KB

lunes, 4 de febrero de 2019 19:06

Last timestamp: 04/02/2019

Links:

- TensorFlow Object Counting API: https://github.com/ahmetozlu/tensorflow_object_counting_api

Docs

lunes, 4 de febrero de 2019 19:18

Last timestamp: 04/02/2019

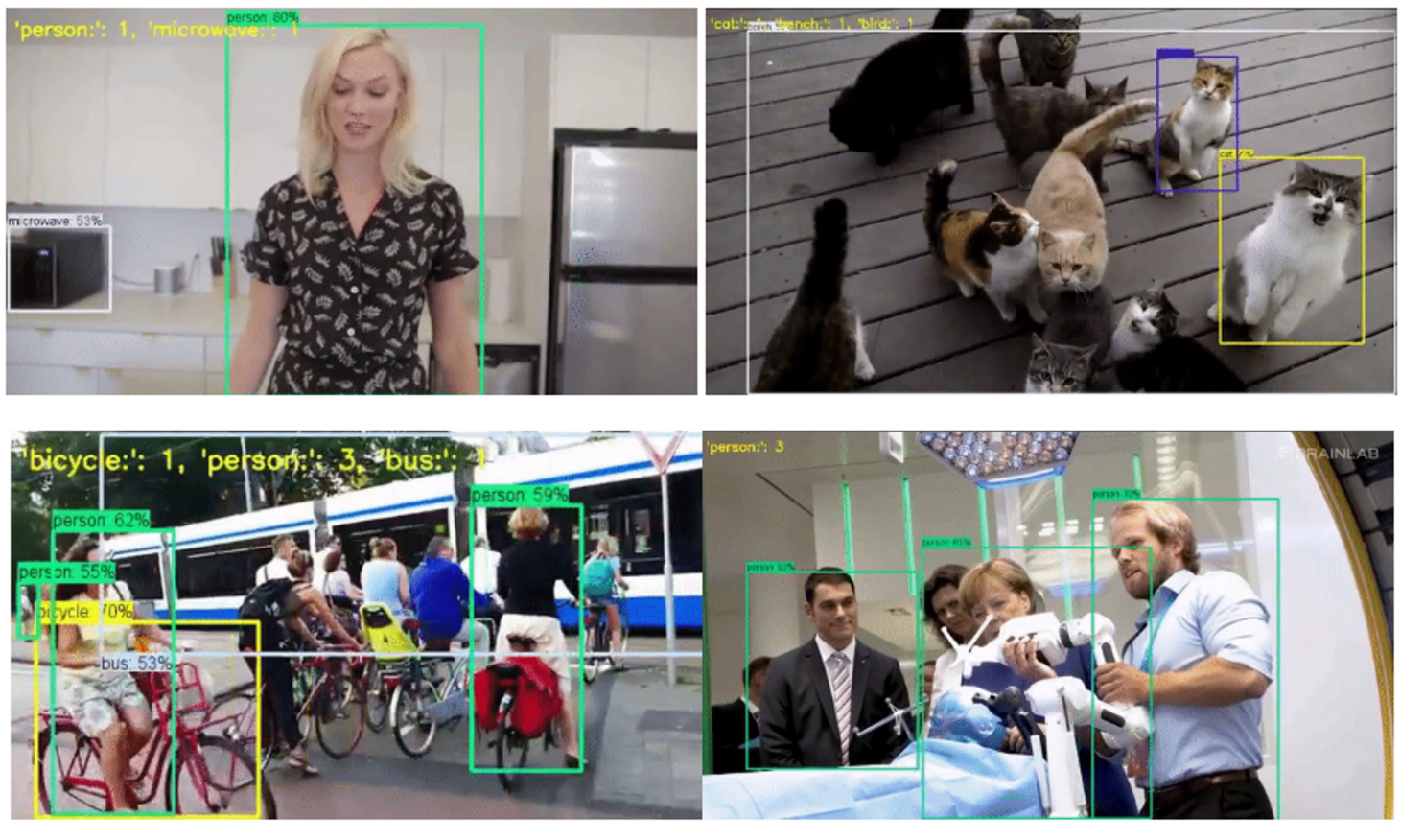
Sources:

- GitHub - TensorFlow Object Counting API: https://github.com/ahmetozlu/tensorflow_object_counting_api

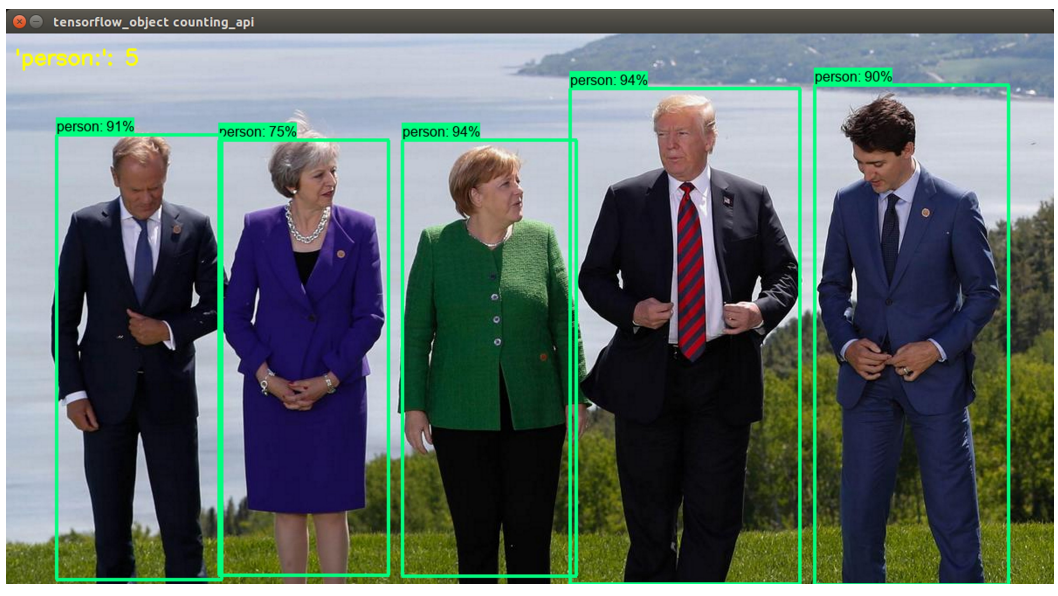
TensorFlow Object Counting API

- The TensorFlow Object Counting API is an open source framework built on top of TensorFlow that makes it easy to develop object counting systems.
- The API can:
  - **Cumulative Counting Mode:**

  - **Real-Time Counting Mode:**



  - **Object Counting On Single Image:**



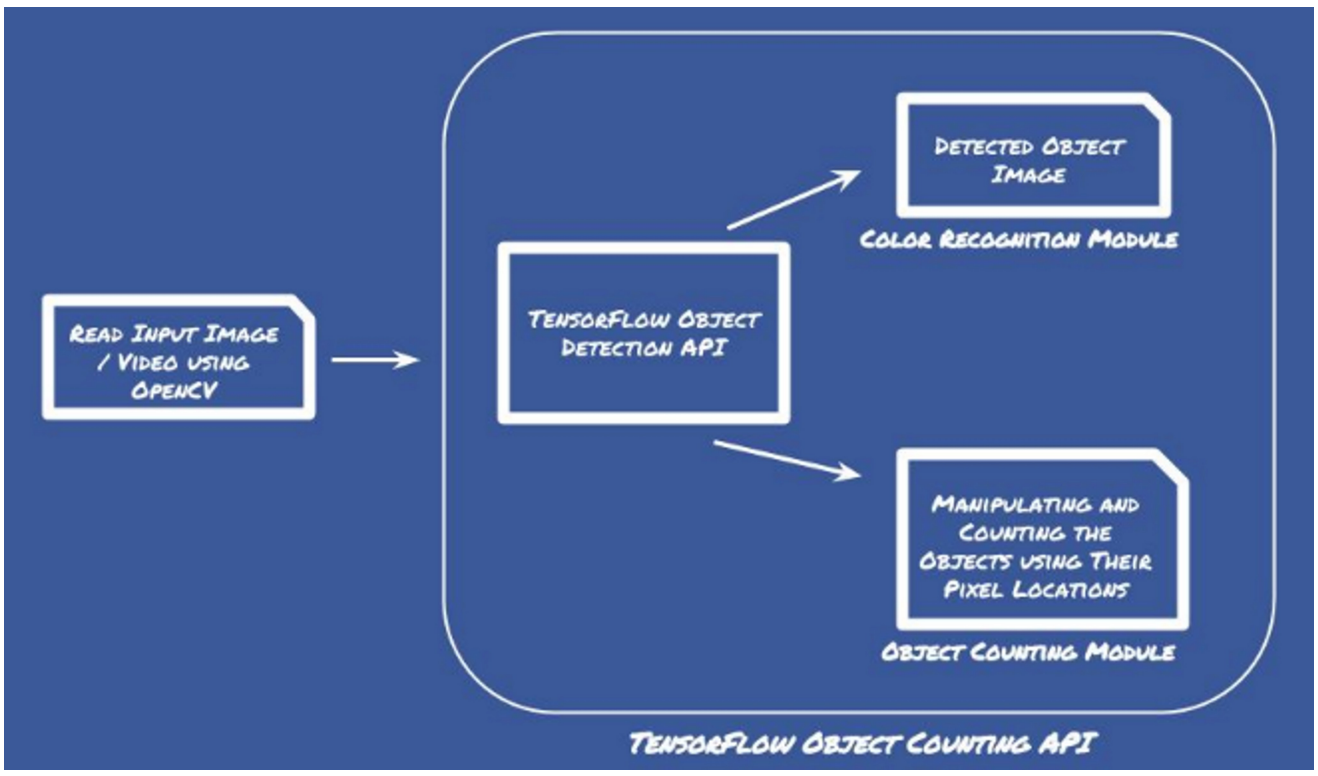
General Capabilities of The TensorFlow Object Counting API:

- Detect just the targeted objects.
- Detect all the objects.
- Count just the targeted objects.
- Count all the objects.
- Predict color of the targeted objects.
- Predict color of all the objects.
- Predict speed of the targeted objects.
- Predict speed of all the objects.
- Print out the detection-counting result in a .csv file as an analysis report.
- Save and store detected objects as new images under detected_object folder.
- Select, download and use state of the art models that are trained by Google Brain Team (Detection Model Zoo).
- Use your own trained models or a fine-tuned model to detect specific object(s).
- Save detection and counting results as a new video or show detection and counting results in real time.
- Process images or videos depending on your requirements.

General Architectural Design Features of The TensorFlow Object Counting API:

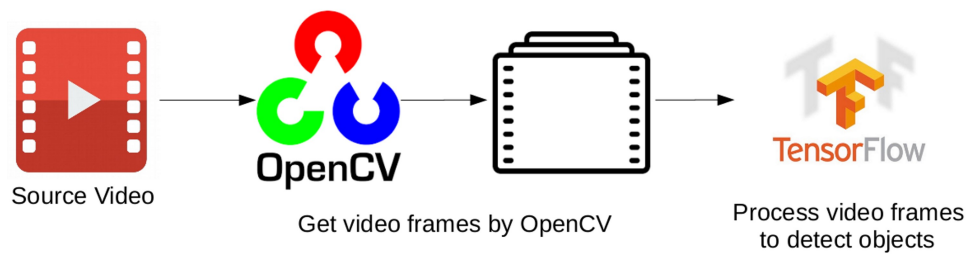
- Lightweight, runs in real-time.
- Scalable and well-designed framework, easy usage.
- Gets "Pythonic Approach" advantages.
- It supports REST Architecture and RESTful Web Services.

System Architecture



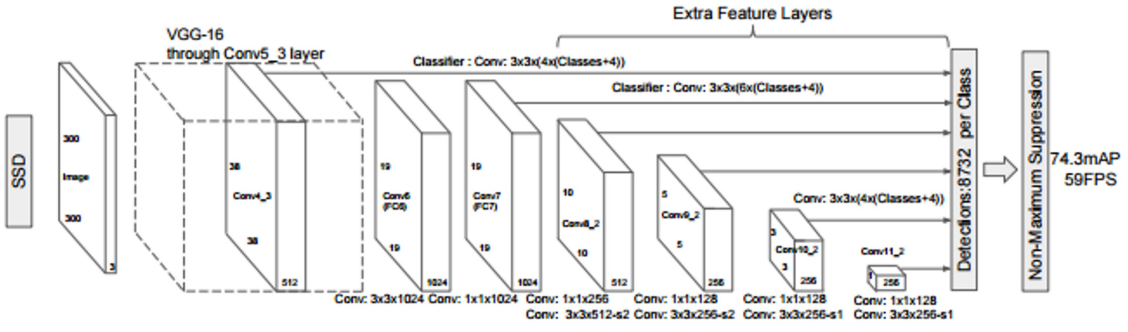
- Object detection and classification have been developed on top of *TensorFlow Object Detection API*.
- Object color prediction has been developed using OpenCV via K-Nearest Neighbors Machine Learning Classification Algorithm is Trained Color Histogram Features.

Tracker



- Source video is read frame by frame with OpenCV.
- Each frames is processed by "*SSD with Mobilenet (ssd_mobilenet_v1_coco_2017_11_17*)" model is developed on TensorFlow.
- This is a loop that continue working till reaching end of the video.

Models



- By default use an "SSD with Mobilenet" model in the project.
- Other models (TF Detection Model Zoo) that can be run out-of-the-box with varying speeds and accuracies.
- You can perform transfer learning on trained TensorFlow models to build your custom object counting systems.

Installation

lunes, 4 de febrero de 2019 19:36

Last timestamp: 04/02/2019

Sources:

- https://github.com/ahmetozlu/tensorflow_object_counting_api#installation

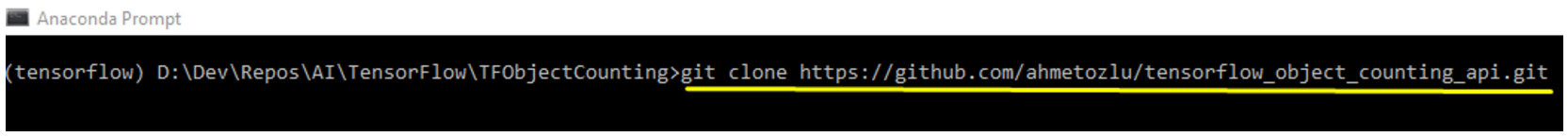
Dependencies:

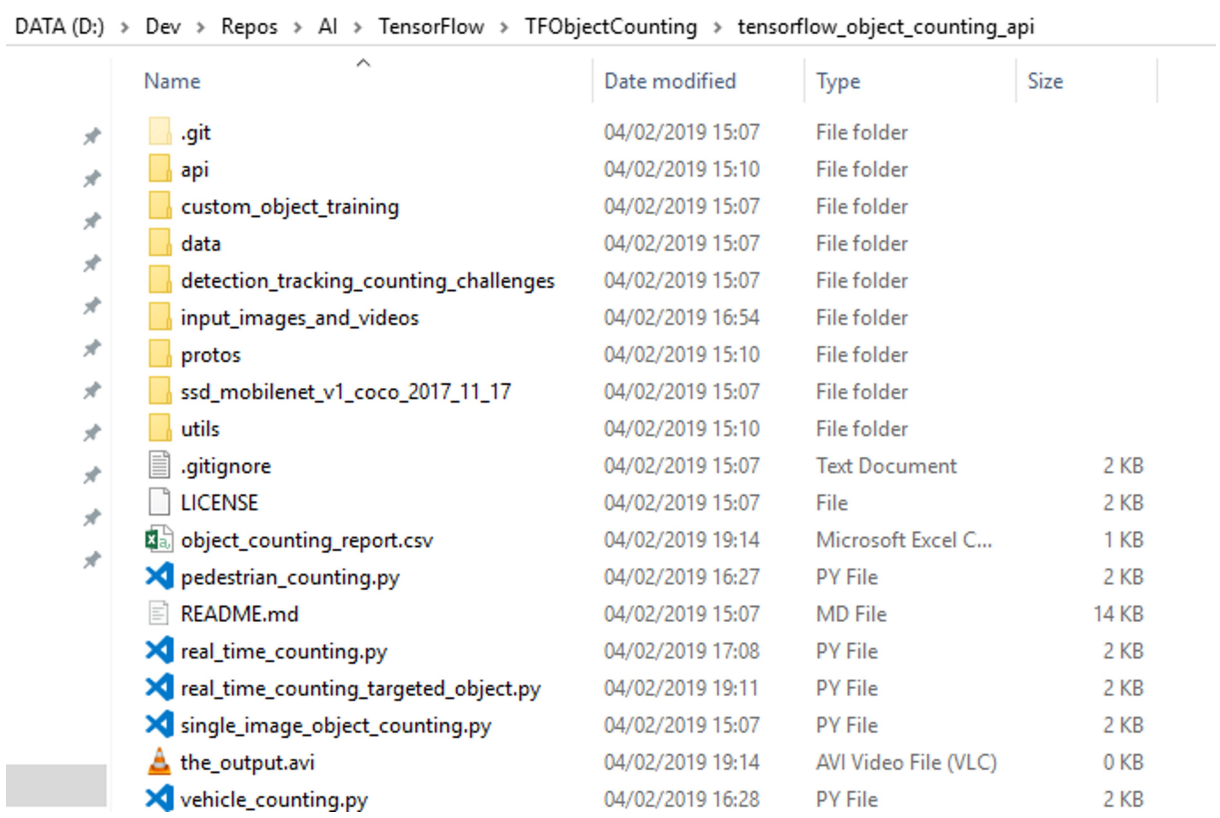
- TensorFlow Object Detection API.
- Protobuf >= 3.0.0.
- Python-tk.
- Pillow >= 1.0.
- Lxml.
- tf Slim (which is included in the "tensorflow/models/research/" checkout).
- Jupyter notebook.
- Matplotlib.
- Tensorflow.
- cython.
- Contextlib2.
- cocoapi.

TensorFlow Object Detection API have to be installed to run TensorFlow Object Counting API.

My Steps:

- I've cloned the project to folder *<TensorFlow>/<TFObjectCounting>/.*





- Run the experiments in the project.

PoCs

martes, 5 de febrero de 2019 10:29

PoC #1

martes, 5 de febrero de 2019 10:22

Last timestamp: 05/02/2019

Sources:

- https://github.com/ahmetozlu/tensorflow_object_counting_api#11-for-detecting-tracking-and-counting-the-pedestrians-with-disabled-color-prediction

Objective: Usage of "Cumulative Counting Mode" - For detecting, tracking and counting the pedestrians with disabled color prediction

Configurations:

- **Model:** *ssd_mobilenet_v1_coco_2017_11_17*
- **Input video:** *./input_images_and_videos/pedestrian_survaillance.mp4*
- **Command:** *python pedestrian_counting.py*
- **Code:**

```
# Imports
import tensorflow as tf
from distutils.version import StrictVersion

# Object detection imports
from utils import backbone
from api import object_counting_api

if StrictVersion(tf.__version__) < StrictVersion('1.4.0'):
  raise ImportError('Please upgrade your tensorflow installation to v1.4.* or later!')

input_video = "./input_images_and_videos/pedestrian_survaillance.mp4"

# By default I use an "SSD with Mobilenet" model here. See the detection model zoo (https://github.c
detection_graph, category_index = backbone.set_model('ssd_mobilenet_v1_coco_2017_11_17')

fps = 30 # change it with your input video fps
width = 626 # change it with your input video width
height = 360 # change it with your input vide height
is_color_recognition_enabled = 0 # set it to 1 for enabling the color prediction for the detected ob
roi = 385 # roi line position
deviation = 1 # the constant that represents the object counting area

object_counting_api.cumulative_object_counting_x_axis(input_video, detection_graph, category_index,
  is_color_recognition_enabled, fps, width, height, roi, deviation) # counting all the objects
```

Results:

- **Output Video:** *\pocs\poc_01\poc_output_01_v1.avi*

https://www.youtube.com/watch?v=TSKcESpJYpI

- **DAL's Notes:**
  - Como se puede ver en el video, el API detecta bien las personas, pero NO cuenta las personas al pasar por la línea de marcación puesta (ROI) para este propósito. Se puede observar que el contador de personas (Detected Pedestrians) sigue en CERO.
  - Investigando un poco más de cómo hace el API para contar los objetos que pasan por el ROI me encontré estos dos archivos ubicados en la carpeta "*\tensorflow_object_counting_api\utils\object_counting_module*" del API de Object Counting:
    - *object_counter_x_axis.py*
    - *object_counter.py*

```
if (abs(((bottom+top)/2)-roi_position) < deviation):
  is_vehicle_detected.insert(0,1)
  update_csv = True
  image_saver.save_image(crop_img) # save detected object image
```

  - El código de está PoC usa el la validación del archivo "object_counter_x_axis.py" (ya que se cuentan las personas que pasen sobre el eje X). Al poner un print en el código se ve lo siguiente:

```python
abs_val = abs(((bottom+top)/2)-roi_position)
print(f"[INFO] abs_val - object_counter_x_axis.py = {abs_val}")

if (abs_val < deviation):
    is_vehicle_detected.insert(0,1)
    update_csv = True
    image_saver.save_image(crop_img) # save detected object image
```

Select Anaconda Prompt

```
[INFO] abs_val - object_counter_x_axis.py = 187.38067626953125
writing frame
[INFO] abs_val - object_counter_x_axis.py = 186.9838809967041
[INFO] abs_val - object_counter_x_axis.py = 147.61870861053467
writing frame
[INFO] abs_val - object_counter_x_axis.py = 186.26948535442352
[INFO] abs_val - object_counter_x_axis.py = 148.9818125963211
writing frame
[INFO] abs_val - object_counter_x_axis.py = 185.65738379955292
[INFO] abs_val - object_counter_x_axis.py = 147.6096749305725
writing frame
[INFO] abs_val - object_counter_x_axis.py = 185.04642486572266
[INFO] abs_val - object_counter_x_axis.py = 149.1403365135193
writing frame
[INFO] abs_val - object_counter_x_axis.py = 185.88525354862213
[INFO] abs_val - object_counter_x_axis.py = 151.5386551618576
writing frame
[INFO] abs_val - object_counter_x_axis.py = 155.91089189052582
[INFO] abs_val - object_counter_x_axis.py = 186.82374238967896
writing frame
[INFO] abs_val - object_counter_x_axis.py = 188.04478585720062
[INFO] abs_val - object_counter_x_axis.py = 150.53204357624054
writing frame
[INFO] abs_val - object_counter_x_axis.py = 188.4408140182495
[INFO] abs_val - object_counter_x_axis.py = 155.70447444915771
writing frame
[INFO] abs_val - object_counter_x_axis.py = 183.42763662338257
```

- Adicionalmente de ver que se usa el cálculo de "*object_counter_x_axis.py*", se ve que el valor de la fórmula siempre es mayor a 100, por lo cual NUNCA entra al "*if*" porque la condición "*abs_val < 1 (deviation)*" NO se cumple.

PoC #2

martes, 5 de febrero de 2019 10:22

Last timestamp: 05/02/2019

Sources:

- https://github.com/ahmetozlu/tensorflow_object_counting_api#1-usage-of-cumulative-counting-mode

Objective: Usage of "Cumulative Counting Mode" - For detecting, tracking and counting the vehicles with enabled color prediction

Configurations:

- **Model:** *ssd_mobilenet_v1_coco_2017_11_17*
- **Input video:** *./input_images_and_videos/vehicle_survaillance.mp4*
- **Command:** *python vehicle_counting.py*
- **Code:**

```
# Imports
import tensorflow as tf
from distutils.version import StrictVersion

# Object detection imports
from utils import backbone
from api import object_counting_api

if StrictVersion(tf.__version__) < StrictVersion('1.4.0'):
  raise ImportError('Please upgrade your tensorflow installation to v1.4.* or later!')

input_video = "./input_images_and_videos/vehicle_survaillance.mp4"

# By default I use an "SSD with Mobilenet" model here. See the detection model zoo (https://github.c
detection_graph, category_index = backbone.set_model('ssd_mobilenet_v1_coco_2017_11_17')

fps = 24 # change it with your input video fps
width = 640 # change it with your input video width
height = 352 # change it with your input vide height
is_color_recognition_enabled = 0 # set it to 1 for enabling the color prediction for the detected ob
roi = 200 # roi line position
deviation = 3 # the constant that represents the object counting area

object_counting_api.cumulative_object_counting_y_axis(input_video, detection_graph, category_index,
  is_color_recognition_enabled, fps, width, height, roi, deviation) # counting all the objects
```

Results:

- **Output Video:** *\pocs\poc_02\poc_output_02_v1.avi*

https://www.youtube.com/watch?v=URrp5WXNnlk

- **DAL's Notes:**
  - Como se puede ver en el video, el API detecta bien los carros y a diferenciar de la PoC #1 si cuenta bien los 2 primeros carros. Al pasar el 3er carro (camioneta blanca) el contador pasa a 5.