for a loop. At this point we just have to draw the rectangle and show the vehicle ID.

```
1.      # 2. Object Tracking
2.      boxes_ids = tracker.update(detections)
3.      for box_id in boxes_ids:
4.          x, y, w, h, id = box_id
5.          cv2.putText(roi, str(id), (x, y - 15),
cv2.FONT_HERSHEY_PLAIN, 2, (255, 0, 0), 2)
6.          cv2.rectangle(roi, (x, y), (x + w, y + h), (0, 255, 0), 3)
```

In the image you can see the result



Conclusions

As you can also see from the video we have obtained the result that we set ourselves at the beginning of this tutorial.

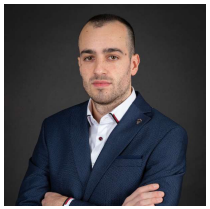
However, you must consider it as an exercise or starting point because on this topic there is a lot to say and the aim of this tutorial was only to make you understand the principle of object tracking.

If you want to integrate Object Tracking into your project, you should use more reliable and advanced object detection methods, as well as tracking methods.

For this purpose, I have recorded a full video course focused on Object Detection and Object Tracking, where you can learn the proper way to detect and track objects.

You can find it here: Object Detection (Opencv & Deep Learning) (https://pysource.com/object-detection-opencv-deep-learning-video-course/)

object_tracking.zip (https://pysource.com/wp-content/uploads/2021/01/object_tracking.zip) Download (https://pysource.com/wp-content/uploads/2021/01/object_tracking.zip)



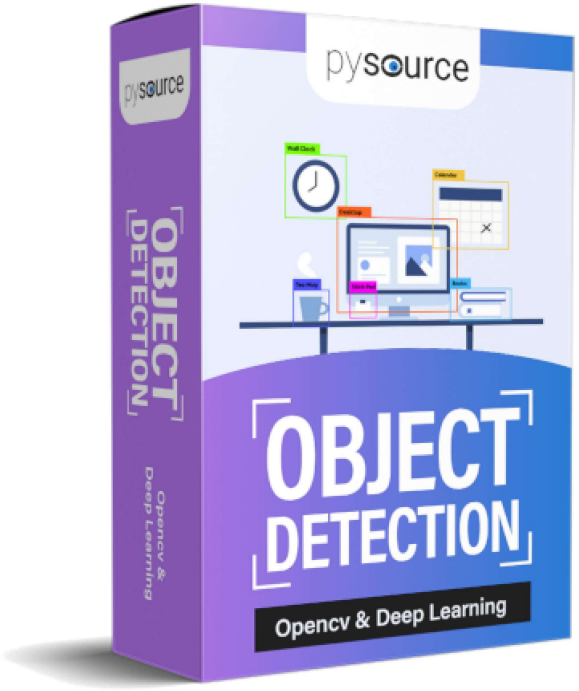
Sergio Canu (https://pysource.com/author/admin/)
Hi there, I'm the founder of Pysource.
I'm a Computer Vision Consultant, developer and Course instructor.
I help Companies and Freelancers to **easily and efficiently build Computer Vision Software**.

(https://www.linkedin.com/company/pysource)

(https://www.youtube.com/channel/UC5hHNks012Ca2o_MPLRUuJw)

(https://pysource.com/object-detection-opencv-deep-learning-video-course/)

(https://pysource.com/object-detection-opencv-deep-learning-video-course/)



Detect and Track any Object (Full Videocourse)

You can Build Software to detect and track any Object even if you have a basic programming knowledge.

More info

(https://pysource.com/object-detection-opencv-deep-learning-video-course/)

(https://pysource.com/object-detection-opencv-deep-learning-video-course/)

(https://pysource.com/object-detection-opencv-deep-learning-video-course/)

Learn to build Computer Vision Software

Join the FREE Workshop where I'll teach you how to build a Computer Vision Software to **detect and track any object.**

More info

(https://pysource.com/blueprint-workshop-signup/)