AI VIETNAM
All-in-One Course
(TA Session)

Object Tracking

Project

Dinh-Thang Duong - TA

Minh-Duc Bui - STA

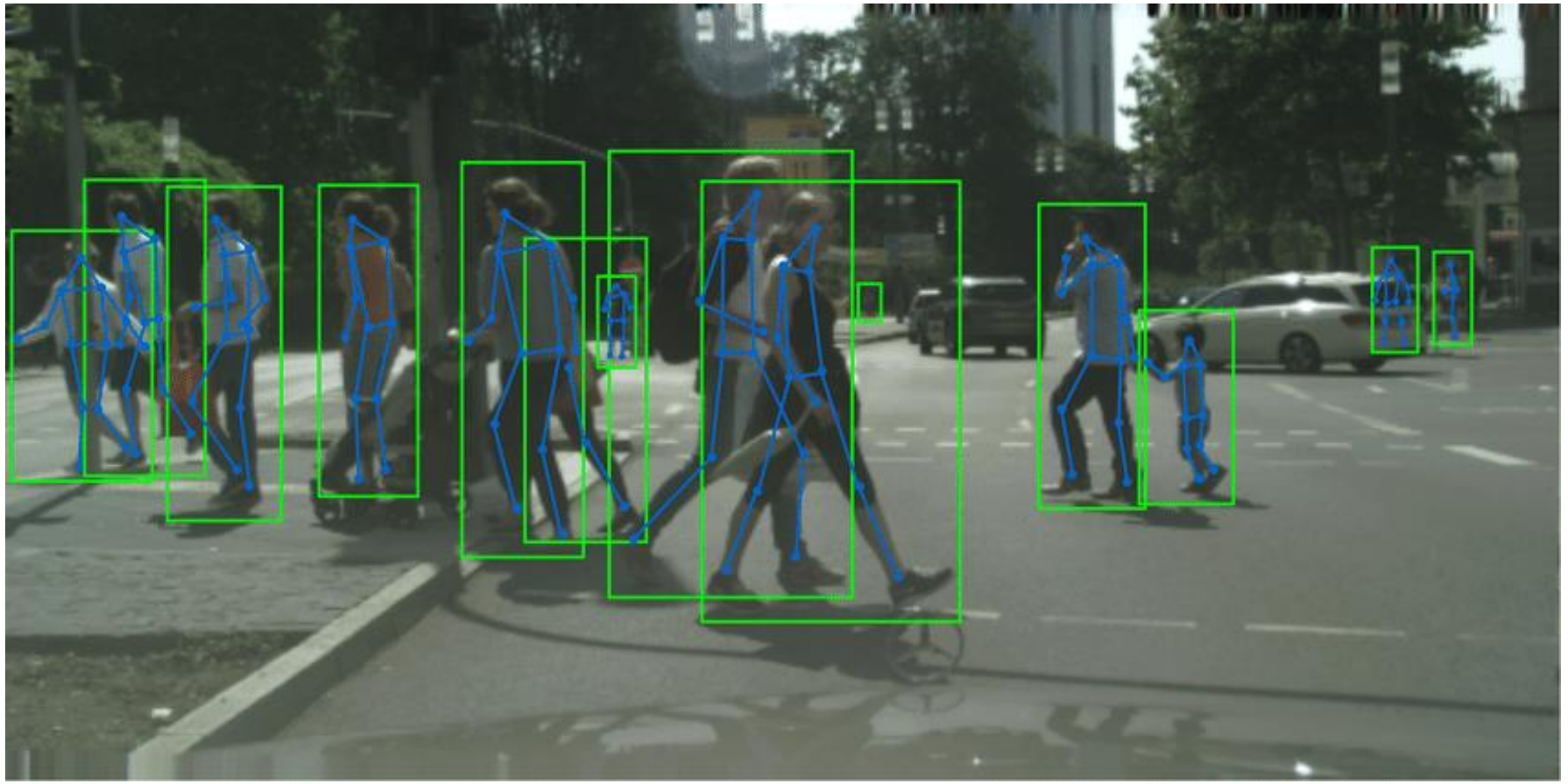
Year 2024

Outline

- Introduction
- Detector
- Tracker
- Question

Introduction

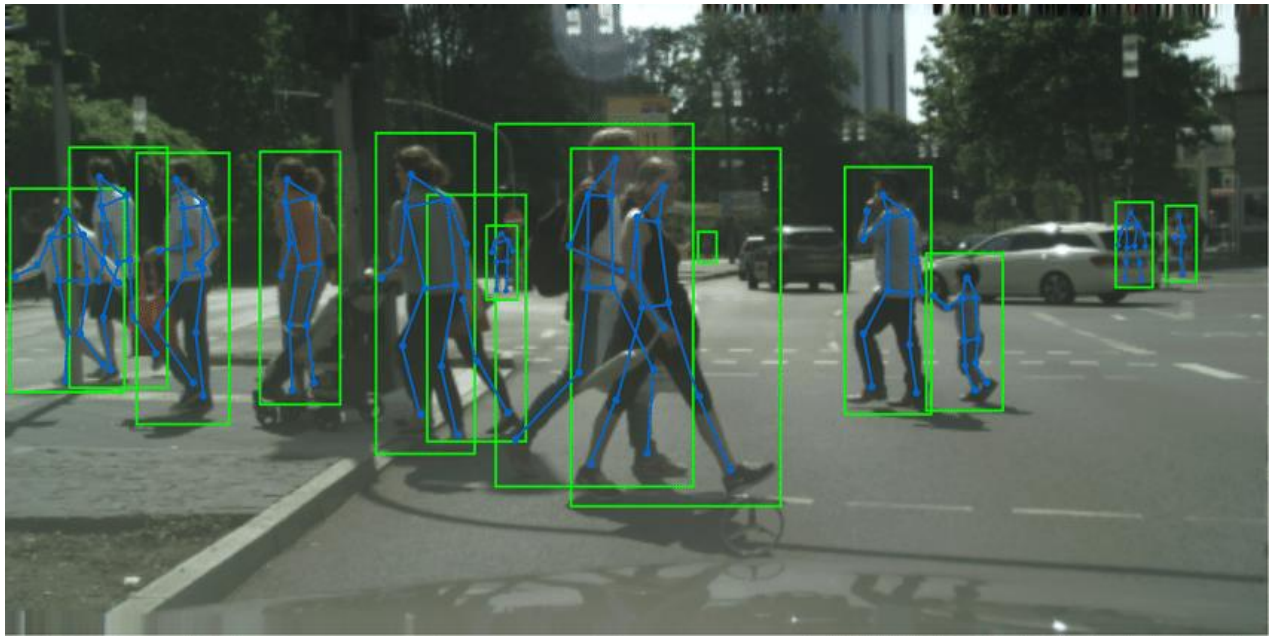
❖ Getting Started



Count the number of people inside a image?

Introduction

❖ Getting Started



In an image, we could count the bounding box

But in a video, counting by counting the bounding box would not be a good approach

Introduction

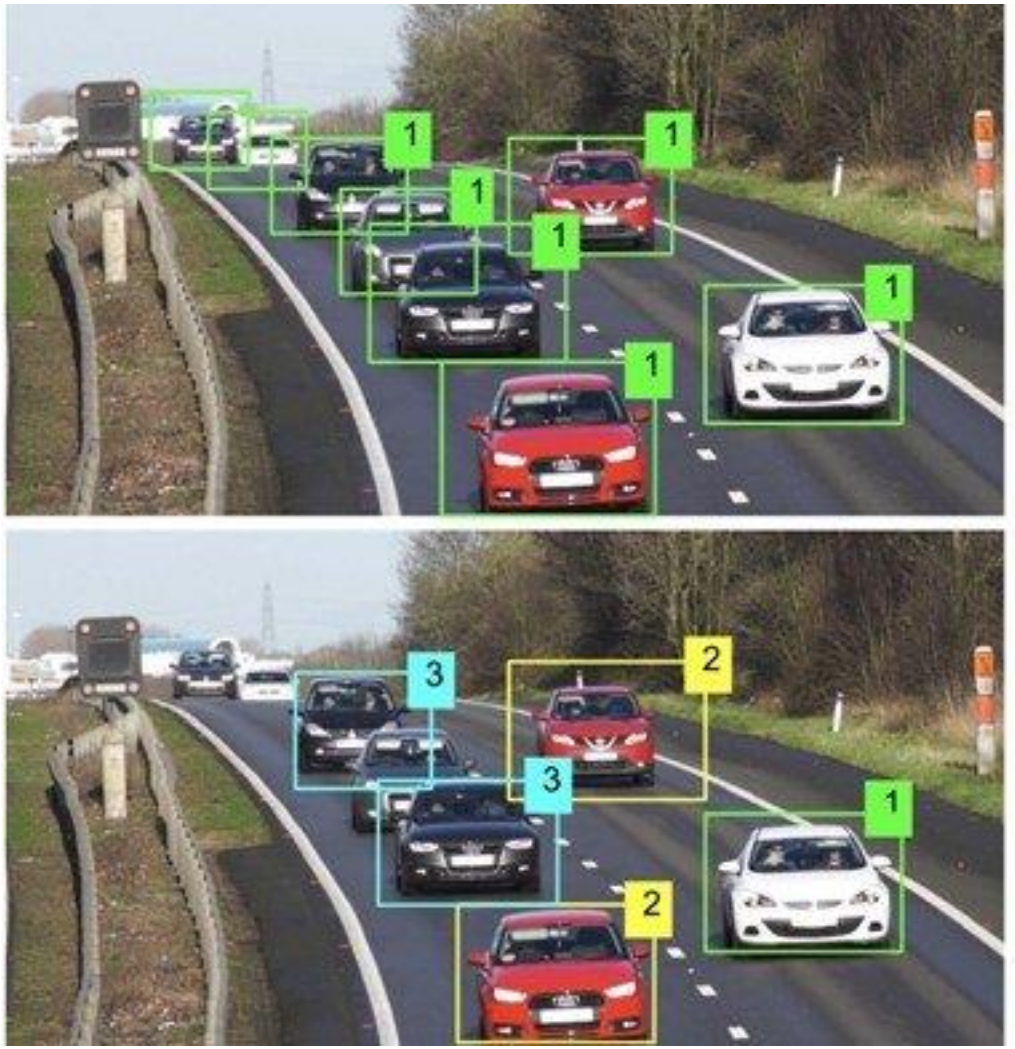
❖ Object tracking

Object Tracking: A computer vision task of detecting and following a moving object (or multiple objects) through sequential frames of a video. This process involves initially identifying the object in the first frame and then continuously tracking its movement, accounting for changes in appearance, scale, and occlusions.

https://trepo.tuni.fi/bitstream/handle/10024/120226/FengYan.pdf;jsessionid=8D1B8C7B34971B7DA0A5A772A1D200F5?sequence=2

Introduction

❖ Getting Started



Typical Object Detection Algorithm

Vs.

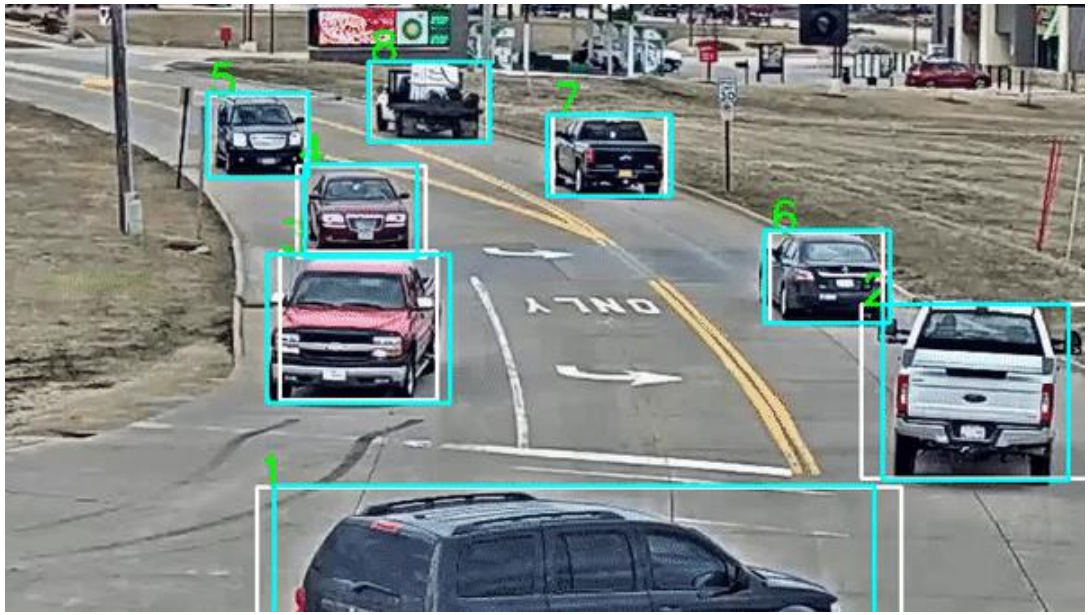
Typical Object Tracking Algorithm

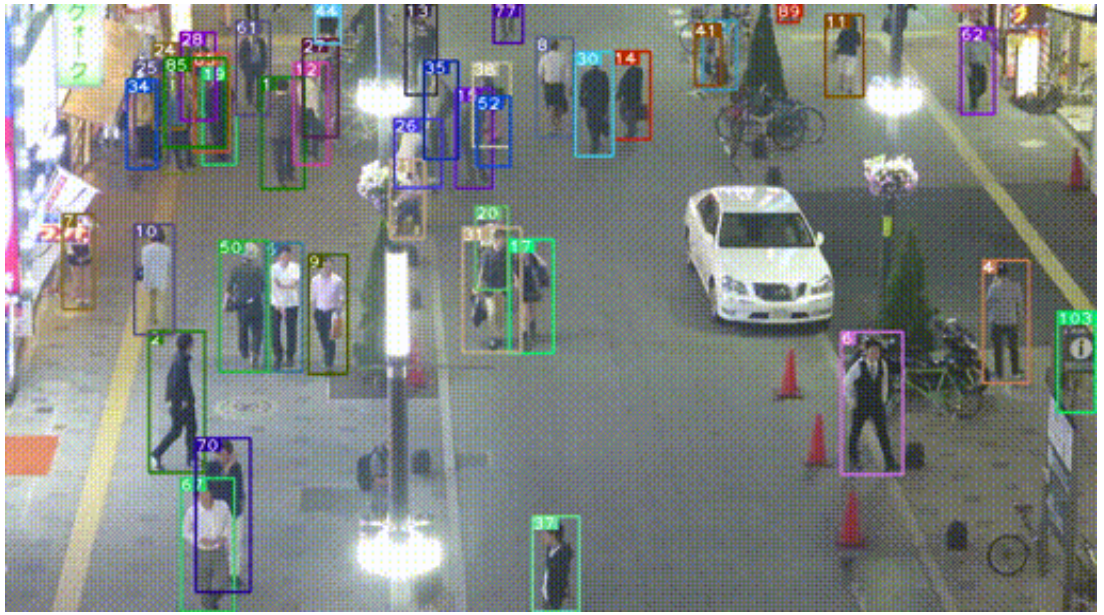
Results (output)

https://viso.ai/deep-learning/object-tracking/

Introduction

❖ Object tracking





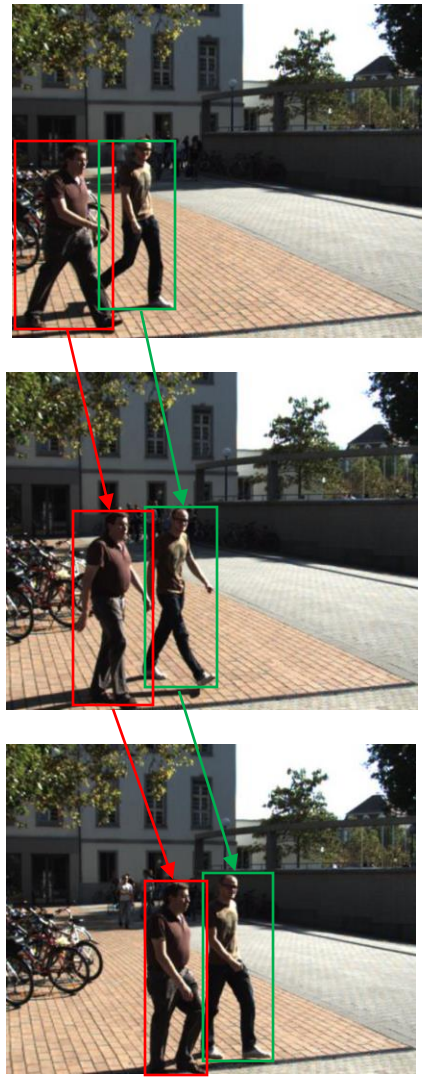
https://www.datature.io/blog/implementing-object-tracking-for-computer-vision
https://viblo.asia/p/sort-deep-sort-mot-goc-nhin-ve-object-tracking-phan-1-Az45bPooZxY

Introduction

❖ Object tracking

Input

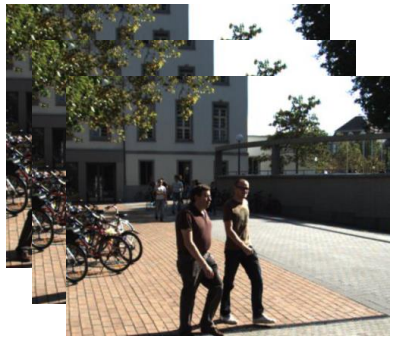
Model



Output

Introduction

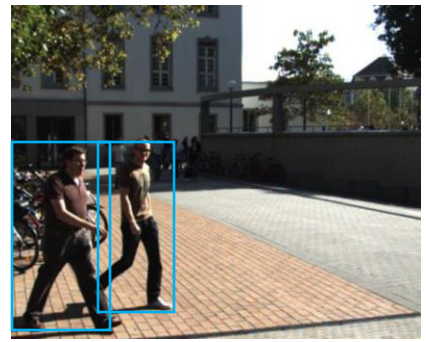
❖ Two-Stage Object Tracking

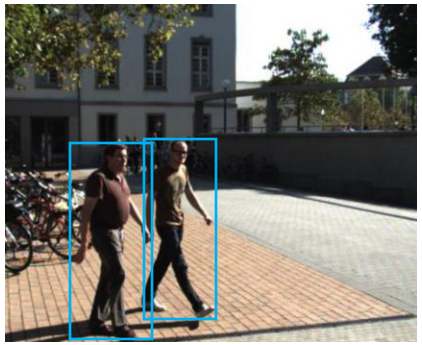


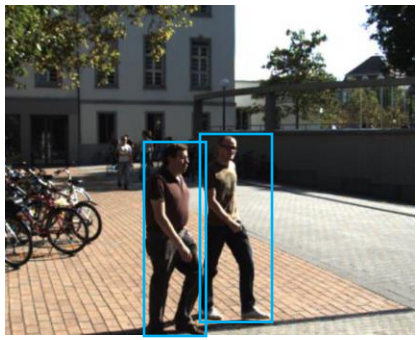
Input

Detector

R-CNN
Faster R-CNN
YOLO
DETR
...



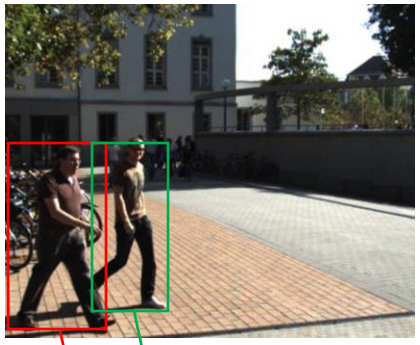


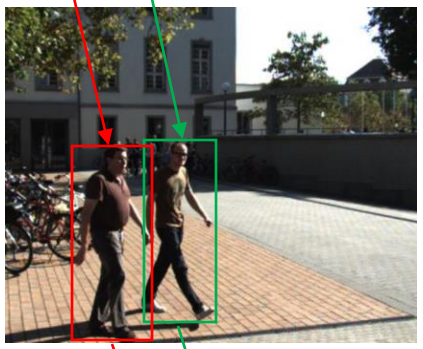


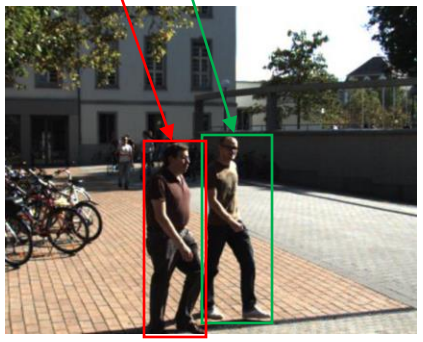
Detections

Tracker

SORT
DeepSORT
BOTSORT
ByteTrack
...







Output

Introduction

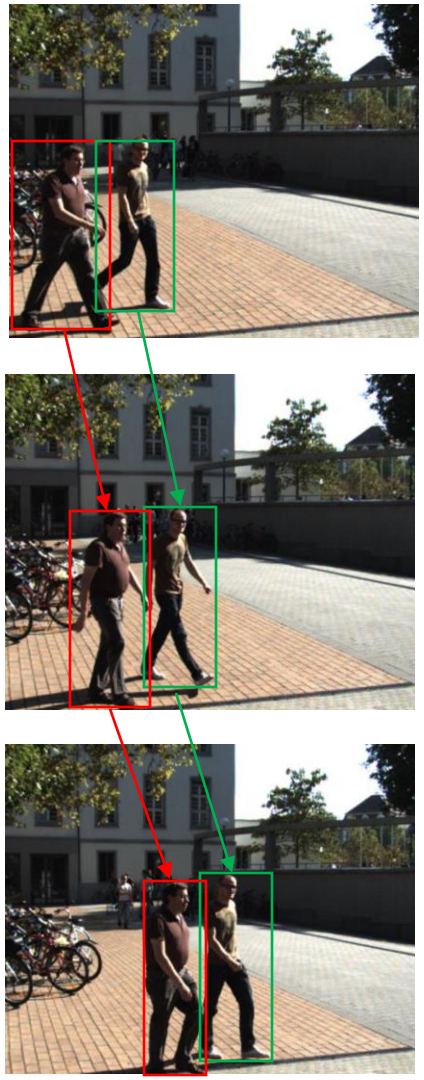
❖ One-Stage Object Tracking

Input

Detection & Tracker

Transformer-based:

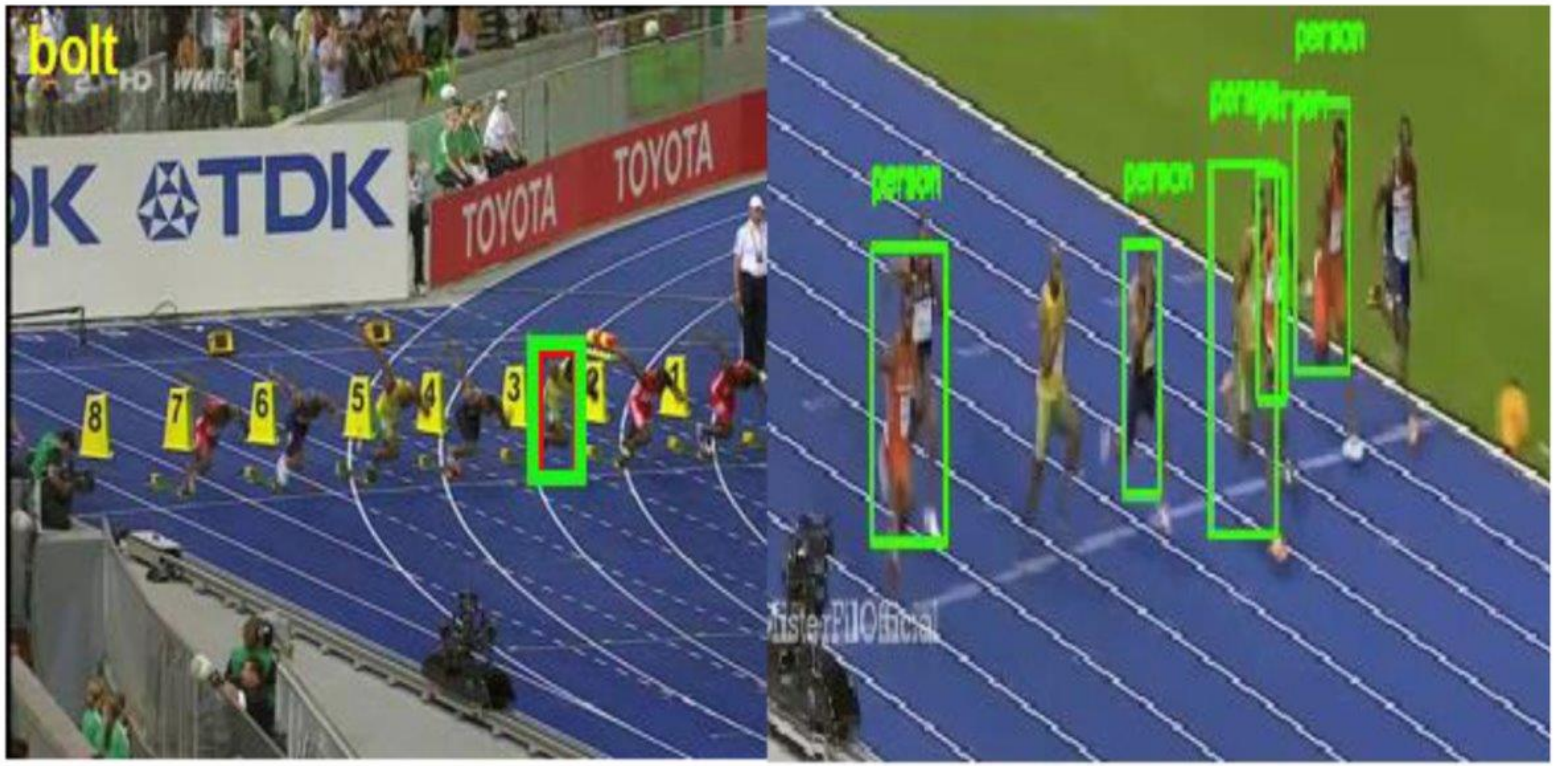
- SwinTrack
- OSTrack
- SeqTrack
- VideoTrack
- ...



Output

Introduction

❖ Single Object and Multi Object Tracking



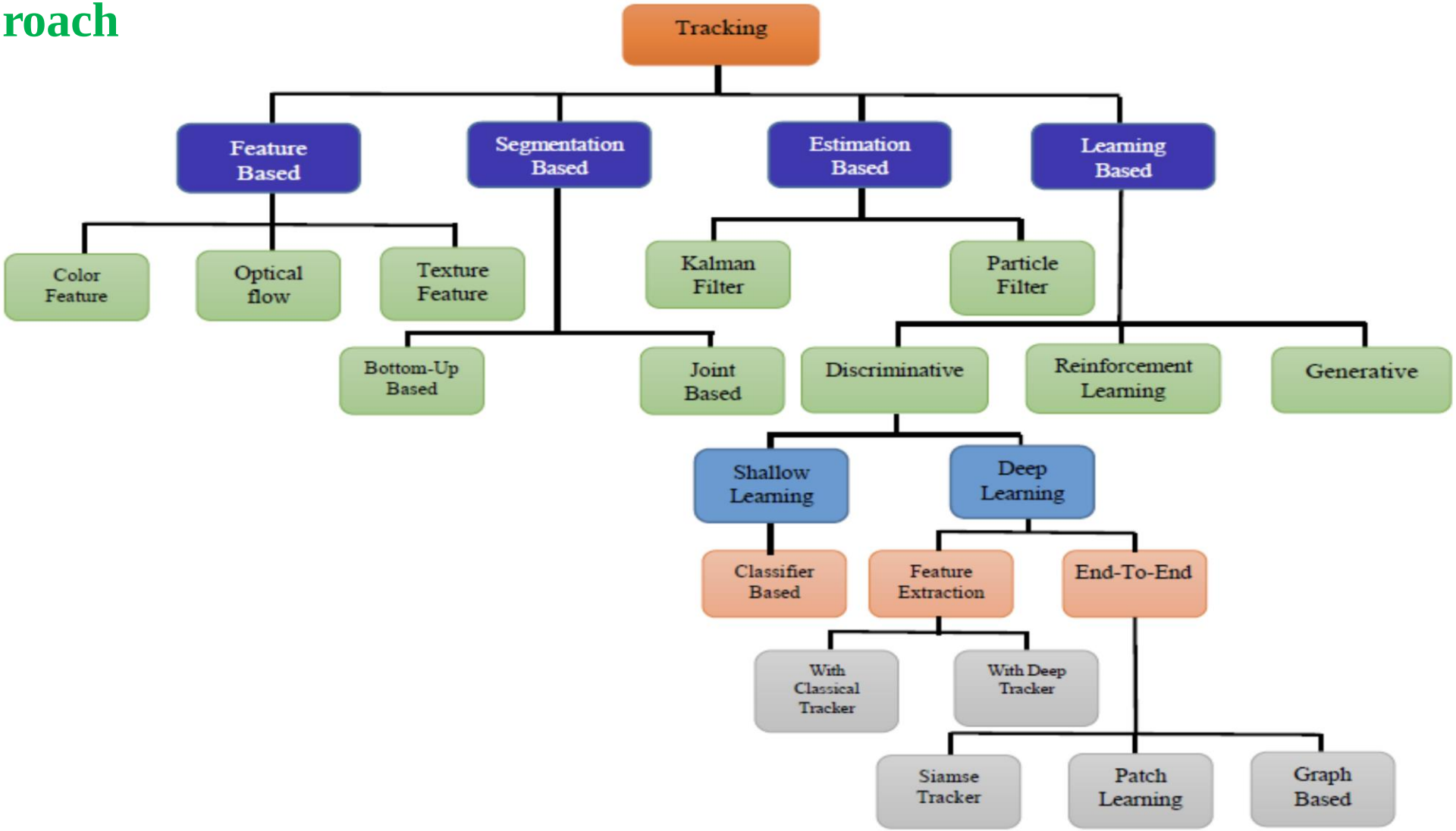
Single Object tracking

Multi Object tracking

Soleimanitaleb, Zahra, and Mohammad Ali Keyvanrad. "Single object tracking: a survey of methods, datasets, and evaluation metrics." arXiv preprint arXiv:2201.13066 (2022).

Introduction

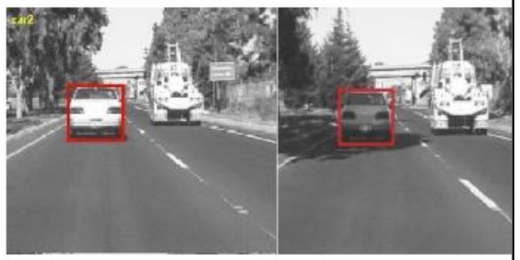
❖ Approach



Soleimanitaleb, Zahra, and Mohammad Ali Keyvanrad. "Single object tracking: a survey of methods, datasets, and evaluation metrics." arXiv preprint arXiv:2201.13066 (2022).

Introduction

❖ Challenges

Challenge	Describe	Example
Illumination Variation	The illumination in the target region is significantly changed.	
Background Clutters	The background near the target has a similar color or texture as the target.	
Low Resolution	The number of pixels inside the ground-truth bounding box is low.	
Scale Variation	The ratio of the bounding boxes of the first frame and the current frame is out of the range.	
Occlusion	The target is partially or fully occluded.	
Change the target position	During the movement, the target may be rotated, deformed, and so on.	
Fast Motion	The motion of the ground truth is large.	

Soleimanitaleb, Zahra, and Mohammad Ali Keyvanrad. "Single object tracking: a survey of methods, datasets, and evaluation metrics." arXiv preprint arXiv:2201.13066 (2022).

Introduction

❖ Challenges

(a) Inter-class occlusion

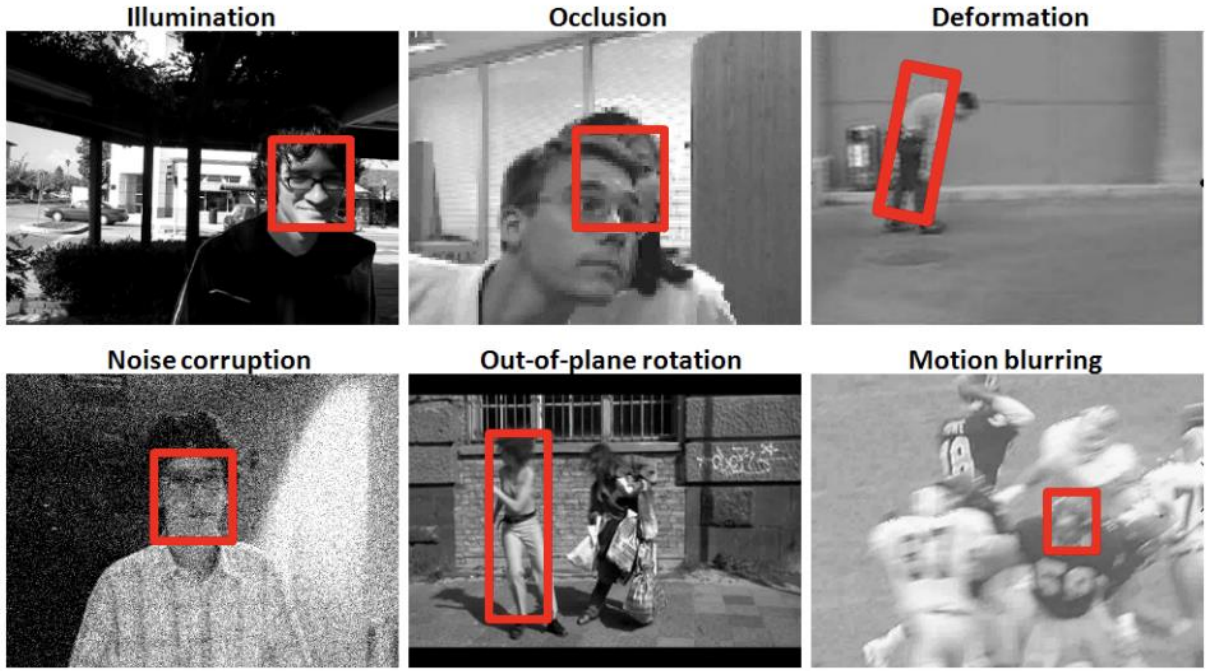
(b) Intra-class occlusion

(c) Scale variance

(d) Complex background

(e) Illumination

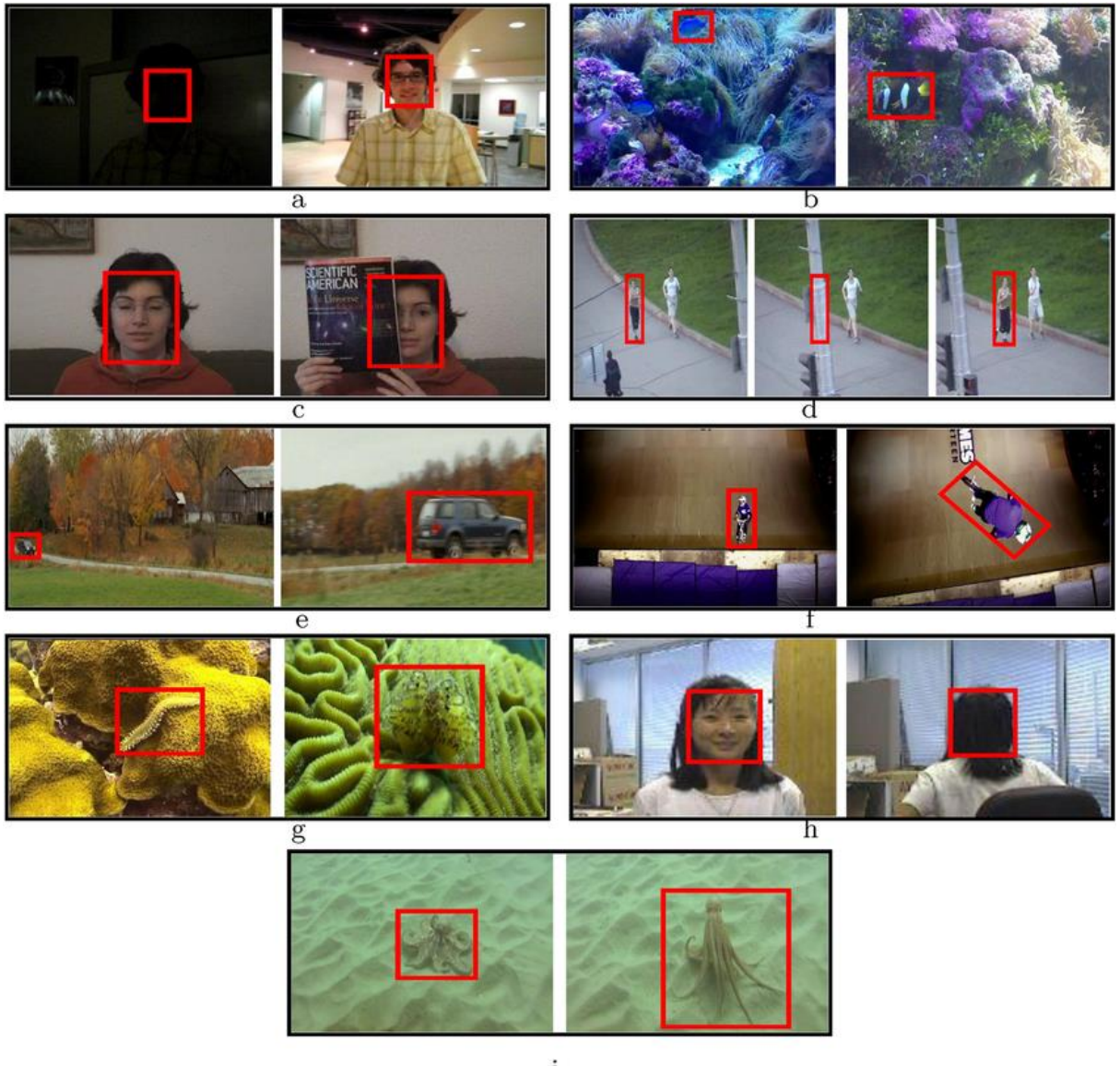
(f) Rainy day



https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9718221
https://www.v7labs.com/blog/object-tracking-guide

Introduction

❖ Challenges



https://www.researchgate.net/figure/Visual-illustration-of-various-challenges-in-object-tracking-Examples-of-a-illumination_fig2_358230702

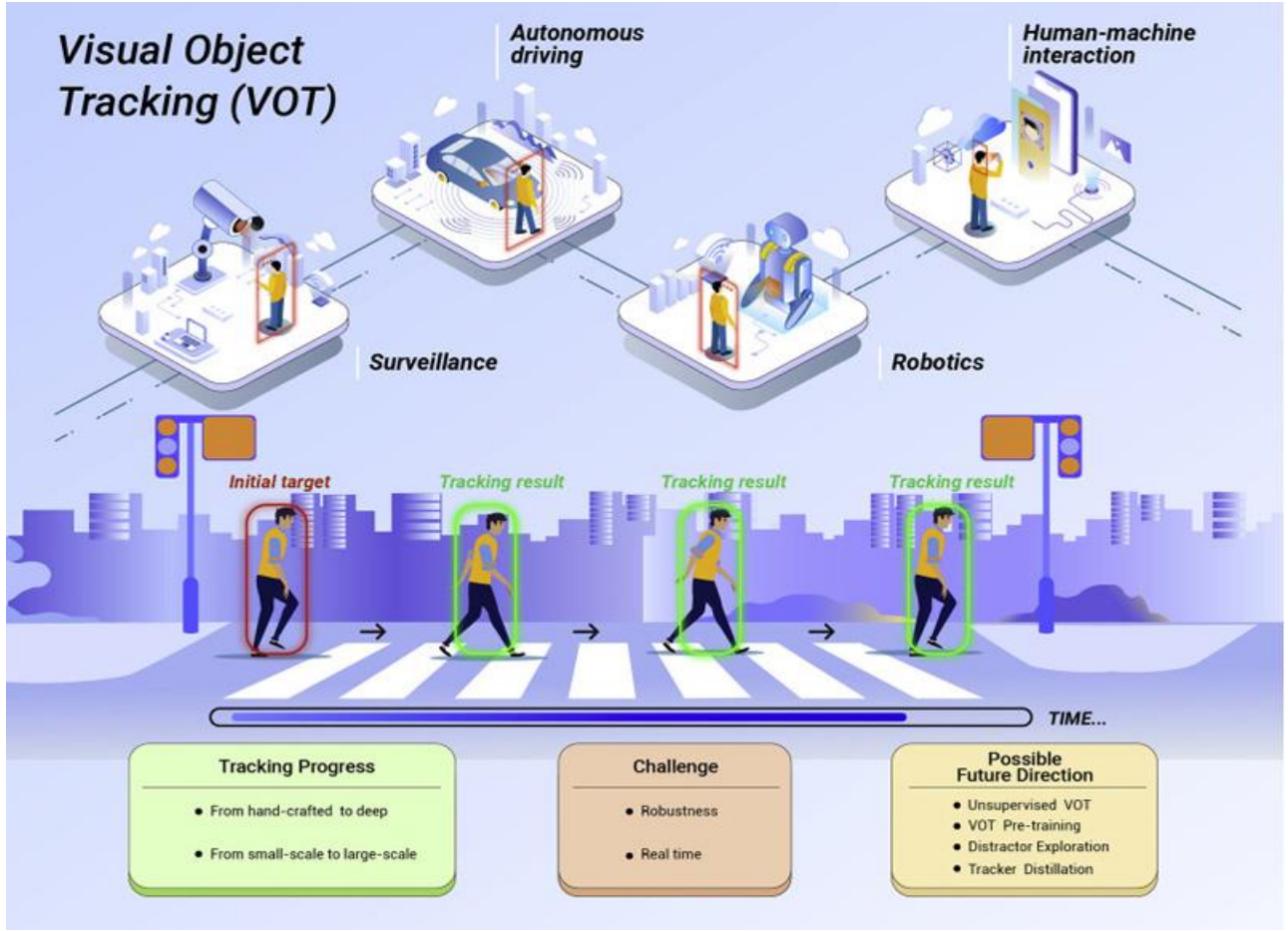
Introduction

❖ Applications

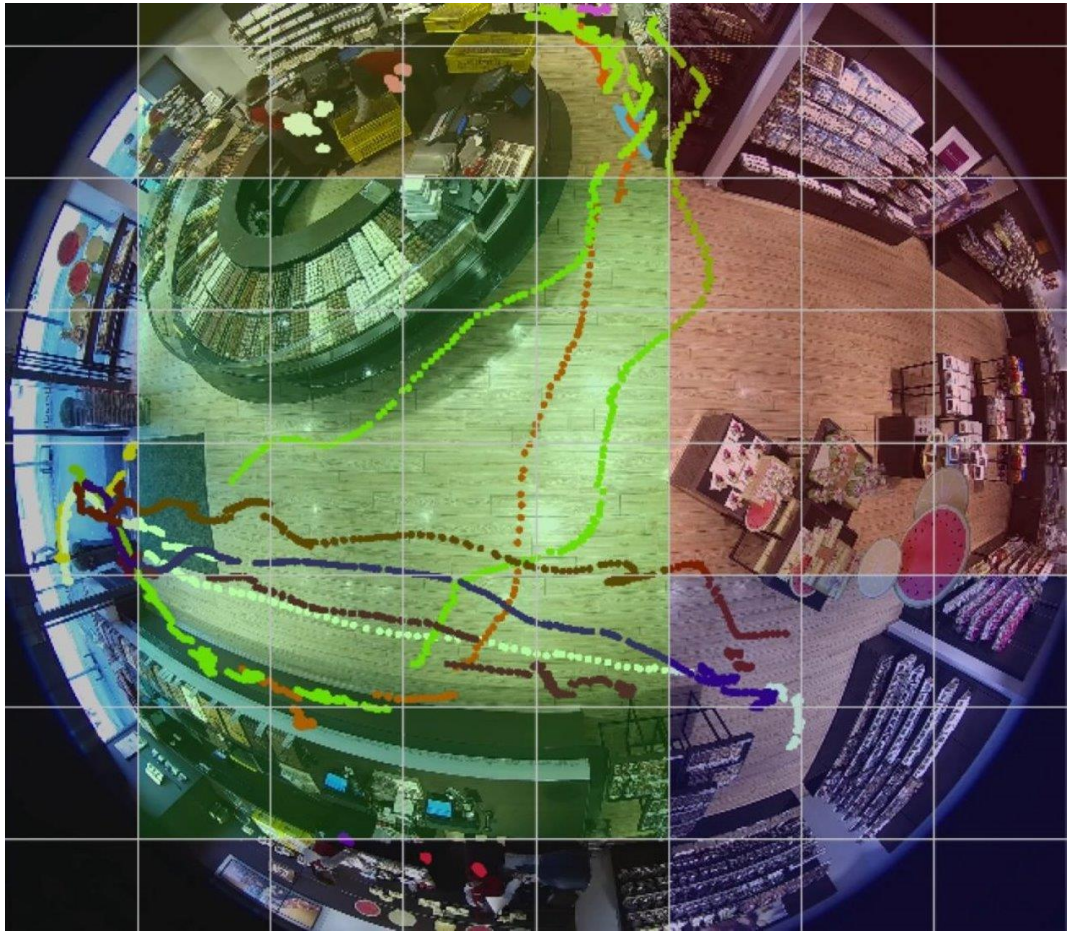
Object Detection and Object Tracking Explained [Real Examples]

Introduction

❖ Applications



In-store people tracking with computer vision



https://www.the-innovation.org/article/id/648ae2a1cf340000a1002a1b
https://viso.ai/deep-learning/object-tracking/

Introduction

❖ Applications

Object Counting

https://www.the-innovation.org/article/id/648ae2a1cf340000a1002a1b

Introduction

❖ Project Description

Description: Given MOT17 dataset about multiple object tracking (download here), build a pedestrian tracking program using YOLOv8 and DeepSORT.

MOT17

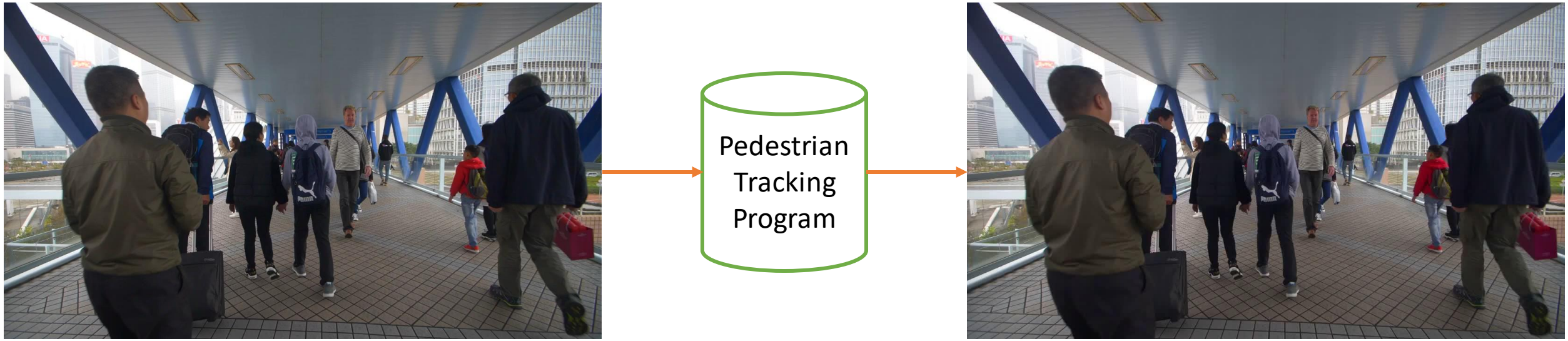
MOT17 Challenge. All MOT16 sequences are used with a new, more accurate ground truth. Each sequences is provided with 3 sets of detections: DPM, Faster-RCNN, and SDP.

Training Set

Sample	Name	FPS	Resolution	Length	Tracks	Boxes	Density	Description	Source	Ref.
	MOT17-13-SDP	25	1920x1080	750 (00:30)	110	11642	15.5	Filmed from a bus on a busy intersection	link	[1]
	MOT17-11-SDP	30	1920x1080	900 (00:30)	75	9436	10.5	Forward moving camera in a busy shopping mall	link	[1]
	MOT17-10-SDP	30	1920x1080	654 (00:22)	57	12839	19.6	A pedestrian scene filmed at night by a moving camera	link	[2]
	MOT17-09-SDP	30	1920x1080	525 (00:18)	26	5325	10.1	A pedestrian street scene filmed from a low angle.	link	[2]
	MOT17-05-SDP	14	640x480	837 (01:00)	133	6917	8.3	Street scene from a moving platform	link	[3]

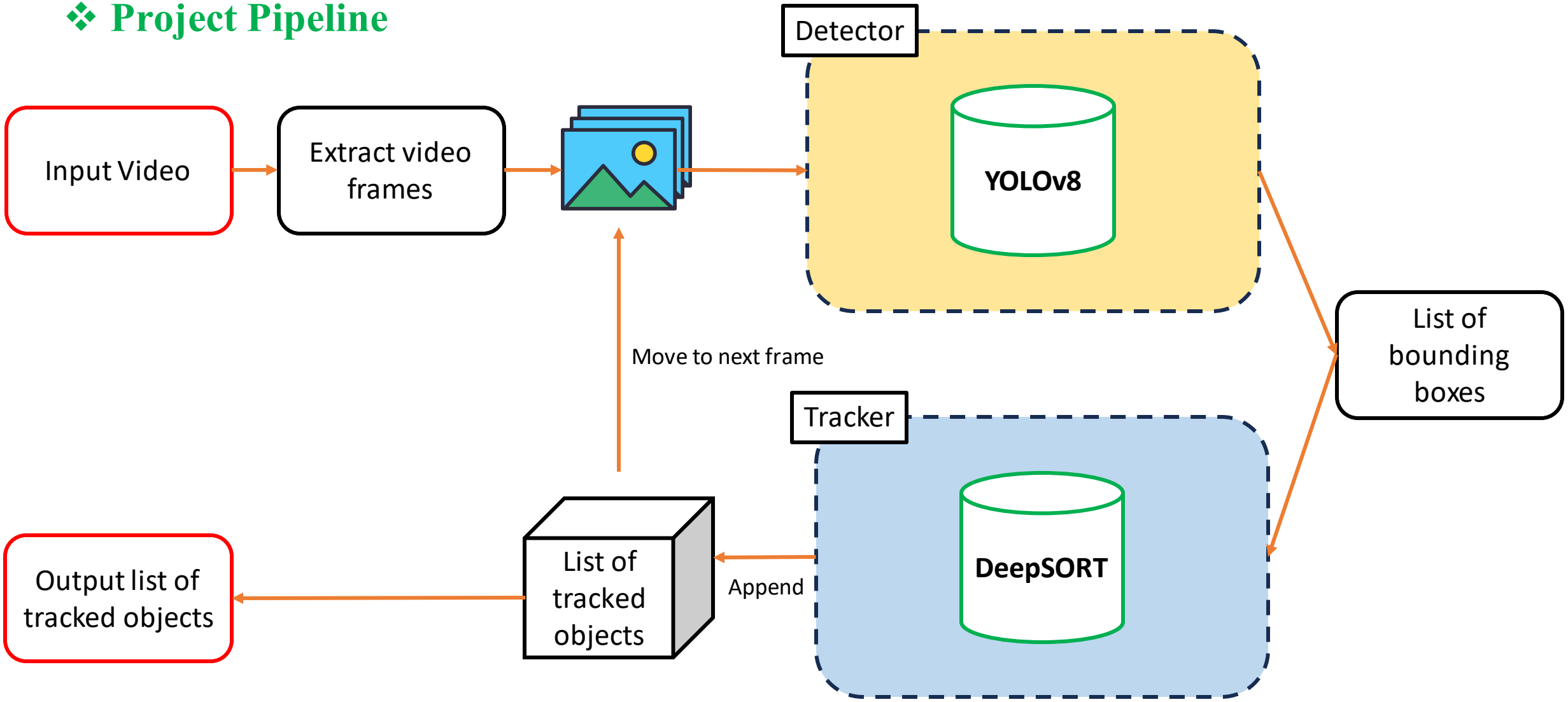
Introduction

❖ Project Description



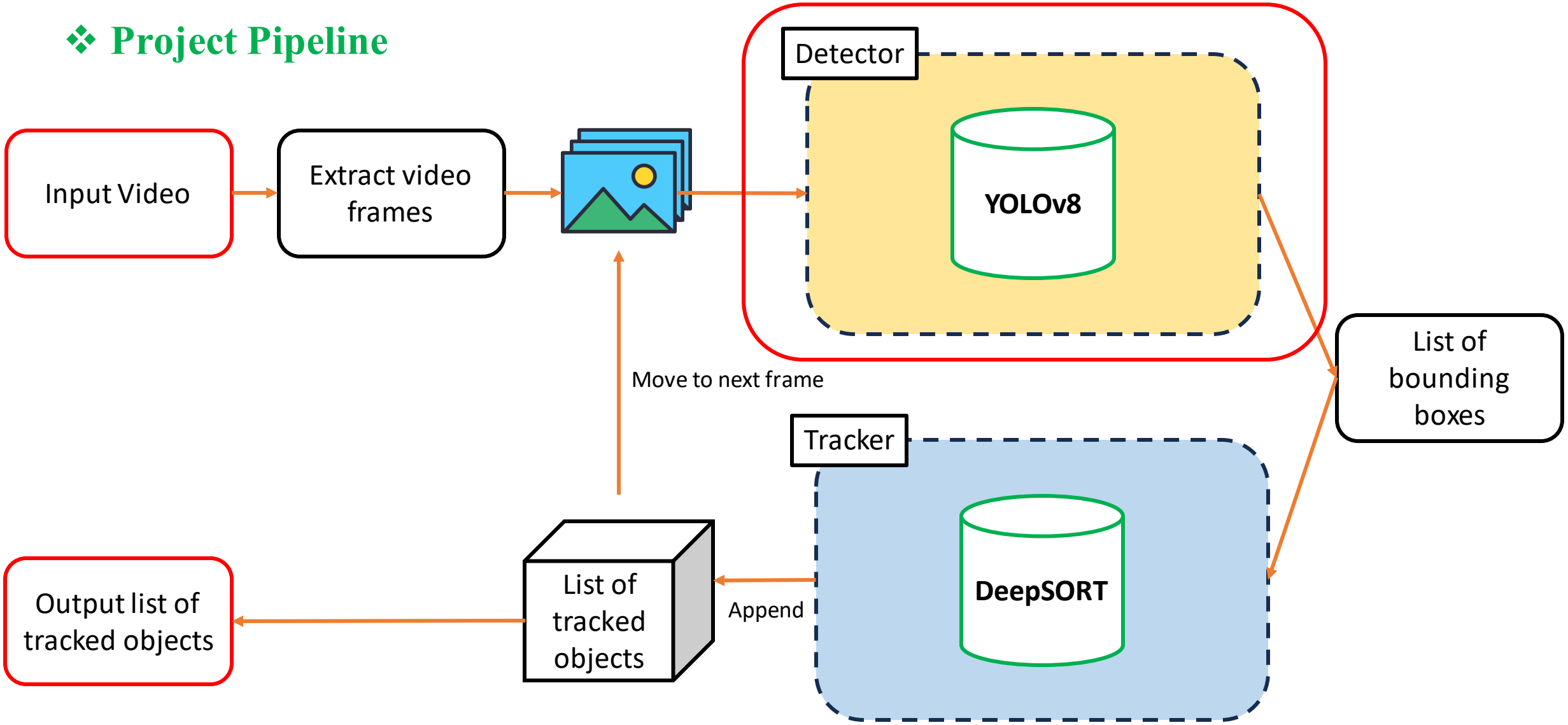
Introduction

❖ Project Pipeline



Detector

❖ Project Pipeline



Detector

❖ Coding Step 1: Download MOT17 dataset

```
# way 1
!wget https://motchallenge.net/data/MOT17.zip

# way 2: might not be available
# !gdown 1vOj9OpxeyozWzpPCtUY7fDVaBQwsPM9n
```

```
Downloading...
From (uriginal): https://drive.google.com/uc?id=1vOj9OpxeyozWzpPCtUY7fDVaBQwsPM9n
From (redirected): https://drive.google.com/uc?id=1vOj9OpxeyozWzpPCtUY7fDVaBQwsPM9n&confirm=t&uuid=f6c0598b-accf-473e-95b2-8be35b5c7932
To: /home/aivn12s1/thangdd/project_object_tracking/MOT17.zip
100%|████████████████████████████████████████| 5.86G/5.86G [09:50<00:00, 9.92MB/s]
```

```
!unzip -qq MOT17.zip
```

Detector

❖ Coding Step 2: Install and import necessary libraries

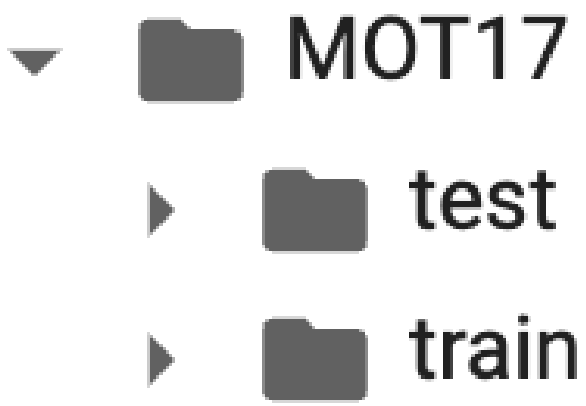
```
!pip install ultralytics -q
```

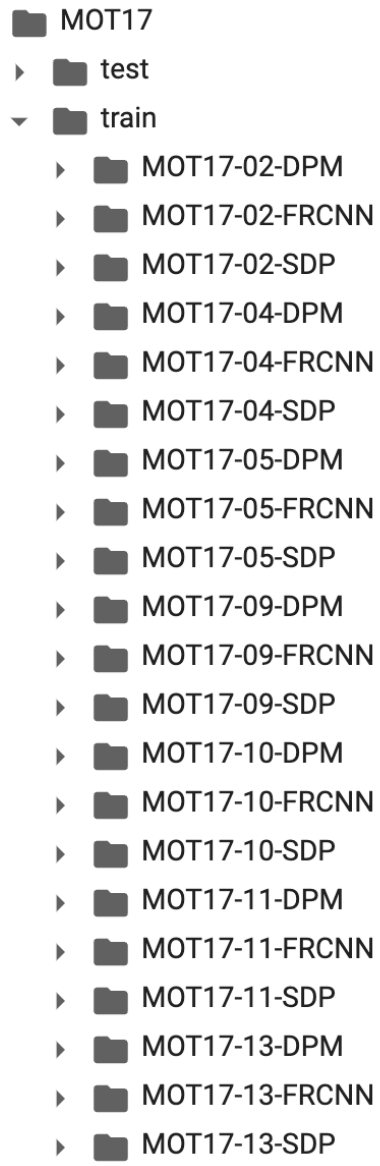
```
import pandas as pd
import os
import yaml
import shutil
import configparser
import ultralytics
ultralytics.checks()

from tqdm import tqdm
from ultralytics import YOLO
```

```
Ultralytics YOLOv8.0.222 🚀 Python-3.11.5 torch-2.1.1 CUDA:0 (NVIDIA GeForce RTX 3060, 12036MiB)
Setup complete ✅ (20 CPUs, 31.1 GB RAM, 470.7/915.3 GB disk)
```

Detector

❖ Coding Step 3: MOT17 Dataset





Detector

❖ Coding Step 3: MOT17 Dataset

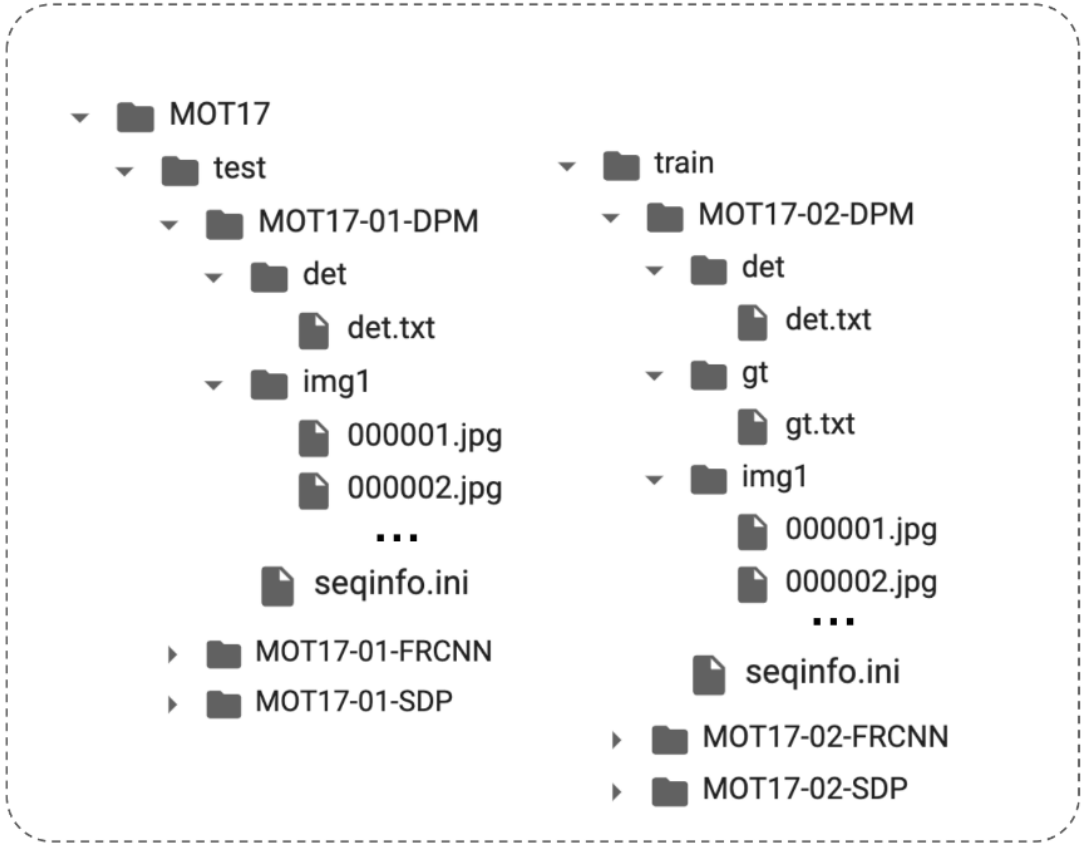
seqinfo.ini

```
1 [Sequence]
2 name=MOT17-02-DPM
3 imDir=img1
4 frameRate=30
5 seqLength=600
6 imWidth=1920
7 imHeight=1080
8 imExt=.jpg
9
10
```

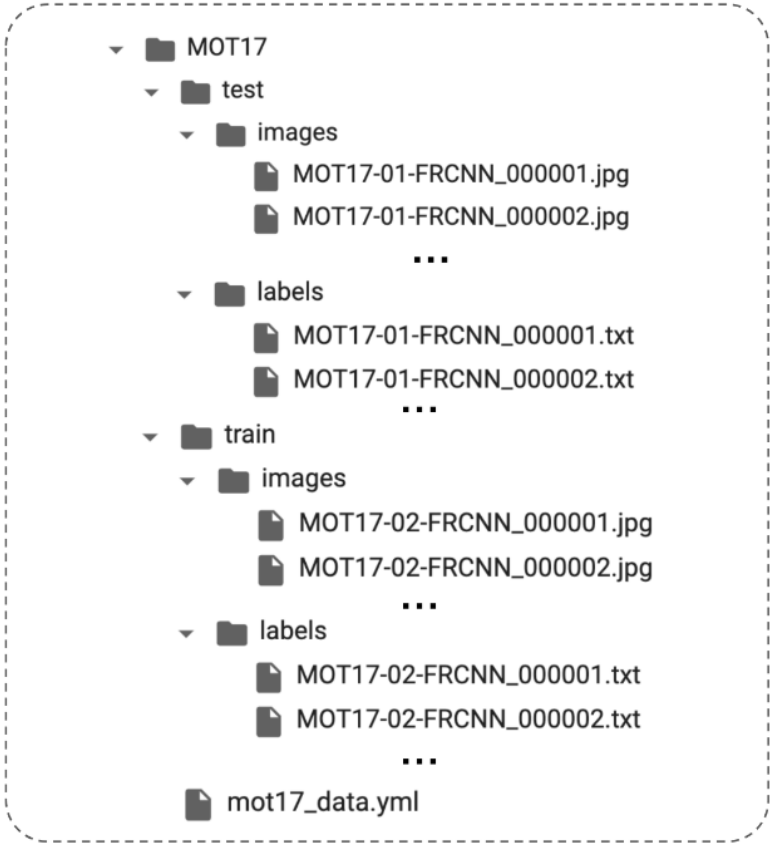
Detector

❖ Coding Step 3: MOT17 Dataset

Before Processing

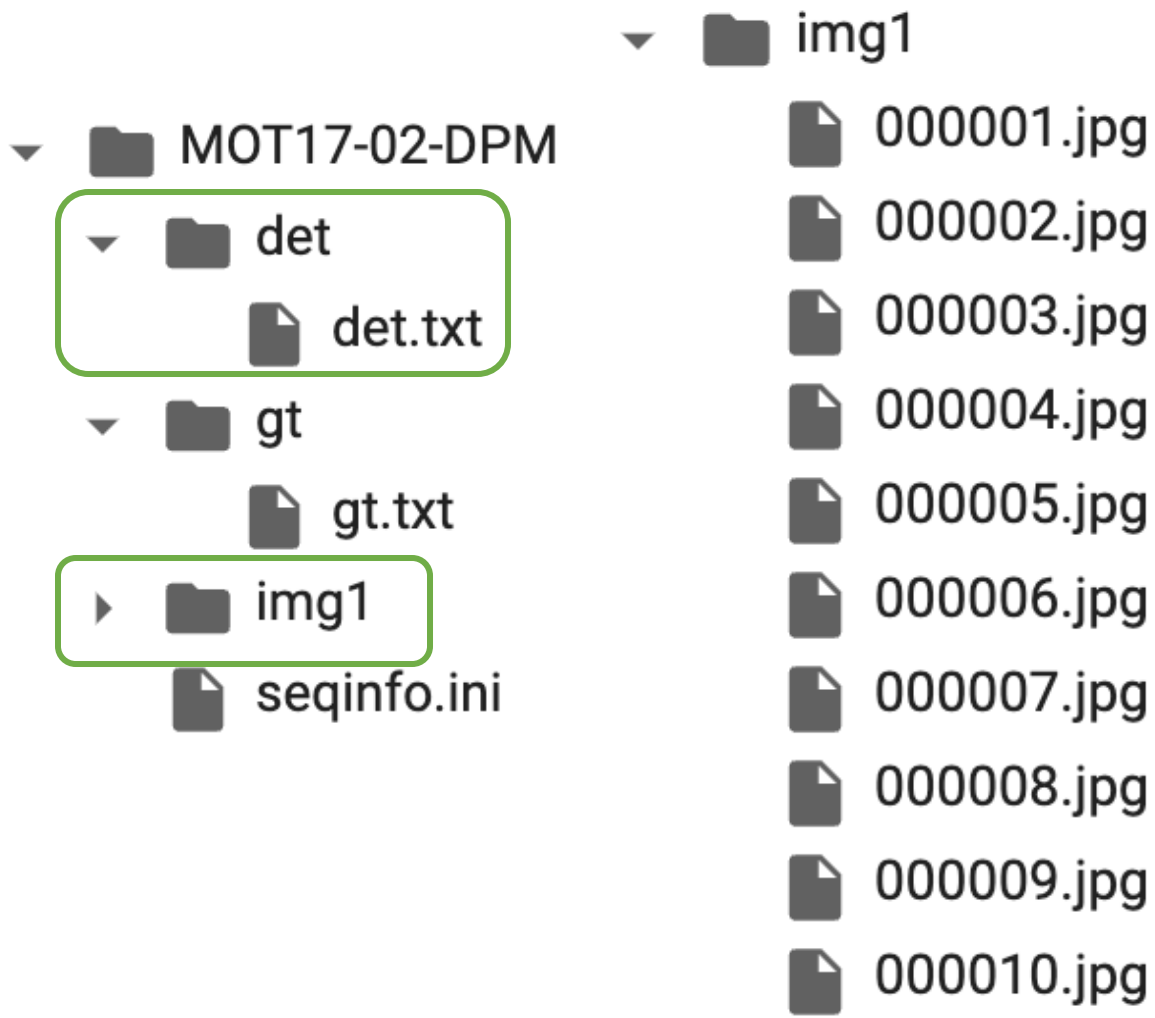


After Processing



Detector

❖ Coding Step 3: MOT17 Dataset



det.txt

```
1 1,-1,1359.1,413.27,120.26,362.77,2.3092,-1,-1,-1
2 1,-1,571.03,402.13,104.56,315.68,1.5028,-1,-1,-1
3 1,-1,650.8,455.86,63.98,193.94,0.33276,-1,-1,-1
4 1,-1,721.23,446.86,41.871,127.61,0.27401,-1,-1,-1
5 1,-1,454.06,434.36,97.492,294.47,0.20818,-1,-1,-1
6 1,-1,1254.6,446.72,33.822,103.47,0.14776,-1,-1,-1
7 1,-1,1301.1,237.38,195.98,589.95,0.051818,-1,-1,-1
8 1,-1,1480.3,413.27,120.26,362.77,-0.020474,-1,-1,-1
9 1,-1,552.72,473.9,29.314,89.943,-0.087553,-1,-1,-1
10 1,-1,1097,433,39,119,-0.17964,-1,-1,-1
```

Create a labels folder based on det.txt

Detector

❖ Coding Step 4: Define convert to YOLO bounding box function

The file format should be the same as the ground truth file, which is a CSV text-file containing one object instance per line. Each line must contain 10 values:

```
<frame>, <id>, <bb_left>, <bb_top>, <bb_width>, <bb_height>, <conf>, <x>, <y>, <z>
```

The `conf` value contains the detection confidence in the `det.txt` files. For the ground truth, it acts as a flag whether the entry is to be considered. A value of 0 means that this particular instance is ignored in the evaluation, while any other value can be used to mark it as active. For submitted results, *all* lines in the .txt file are considered. The world coordinates `x,y,z` are ignored for the 2D challenge and can be filled with -1. Similarly, the bounding boxes are ignored for the 3D challenge. However, each line is still required to contain 10 values.

All frame numbers, target IDs and bounding boxes are 1-based. Here is an example:

Detector

❖ Coding Step 4: Define convert to YOLO bounding box function

det.txt

```
1 1,-1,1359.1,413.27,120.26,362.77,2.3092,-1,-1,-1
2 1,-1,571.03,402.13,104.56,315.68,1.5028,-1,-1,-1
3 1,-1,650.8,455.86,63.98,193.94,0.33276,-1,-1,-1
4 1,-1,721.23,446.86,41.871,127.61,0.27401,-1,-1,-1
5 1,-1,454.06,434.36,97.492,294.47,0.20818,-1,-1,-1
6 1,-1,1254.6,446.72,33.822,103.47,0.14776,-1,-1,-1
7 1,-1,1301.1,237.38,195.98,589.95,0.051818,-1,-1,-1
8 1,-1,1480.3,413.27,120.26,362.77,-0.020474,-1,-1,-1
9 1,-1,552.72,473.9,29.314,89.943,-0.087553,-1,-1,-1
10 1,-1,1097,433,39,119,-0.17964,-1,-1,-1
```

The file format should be the same as the ground truth file, which is a CSV text-file containing one object instance per line. Each line must contain 10 values:

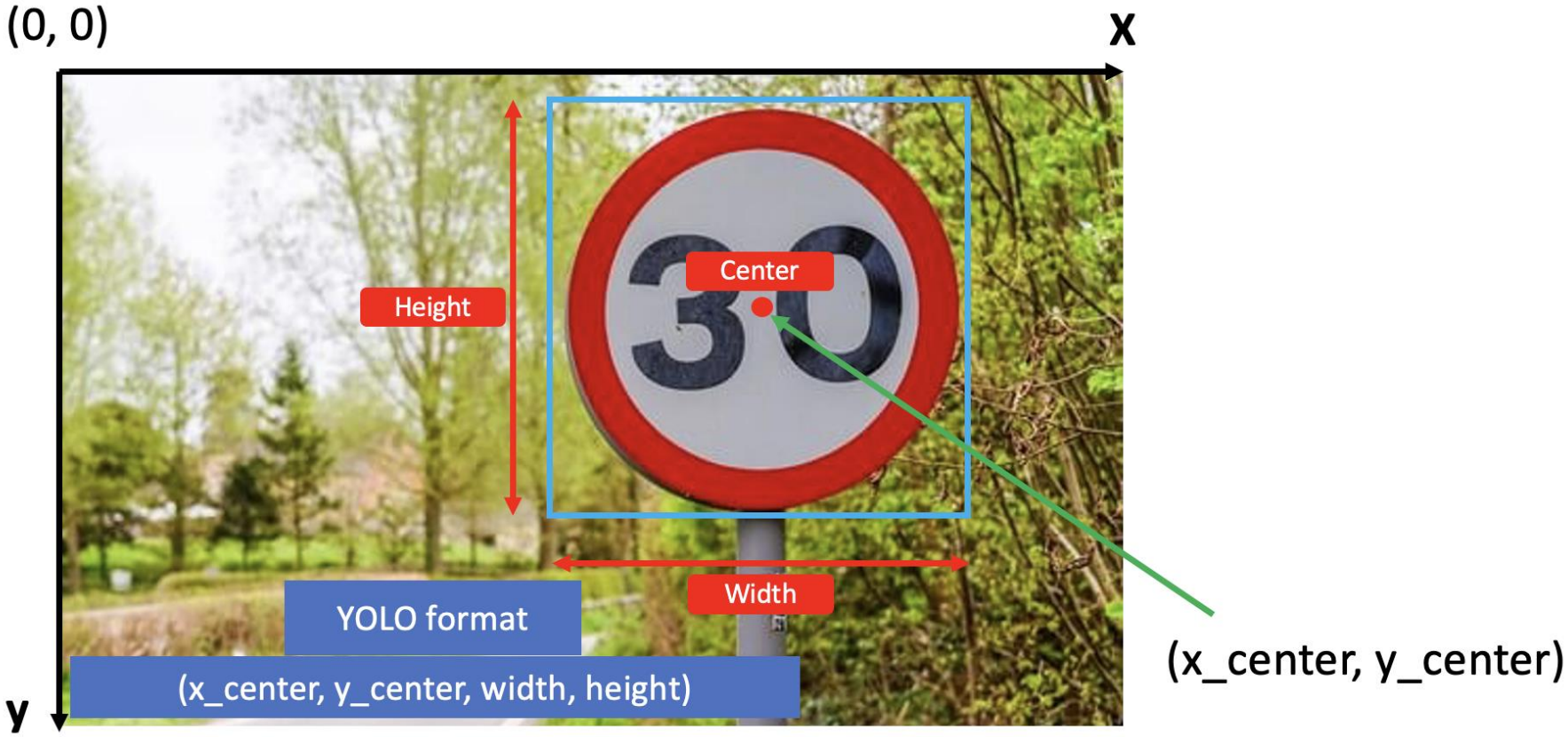
`<frame>, <id>, <bb_left>, <bb_top>, <bb_width>, <bb_height>, <conf>, <x>, <y>, <z>`

The `conf` value contains the detection confidence in the `det.txt` files. For the ground truth, it acts as a flag whether the entry is to be considered. A value of 0 means that this particular instance is ignored in the evaluation, while any other value can be used to mark it as active. For submitted results, *all* lines in the .txt file are considered. The world coordinates `x,y,z` are ignored for the 2D challenge and can be filled with -1. Similarly, the bounding boxes are ignored for the 3D challenge. However, each line is still required to contain 10 values.

All frame numbers, target IDs and bounding boxes are 1-based. Here is an example:

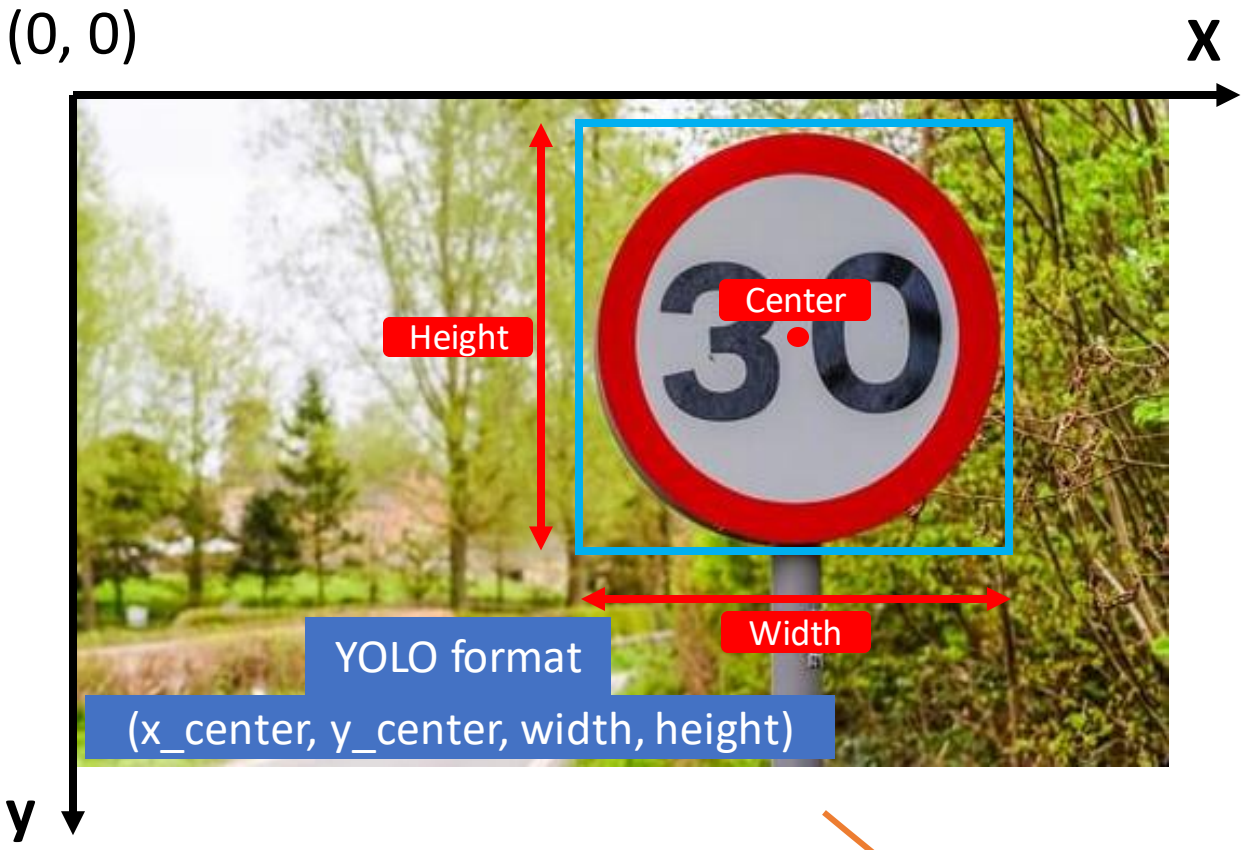
Detector

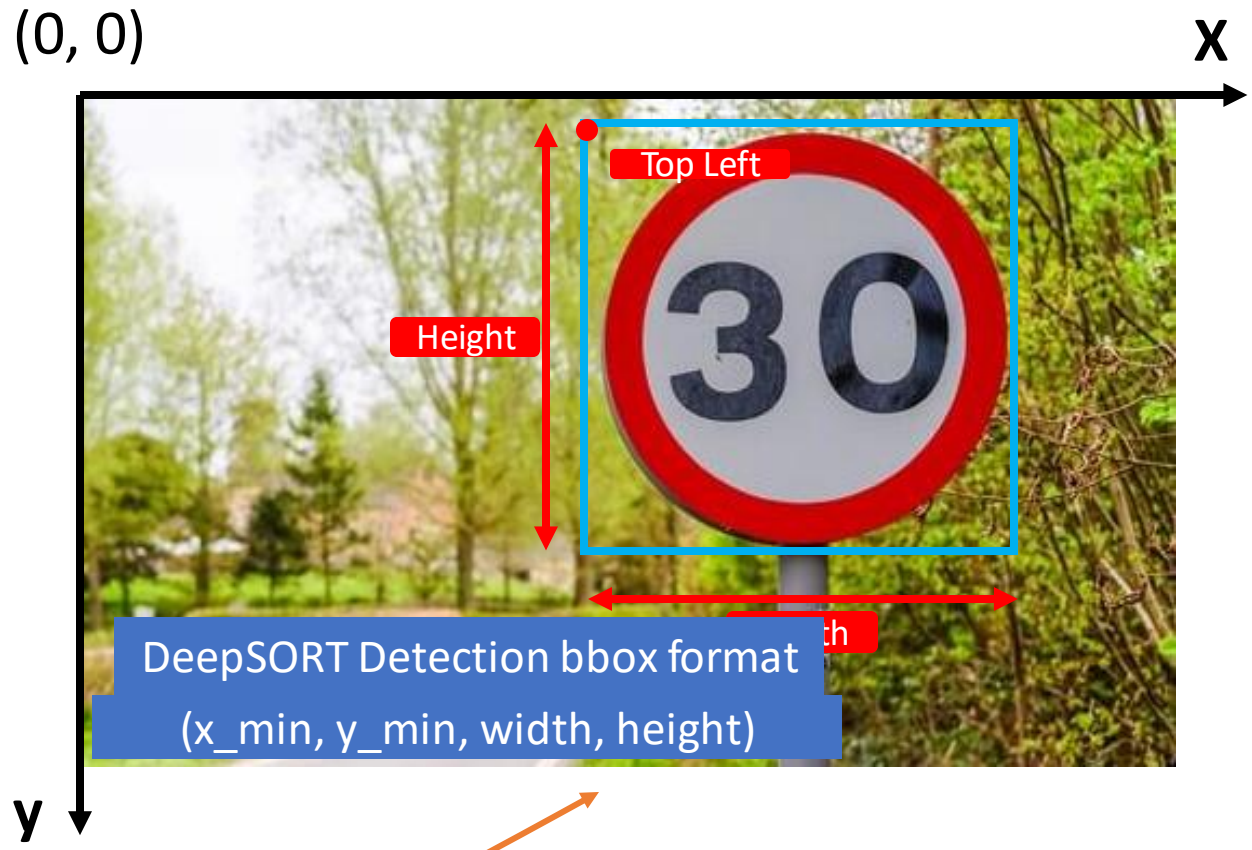
❖ Coding Step 4: Define convert to YOLO bounding box function



Detector

❖ Coding Step : Define YOLOv8 class

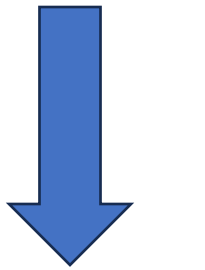




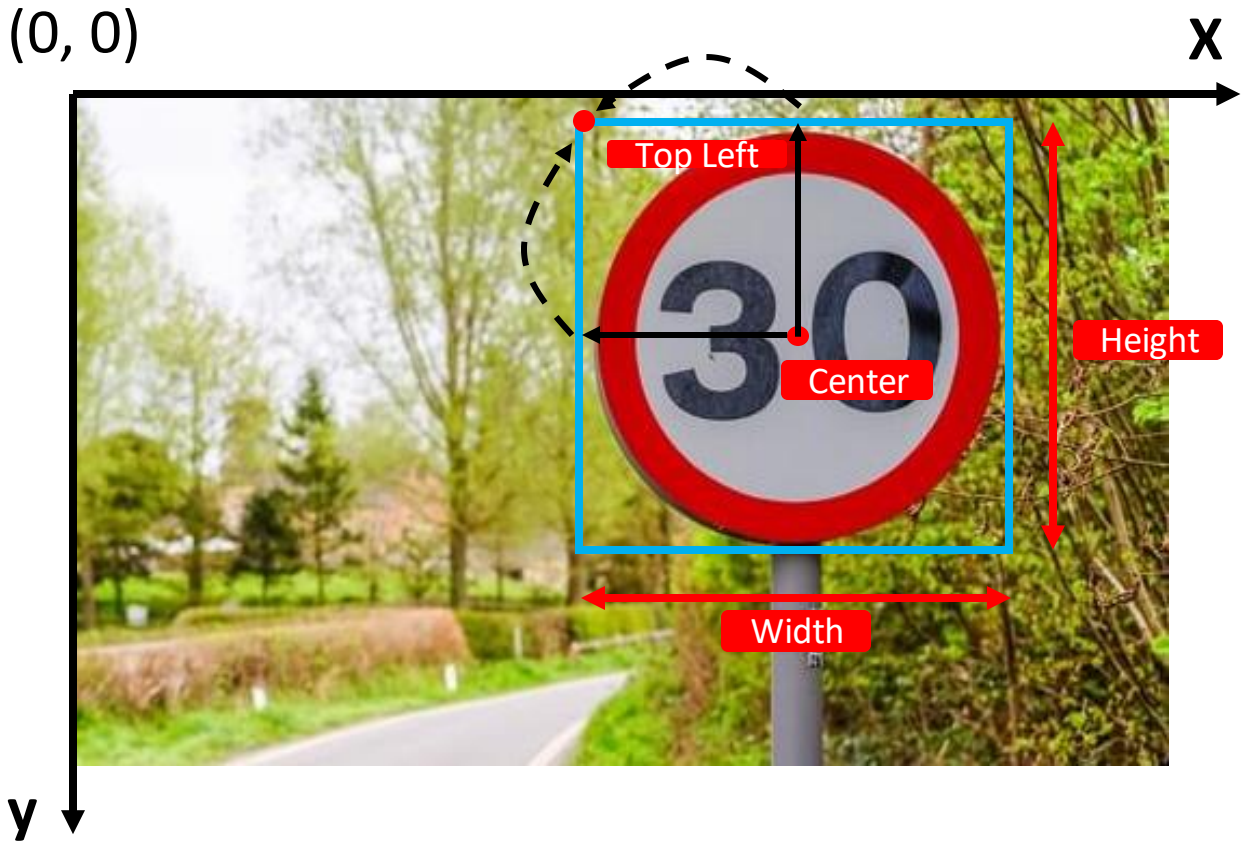
Detector

❖ Coding Step : Define YOLOv8 class

[x_center, y_center, width, height]



[x_min, y_min, width, height]



$$\begin{cases} x_{min} = x_{center} - \dfrac{width}{2} \\ y_{min} = y_{center} - \dfrac{height}{2} \end{cases}$$

Detector

❖ Coding Step 4: Define convert to YOLO bounding box function

```python
def convert_to_yolo_format(bb, img_width, img_height):
    x_center = bb['bb_left'] + (bb['bb_width'] / 2)
    y_center = bb['bb_top'] + (bb['bb_height'] / 2)

    # Normalize the coordinates by the dimensions of the image
    x_center /= img_width
    y_center /= img_height
    bb_width_normalized = bb['bb_width'] / img_width
    bb_height_normalized = bb['bb_height'] / img_height

    # Clip the values to make sure they are between 0 and 1
    x_center = max(min(x_center, 1), 0)
    y_center = max(min(y_center, 1), 0)
    bb_width_normalized = max(min(bb_width_normalized, 1), 0)
    bb_height_normalized = max(min(bb_height_normalized, 1), 0)

    return (x_center, y_center, bb_width_normalized, bb_height_normalized)
```

Detector

❖ Coding Step 5: Create labels folders for a subfolder data

```python
def process_folder(folder_path):
    # Read image dimensions from seqinfo.ini
    config = configparser.ConfigParser()
    config.read(os.path.join(folder_path, 'seqinfo.ini'))
    img_width = int(config['Sequence']['imWidth'])
    img_height = int(config['Sequence']['imHeight'])

    # Load ground truth data
    gt_path = os.path.join(folder_path, 'det/det.txt')
    gt_data = pd.read_csv(
        gt_path,
        header=None,
        names=['frame', 'id', 'bb_left', 'bb_top', 'bb_width', 'bb_height', 'conf', 'class', 'visibility']
    )

    labels_folder = os.path.join(folder_path, 'labels')
    os.makedirs(labels_folder, exist_ok=True)

    for frame_number in gt_data['frame'].unique():
        frame_data = gt_data[gt_data['frame'] == frame_number]
        label_file = os.path.join(labels_folder, f'{frame_number:06d}.txt')

        with open(label_file, 'w') as file:
            for _, row in frame_data.iterrows():
                yolo_bb = convert_to_yolo_format(row, img_width, img_height)
                file.write(f'0 {yolo_bb[0]} {yolo_bb[1]} {yolo_bb[2]} {yolo_bb[3]}\n')
```

Detector

❖ Coding Step 5: Create labels folders for a subfolder data

	MOT17-13-FRCNN	25	1920x1080	750 (00:30)	110	11642	15.5
	MOT17-11-FRCNN	30	1920x1080	900 (00:30)	75	9436	10.5
	MOT17-10-FRCNN	30	1920x1080	654 (00:22)	57	12839	19.6
	MOT17-09-FRCNN	30	1920x1080	525 (00:18)	26	5325	10.1
	MOT17-05-FRCNN	14	640x480	837 (01:00)	133	6917	8.3
	MOT17-04-FRCNN	30	1920x1080	1050 (00:35)	83	47557	45.3
	MOT17-02-FRCNN	30	1920x1080	600 (00:20)	62	18581	31.0

```python
def process_all_folders(base_directory):
    # List all subdirectories in the base directory
    for folder_name in tqdm(os.listdir(base_directory)):
        folder_path = os.path.join(base_directory, folder_name)

        # Delete folder not contain 'FRCNN' in name
        if 'FRCNN' not in folder_name:
            os.system(f'rm -rf {folder_path}')
            continue

        if os.path.isdir(folder_path):
            process_folder(folder_path)
```

We only use a subset of the dataset for demo purpose

Detector

❖ Coding Step 5: Create labels folders for a subfolder data

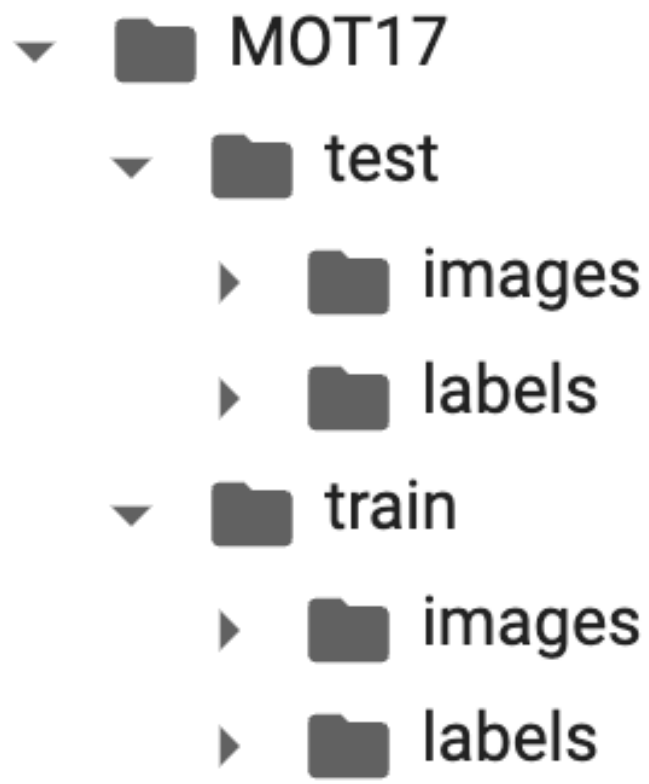
```python
def process_all_folders(base_directory):
    # List all subdirectories in the base directory
    for folder_name in tqdm(os.listdir(base_directory)):
        folder_path = os.path.join(base_directory, folder_name)

        # Delete folder not contain 'FRCNN' in name
        if 'FRCNN' not in folder_name:
            os.system(f'rm -rf {folder_path}')
            continue

        if os.path.isdir(folder_path):
            process_folder(folder_path)
```

We only use a subset of the dataset for demo purpose

Detector

❖ Coding Step 6: Move all images and labels

We need to merge all images and labels folders into two folders



Detector

❖ Coding Step 6: Move all images and labels

```python
def rename_and_move_files(src_folder, dst_folder, folder_name, file_extension):
    for filename in os.listdir(src_folder):
        if filename.endswith(file_extension):
            # Include folder name in the new filename
            new_filename = f'{folder_name}_{filename}'
            shutil.move(
                os.path.join(src_folder, filename),
                os.path.join(dst_folder, new_filename)
            )
```

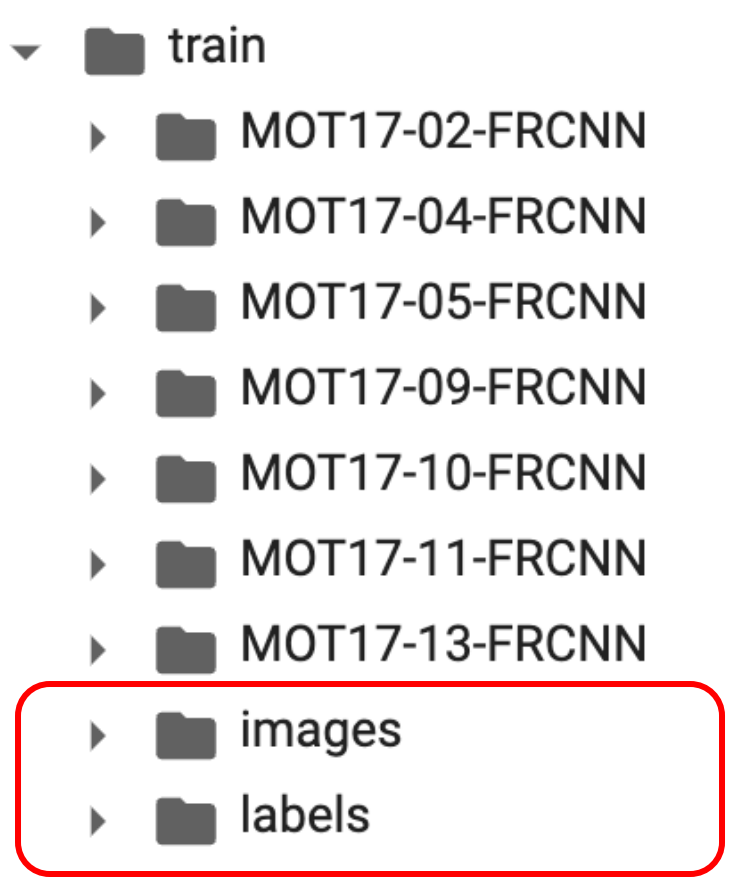
Detector

❖ Coding Step 6: Move all images and labels

```python
def move_files_all_folders(base_directory):
    images_dir = os.path.join(base_directory, 'images')
    labels_dir = os.path.join(base_directory, 'labels')
    os.makedirs(images_dir, exist_ok=True)
    os.makedirs(labels_dir, exist_ok=True)

    for folder_name in tqdm(os.listdir(base_directory)):
        if folder_name in ['images', 'labels']:  # Skip these folders
            continue

        folder_path = os.path.join(base_directory, folder_name)
        if os.path.isdir(folder_path):
            rename_and_move_files(os.path.join(folder_path, 'img1'), images_dir, folder_name, '.jpg')
            rename_and_move_files(os.path.join(folder_path, 'labels'), labels_dir, folder_name, '.txt')
```

Detector

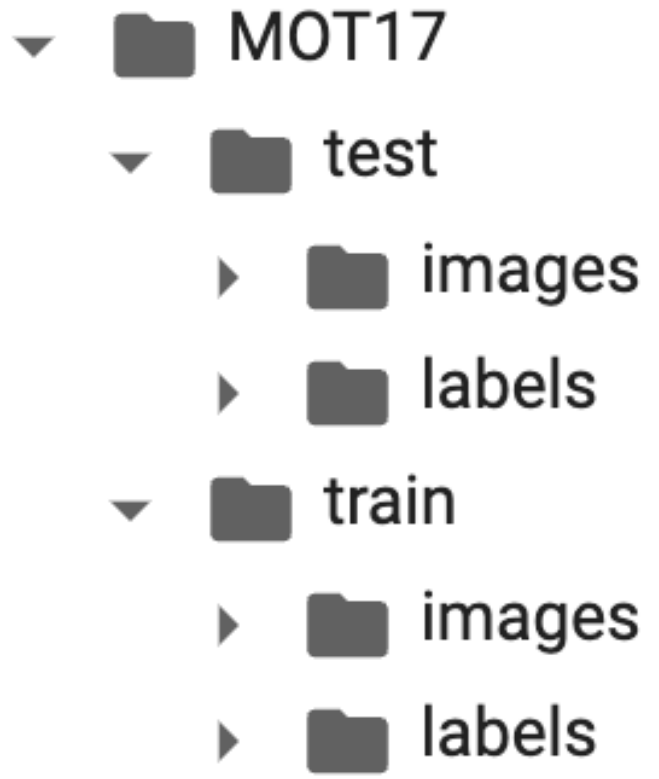
❖ Coding Step 6: Move all images and labels



Detector

❖ Coding Step 7: Remove unnecessary files/folders

Delete all unnecessary folders to completely becomes YOLO structure format



Detector

❖ Coding Step 7: Remove unnecessary files/folders

```python
def delete_subfolders(base_directory):
    for folder_name in os.listdir(base_directory):
        folder_path = os.path.join(base_directory, folder_name)
        if os.path.isdir(folder_path) and folder_name not in ['images', 'labels']:
            shutil.rmtree(folder_path)
            print(f"Deleted folder: {folder_name}")
```

```python
# Delete all subfolders except 'images' and 'labels'
delete_subfolders('MOT17/train')
delete_subfolders('MOT17/test')
```

```
Deleted folder: MOT17-05-FRCNN
Deleted folder: MOT17-02-FRCNN
Deleted folder: MOT17-11-FRCNN
Deleted folder: MOT17-09-FRCNN
Deleted folder: MOT17-13-FRCNN
Deleted folder: MOT17-04-FRCNN
Deleted folder: MOT17-10-FRCNN
Deleted folder: MOT17-06-FRCNN
Deleted folder: MOT17-03-FRCNN
Deleted folder: MOT17-12-FRCNN
Deleted folder: MOT17-08-FRCNN
Deleted folder: MOT17-14-FRCNN
Deleted folder: MOT17-07-FRCNN
Deleted folder: MOT17-01-FRCNN
```

Detector

❖ Coding Step 8: Create file .yaml

```python
class_labels = [
    'objects'
]
dataset_root_dir = os.path.join(
    os.getcwd(),
    'MOT17'
)
yolo_yaml_path = os.path.join(
    dataset_root_dir,
    'mot17_data.yml'
)

data_yaml = {
    'path': dataset_root_dir,
    'train': 'train/images',
    'val': 'test/images',
    'nc': len(class_labels),
    'names': class_labels
}

with open(yolo_yaml_path, 'w') as f:
    yaml.dump(data_yaml, f, default_flow_style=False)
```

```yaml
names:
- objects
nc: 1
path: /home/aivn12s1/thangdd/project_object_tracking/MOT17
train: train/images
val: test/images

```

Detector

❖ Coding Step 9: Training

```python
from ultralytics import YOLO

# Load the YOLOv8 model
model = YOLO('yolov8s.pt')

# Config
epochs = 30
batch_size = -1 # Auto scale based on GPU memory
img_size = 640
project_name = 'models/yolo'
name = 'yolov8s_mot17_det'

# Train the model
results = model.train(
    data=yolo_yaml_path,
    epochs=epochs,
    batch=batch_size,
    imgsz=img_size,
    project=project_name,
    name=name
)
```

```
Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 1/30      8.12G      1.095     0.7265     0.9617        104        640: 100%|██████████|
              Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                all       5919     110141       0.85      0.782      0.882      0.553

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 2/30      8.14G      0.998     0.5988     0.9331         56        640: 100%|██████████|
              Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                all       5919     110141      0.893      0.815      0.907      0.607

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 3/30      7.81G     0.9428     0.5573      0.917         85        640: 100%|██████████|
              Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                all       5919     110141      0.869      0.815      0.894      0.571

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 4/30      7.89G       0.93     0.5494     0.9146         71        640: 100%|██████████|
              Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                all       5919     110141      0.879      0.798      0.882      0.577

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 5/30      7.91G     0.8723      0.514     0.9004         87        640: 100%|██████████|
              Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
```

Detector

❖ Coding Step 10: Evaluation

```python
from ultralytics import YOLO

# Load the trained model
model_path = os.path.join(
    project_name, name, 'weights/best.pt'
)
model = YOLO(model_path)

metrics = model.val(
    project=project_name,
    name='detect/val'
)
```

```
Ultralytics YOLOv8.0.222 🚀 Python-3.11.5 torch-2.1.1 CUDA:0 (NVIDIA GeForce RTX 3060, 12036MiB)
Model summary (fused): 168 layers, 11125971 parameters, 0 gradients, 28.4 GFLOPs
val: Scanning /home/aivn12s1/thangdd/project_object_tracking/MOT17/test/labels.cache... 5908 images, 11 backgrounds, 0 corrupt: 100%|██████████| 5919/5919 [00:
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 370/370 [00:33<00:00, 10.95it/s]
                   all       5919     110141      0.903      0.809      0.908      0.655
Speed: 0.1ms preprocess, 3.4ms inference, 0.0ms loss, 0.2ms postprocess per image
Results saved to models/yolo/detect/val
```

Detector

❖ Coding Step 11: Inference

Detector

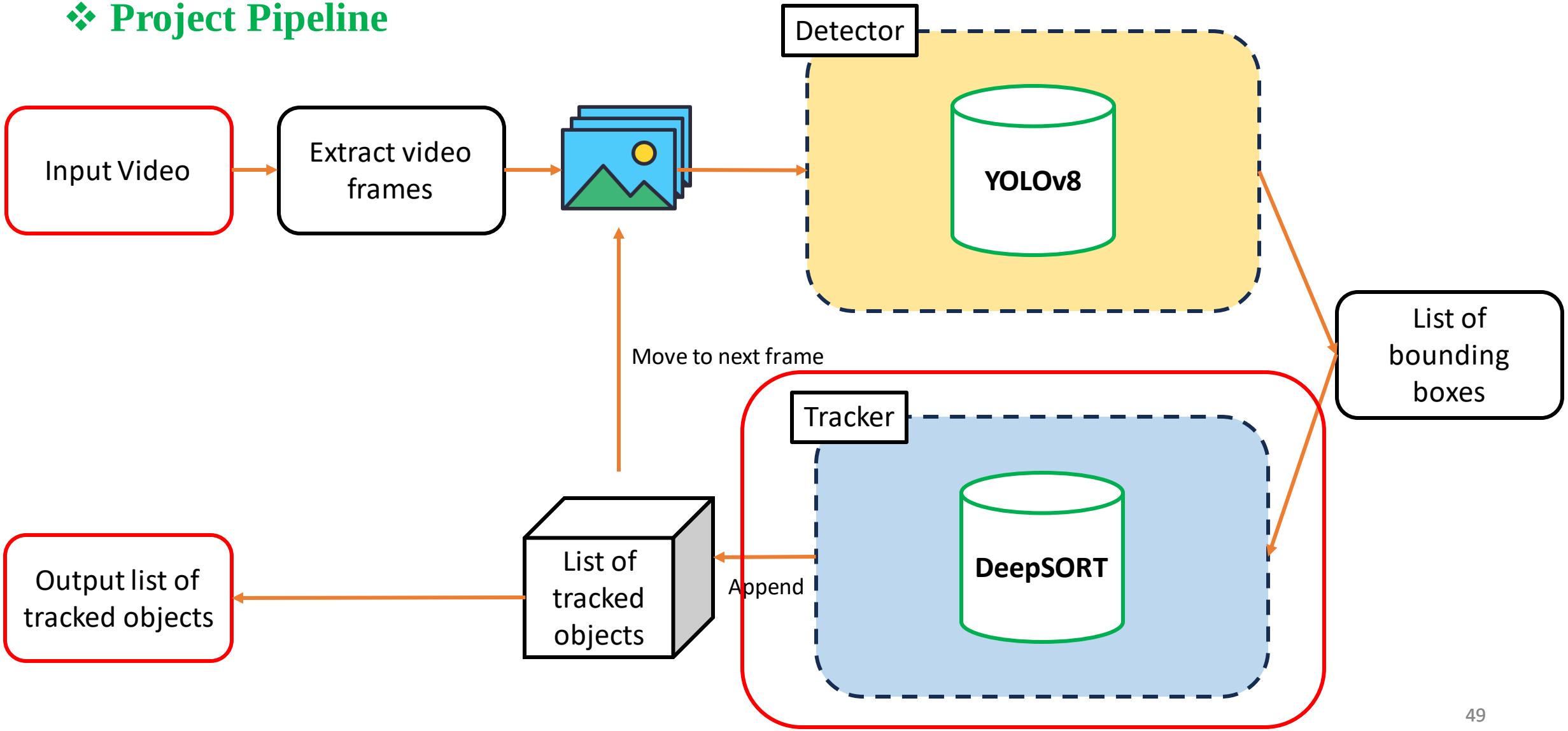
❖ Coding Step 12: Define YOLOv8 class

```python
from ultralytics import YOLO

class YOLOv8:
    def __init__(
        self,
        model_path
    ):
        self.model = YOLO(model_path)

    def detect(self, source_img):
        results = self.model.predict(source_img, verbose=False)[0]
        bboxes = results.boxes.xywh.cpu().numpy()
        bboxes[:, :2] = bboxes[:, :2] - (bboxes[:, 2:] / 2)
        scores = results.boxes.conf.cpu().numpy()
        class_ids = results.boxes.cls.cpu().numpy()

        return bboxes, scores, class_ids
```

Class YOLOv8
- Attributes + model
- Methods + detect

Tracker

❖ Project Pipeline



Tracker

❖ Object Tracking: DeepSORT

SIMPLE ONLINE AND REALTIME TRACKING WITH A DEEP ASSOCIATION METRIC

Nicolai Wojke[†], *Alex Bewley*[◇], *Dietrich Paulus*[†]

University of Koblenz-Landau[†], Queensland University of Technology[◇]

ABSTRACT

Simple Online and Realtime Tracking (SORT) is a pragmatic approach to multiple object tracking with a focus on simple, effective algorithms. In this paper, we integrate appearance information to improve the performance of SORT. Due to this extension we are able to track objects through longer periods of occlusions, effectively reducing the number of identity switches. In spirit of the original framework we place much of the computational complexity into an offline pre-training stage where we learn a deep association metric on a large-scale person re-identification dataset. During online application, we establish measurement-to-track associations using nearest neighbor queries in visual appearance space. Experimental evaluation shows that our extensions reduce the number of identity switches by 45%, achieving overall competitive performance at high frame rates.

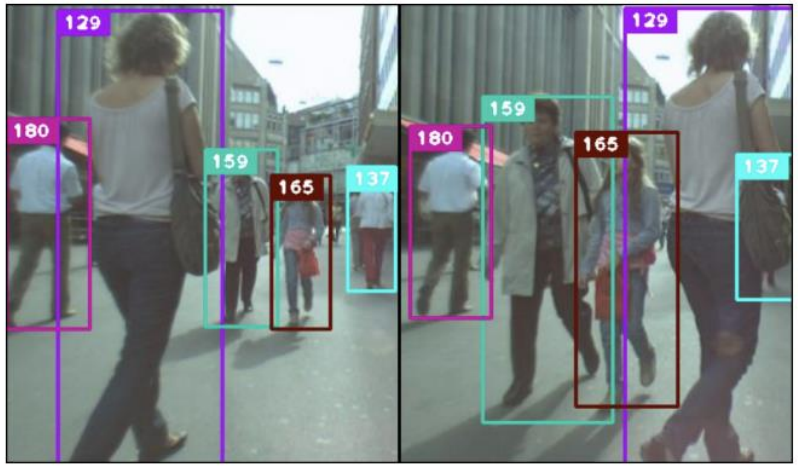
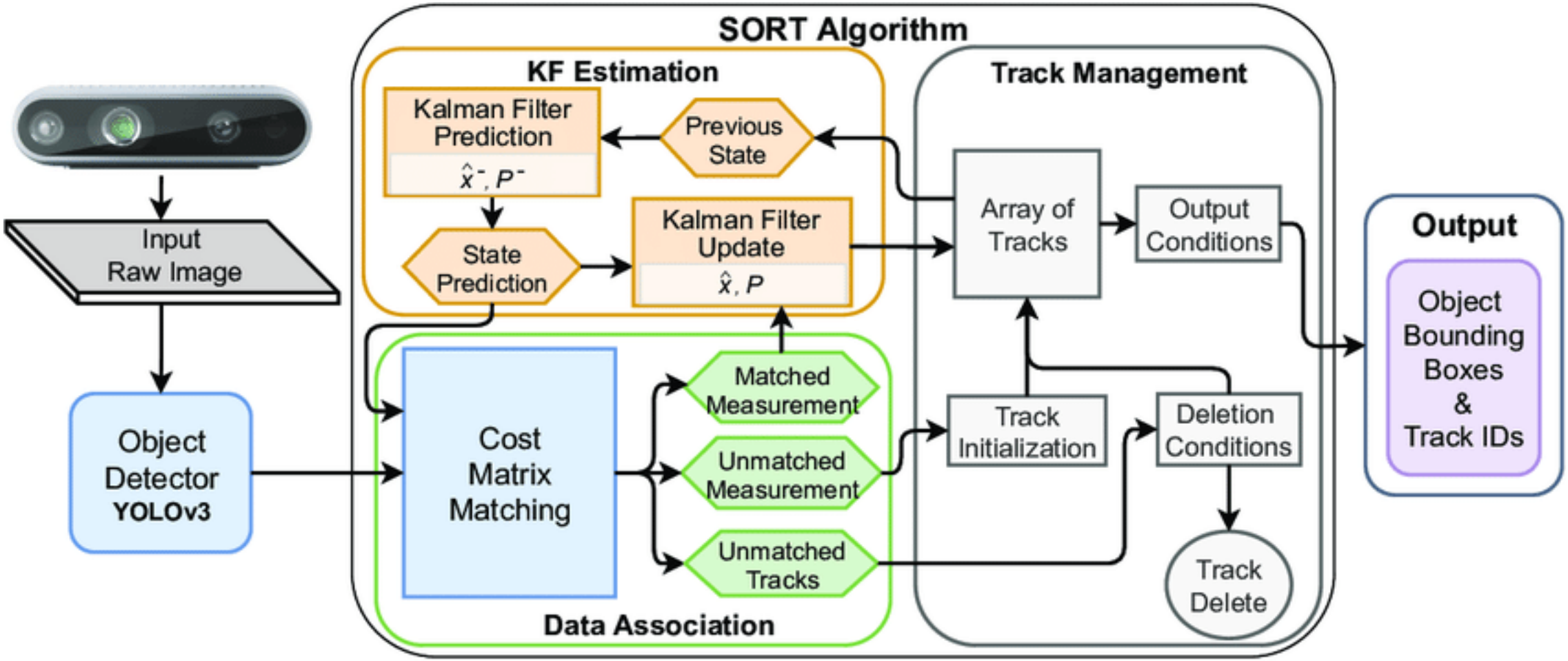


Fig. 1: Exemplary output of our method on the MOT challenge dataset [15] in a common tracking situation with frequent occlusion.

https://arxiv.org/abs/1703.07402

Tracker

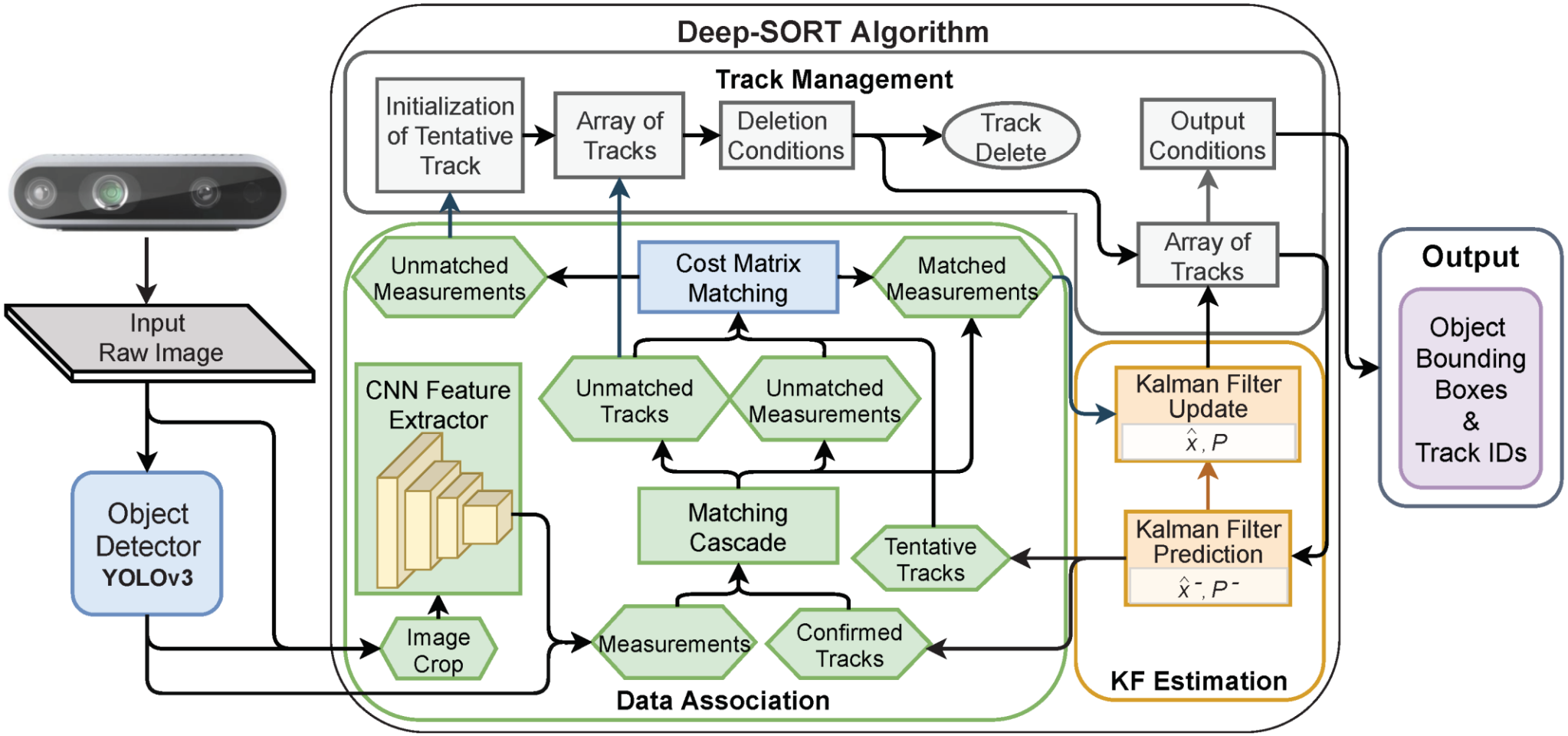
❖ Object Tracking: SORT Pipeline



https://arxiv.org/abs/1602.00763
https://www.mdpi.com/2076-3417/12/3/1319

Tracker

❖ Object Tracking: DeepSORT Pipeline



https://www.mdpi.com/2076-3417/12/3/1319

Tracker

❖ Coding Step 1: Install libraries

Dependencies

The code is compatible with Python 2.7 and 3. The following dependencies are needed to run the tracker:

- NumPy
- sklearn
- OpenCV

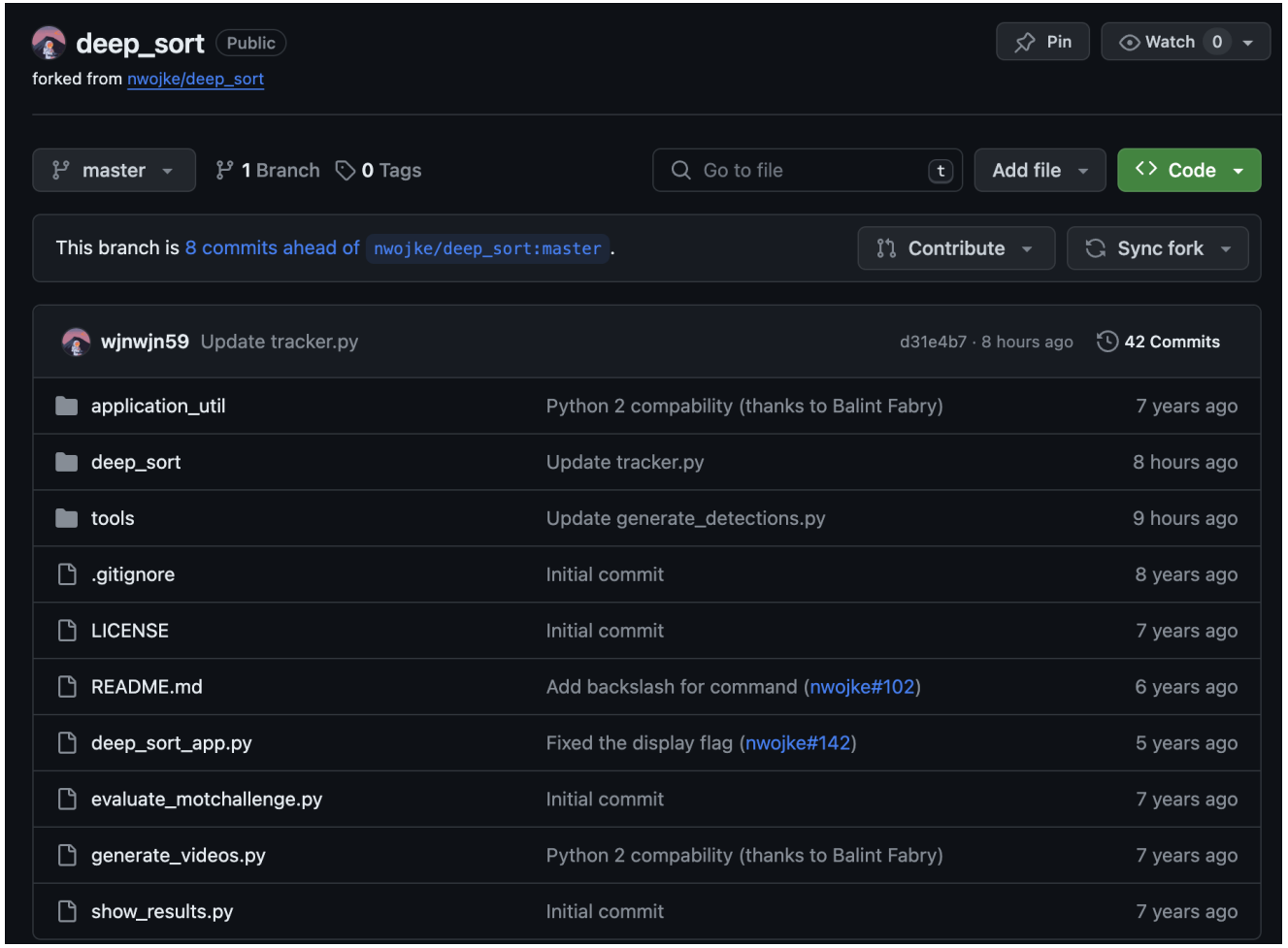
Additionally, feature generation requires TensorFlow (>= 1.0).

```
!pip install ultralytics -q
!pip install scikit-learn numpy opencv-python tensorflow spacy -q
!pip install gdown==4.6.0 -q
```

```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 699.8/699.8 kB 6.0 MB/s eta 0:00:00
```

Tracker

❖ Coding Step 2: Install DeepSORT source code




```
1 !git clone https://github.com/wjnwjn59/deep_sort.git
```

```
Cloning into 'deep_sort'...
remote: Enumerating objects: 167, done.
remote: Counting objects: 100% (25/25), done.
remote: Compressing objects: 100% (25/25), done.
remote: Total 167 (delta 9), reused 0 (delta 0), pack-reused 142
Receiving objects: 100% (167/167), 77.63 KiB | 1.23 MiB/s, done.
Resolving deltas: 100% (92/92), done.
```

Install modified version of DeepSORT

Tracker

❖ Coding Step 3: Install DeepSORT checkpoint

First, clone the repository:

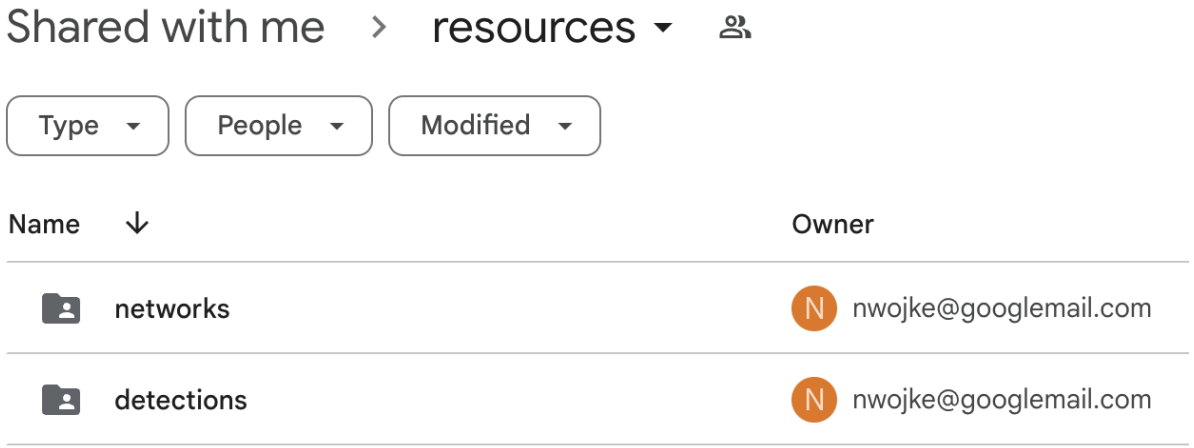
```
git clone https://github.com/nwojke/deep_sort.git
```

Then, download pre-generated detections and the CNN checkpoint file from here.

NOTE: The candidate object locations of our pre-generated detections are taken from the following paper:

```
F. Yu, W. Li, Q. Li, Y. Liu, X. Shi, J. Yan. POI: Multiple Object Tracking with
High Performance Detection and Appearance Feature. In BMTT, SenseTime Group
Limited, 2016.
```

We have replaced the appearance descriptor with a custom deep convolutional neural network (see below).



Tracker

❖ Coding Step 3: Install DeepSORT checkpoint

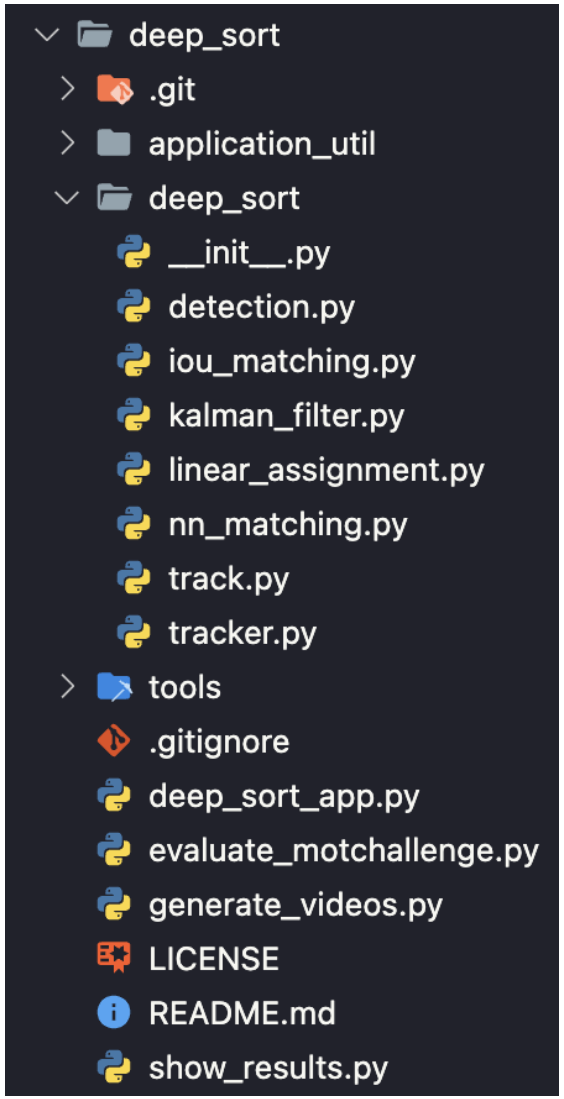
```
1 !gdown --no-check-certificate --folder https://drive.google.com/open?id=18fKzfqnqhqW3s9zwsCbnVJ5XF2JFeqMp
```

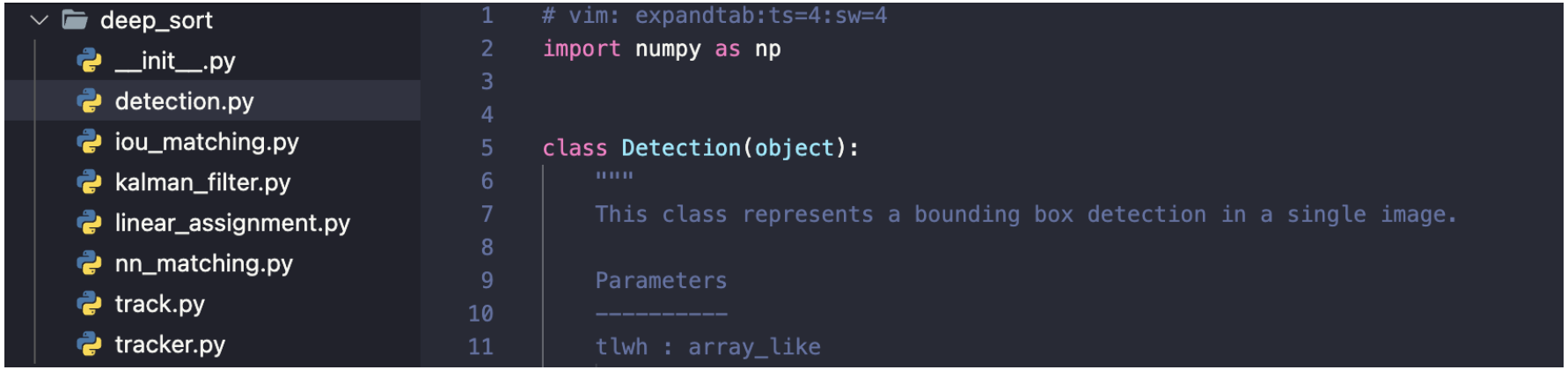
```
Retrieving folder list
Retrieving folder 1VVqtL0klSUvLnmBKS89il1EKC3IxUBVK detections
Retrieving folder 1qNWOpUtKG8GqEiL-LbBdXyvifUtcbOvc MOT16_POI_test
Processing file 1aEzvFHPK-N6hqLXMqhh3i9JJzn7WFUA3 MOT16-01.npy
Processing file 1h_ktJDBIEXaSBAA-RxKNYnL9e4fp2HPd MOT16-03.npy
Processing file 1ilOElwfYZLwQKH57HoYdXfuYhpPibfqF MOT16-06.npy
Processing file 1TajzH3GbumKmtYvKBvOtGERFGD0tStwG MOT16-07.npy
Processing file 1WB9Mi4RLVPHV4_20sVq7FdoeG5JYQ_J1 MOT16-08.npy
Processing file 1mksH9GWNT7zmcuq6rlRev8pevZz8Rfsm MOT16-12.npy
Processing file 1FVVhn_IpxQ_jkYhc0CUQHSQMm1SMTEBj MOT16-14.npy
Retrieving folder 1DcOcApOkxP3NdeIUXxVF1KNex6T6YDq3 MOT16_POI_train
Processing file 1Va__9NWU2ZCmaxIq4oIabi05NYWEOk1K MOT16-02.npy
Processing file 1EH7orgDPp7kqRY5OA0hEctcEtQnYq0Ea MOT16-04.npy
Processing file 1RCfHJx5ZoUecapbZCsgp0tCEiItvLsd8 MOT16-05.npy
Processing file 1VLOvn-mbpY0Q1rsMONQZhaEQIGEmyLQL MOT16-09.npy
Processing file 1SbMhOgYPvZ84xE8lRtXc7CLXJF86lwf4 MOT16-10.npy
Processing file 1a4w-HopWJHLFVi4e5wM_CEpv_ZgAVSys MOT16-11.npy
Processing file 1EOOPm2-09roynRlIxUCRSxBhChY8PA9D MOT16-13.npy
Retrieving folder 1m2ebLHB2JThZC8vWGDYEKGsevLssSkjo networks
Processing file 1hF6Cehn1SNZvh-M7FItSjEFojf_MVUba mars-small128.ckpt-68577
Processing file 1FkpWjshRY1YZC3dtQT9DNUbVZLu97uqi mars-small128.ckpt-68577.meta
Processing file 1bB66hP9voDXuoBoaCcKYY7a8IYzMMs4P mars-small128.pb
Retrieving folder list completed
Building directory structure
Building directory structure completed
Downloading...
```

Use gdown version 4.6.0 to be able to download a folder on drive

Tracker

❖ Coding Step 4: Import some DeepSORT modules





Import a class/function/variable of a python file by following its path

```
from deep_sort.deep_sort import nn_matching
from deep_sort.deep_sort.detection import Detection
from deep_sort.deep_sort.tracker import Tracker
from deep_sort.tools import generate_detections as gdet
```

Tracker

❖ Coding Step 5: Define DeepSORT class

```python
class DeepSORT:
    def __init__(
        self,
        model_path='resources/networks/mars-small128.pb',
        max_cosine_distance=0.7,
        nn_budget=None,
        classes=['objects']
    ):

        self.encoder = gdet.create_box_encoder(model_path, batch_size=1)
        self.metric = nn_matching.NearestNeighborDistanceMetric(
            'cosine',
            max_cosine_distance,
            nn_budget
        )
        self.tracker = Tracker(self.metric)

        key_list = []
        val_list = []
        for ID, class_name in enumerate(classes):
            key_list.append(ID)
            val_list.append(class_name)
        self.key_list = key_list
        self.val_list = val_list
```

Class DeepSORT
- Attributes + encoder + metric + tracker + key_list + val_list
- Methods + tracking

Tracker

❖ Coding Step 5: Define DeepSORT class

```python
    def tracking(
        self,
        origin_frame,
        bboxes,
        scores,
        class_ids
    ):
        features = self.encoder(origin_frame, bboxes)

        detections = [
            Detection(bbox, score, class_id, feature) \
                for bbox, score, class_id, feature in \
                    zip(bboxes, scores, class_ids, features)
        ]

        self.tracker.predict()
        self.tracker.update(detections)

        tracked_bboxes = []
        for track in self.tracker.tracks:
            if not track.is_confirmed() or track.time_since_update > 5:
                continue
            bbox = track.to_tlbr()
            class_id = track.get_class()
            conf_score = track.get_conf_score()
            tracking_id = track.track_id
            tracked_bboxes.append(
                bbox.tolist() + [class_id, conf_score, tracking_id]
            )

        tracked_bboxes = np.array(tracked_bboxes)

        return tracked_bboxes
```

Class DeepSORT
- Attributes + encoder + metric + tracker + key_list + val_list
- Methods + tracking

Tracker

❖ Coding Step 6: Create draw function

```python
# Draw bounding boxes and labels of detections
for bbox, score, class_id, id_ in zip(bboxes, scores, class_ids, ids):
    color = colors[class_id]

    x1, y1, x2, y2 = bbox.astype(int)

    # Draw rectangle
    cv2.rectangle(det_img, (x1, y1), (x2, y2), color, 2)

    # Draw fill rectangle in mask image
    cv2.rectangle(mask_img, (x1, y1), (x2, y2), color, -1)

    label = classes[class_id]
    caption = f'{label} {int(score * 100)}% ID: {id_}'
    (tw, th), _ = cv2.getTextSize(text=caption, fontFace=cv2.FONT_HERSHEY_SIMPLEX,
                                  fontScale=size, thickness=text_thickness)
    th = int(th * 1.2)

    cv2.rectangle(det_img, (x1, y1),
                  (x1 + tw, y1 - th), color, -1)
    cv2.rectangle(mask_img, (x1, y1),
                  (x1 + tw, y1 - th), color, -1)
    cv2.putText(det_img, caption, (x1, y1),
                cv2.FONT_HERSHEY_SIMPLEX, size, (255, 255, 255), text_thickness, cv2.LINE_AA)

    cv2.putText(mask_img, caption, (x1, y1),
                cv2.FONT_HERSHEY_SIMPLEX, size, (255, 255, 255), text_thickness, cv2.LINE_AA)
```

Tracker

❖ Coding Step 7: Create video tracking function

```
cap = cv2.VideoCapture(video_path)
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

if is_save_result:
    os.makedirs(save_dir, exist_ok=True)
    # Get the video properties
    fps = int(cap.get(cv2.CAP_PROP_FPS))

    # Define the codec and create the video writer
    fourcc = cv2.VideoWriter_fourcc(*'MJPG')

    save_result_name = 'output_video.avi'
    save_result_path = os.path.join(save_dir, save_result_name)
    out = cv2.VideoWriter(save_result_path, fourcc, fps, (width, height))
```

Create a VideoCapture instance with the name of save video path

If paramater is_save_results = True, we create a VideoWriter instance.

Tracker

❖ Coding Step 7: Create video tracking function

```
all_tracking_results = []
tracked_ids = np.array([], dtype=np.int32)
while True:
    ret, frame = cap.read()

    if not ret:
        break

    detector_results = detector.detect(frame)
    bboxes, scores, class_ids = detector_results

    tracker_pred = tracker.tracking(
        origin_frame=frame,
        bboxes=bboxes,
        scores=scores,
        class_ids=class_ids
    )
```

Iterate through each frame of the video. **For each frame, do:**

1. Detection.
2. Tracking with detection result.

Tracker

❖ Coding Step 7: Create video tracking function

```
if tracker_pred.size > 0:
    bboxes = tracker_pred[:, :4]

    class_ids = tracker_pred[:, 4].astype(int)
    conf_scores = tracker_pred[:, 5]
    tracking_ids = tracker_pred[:, 6].astype(int)

    # Get new tracking IDs
    new_ids = np.setdiff1d(tracking_ids, tracked_ids)

    # Store new tracking IDs
    tracked_ids = np.concatenate((tracked_ids, new_ids))

    result_img = draw_detection(
        img=frame,
        bboxes=bboxes,
        scores=conf_scores,
        class_ids=class_ids,
        ids=tracking_ids
    )
else:
    result_img=frame
```

Consider the tracking result:

- Ignore the None result -> Treat as normal frame.
- If not None, concat the tracked ID to the list of tracked IDs.
- Draw the tracked result of to current frame.

Tracker

❖ Coding Step 7: Create video tracking function

```
    if is_save_result == 1:
        out.write(result_img)

    # Break the loop if 'q' is pressed
    if cv2.waitKey(25) & 0xFF == ord('q'):
        break

# Release video capture
cap.release()
if is_save_result:
    out.release()
cv2.destroyAllWindows()

return all_tracking_results
```

After the last frame, break the loop and save (if activated) visualized tracking video.

Tracker

❖ Coding Step 8: Run object tracking on a video

1. Create Detector and Tracker instances

```
yolo_model_path = 'yolov8_mot_det.pt'

detector = YOLOv8(yolo_model_path)
tracker = DeepSORT()
```

2. Call the video_tracking() function

```
video_path = '/content/CityRoam.mp4'
all_tracking_results = video_tracking(
    video_path,
    detector,
    tracker,
    is_save_result=True
)
```

```
from IPython.display import HTML
from base64 import b64encode
import os

# Input video path
output_video_path = 'tracking_results/output_video.avi'

# Compressed video path
compressed_path = 'tracking_results/result_compressed.mp4'

os.system(f"ffmpeg -i {output_video_path} -vcodec libx264 {compressed_path}")
```

```
256
```

```
# Show video
mp4 = open(compressed_path,'rb').read()
data_url = "data:video/mp4;base64," + b64encode(mp4).decode()
HTML("""
<video width=600 controls>
      <source src="%s" type="video/mp4">
</video>
""" % data_url)
```

Tracker

❖ Coding Step 8: Run object tracking on a video

PaddleDetection MOT

❖ Introduction

PaddleDetection MOT

❖ Introduction

PaddleDetection MOT

❖ Introduction: Supported MOT in Paddle

Model libraries

▶ 1. General detection

▶ 2. Instance segmentation

▶ 3. Keypoint detection

▼ 4. Multi-object tracking PP-Tracking

Model	Introduction	Recommended scenarios	Accuracy	Configuration	Download
ByteTrack	SDE Multi-object tracking algorithm with detection model only	Edge-Cloud end	MOT-17 half val: 77.3	Link	Download
FairMOT	JDE multi-object tracking algorithm multi-task learning	Edge-Cloud end	MOT-16 test: 75.0	Link	Download
OC-SORT	SDE multi-object tracking algorithm with detection model only	Edge-Cloud end	MOT-16 half val: 75.5	Link	-

Other multi-object tracking models docs

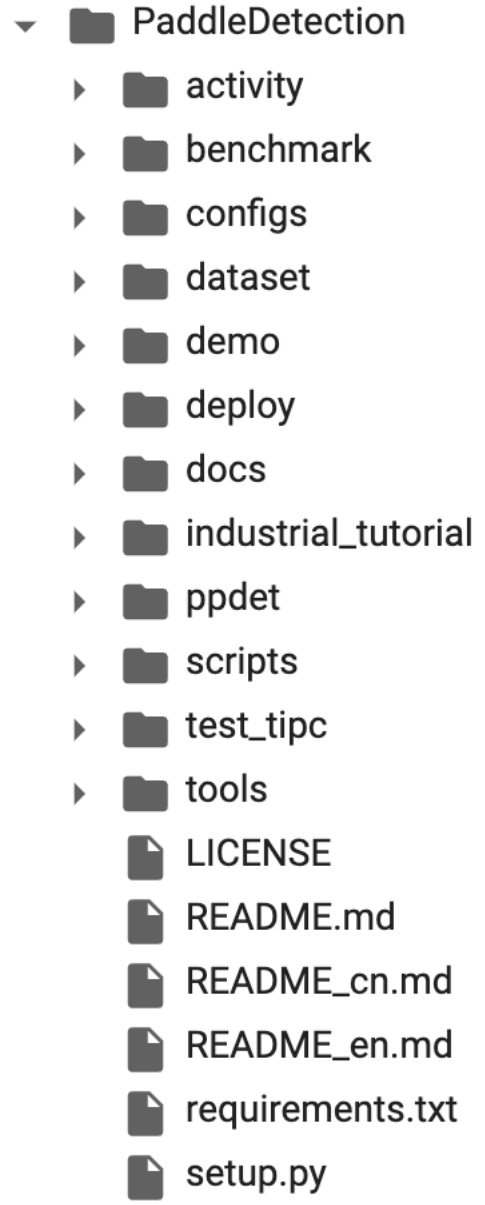
PaddleDetection MOT

❖ Coding Step 1: Install libraries

```
%%shell
git clone https://github.com/PaddlePaddle/PaddleDetection
pip install -q paddlepaddle-gpu pyclipper attrdict gdown -qqq wandb
cd PaddleDetection
git checkout develop
pip install -q -e .
```

```
Cloning into 'PaddleDetection'...
remote: Enumerating objects: 257131, done.
remote: Counting objects: 100% (19/19), done.
remote: Compressing objects: 100% (18/18), done.
remote: Total 257131 (delta 9), reused 1 (delta 1), pack-reused 257112
Receiving objects: 100% (257131/257131), 421.29 MiB | 26.55 MiB/s, done.
Resolving deltas: 100% (209712/209712), done.
```

https://github.com/PaddlePaddle/PaddleDetection



PaddleDetection MOT

❖ Coding Step 2: Test models

```
%cd PaddleDetection
!python ppdet/modeling/tests/test_architectures.py
```

```
/content/PaddleDetection
/content/PaddleDetection/ppdet/modeling/losses/detr_loss.py:341: SyntaxWarning: assertion
  assert (masks is not None and gt_mask is not None,
/content/PaddleDetection/ppdet/modeling/transformers/matchers.py:136: SyntaxWarning: asser
  assert (masks is not None and gt_mask is not None,
W0117 09:53:22.211380  3099 gpu_resources.cc:119] Please NOTE: device: 0, GPU Compute Capa
W0117 09:53:22.212481  3099 gpu_resources.cc:164] device: 0, cuDNN Version: 8.9.
.......
----------------------------------------------------------------------
Ran 7 tests in 0.882s

OK
```

PaddleDetection MOT

❖ Coding Step 3: Download a sample video

```
1 # Download demo video
2 %%shell
3 wget https://drive.google.com/uc?id=1f8tYWPQ93ID_FQUEaexDvUpkF975EmGl -O demo.mp4
```

```
--2024-01-17 09:53:24--  https://drive.google.com/uc?id=1f8tYWPQ93ID_FQUEaexDvUpkF975EmGl
Resolving drive.google.com (drive.google.com)... 74.125.128.138, 74.125.128.139, 74.125.128.101, ...
Connecting to drive.google.com (drive.google.com)|74.125.128.138|:443... connected.
HTTP request sent, awaiting response... 303 See Other
Location: https://drive.usercontent.google.com/download?id=1f8tYWPQ93ID_FQUEaexDvUpkF975EmGl [following]
--2024-01-17 09:53:24--  https://drive.usercontent.google.com/download?id=1f8tYWPQ93ID_FQUEaexDvUpkF975EmGl
Resolving drive.usercontent.google.com (drive.usercontent.google.com)... 173.194.69.132, 2a00:1450:4013:c02::84
Connecting to drive.usercontent.google.com (drive.usercontent.google.com)|173.194.69.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 45714742 (44M) [video/mp4]
Saving to: ‘demo.mp4’

demo.mp4            100%[===================>]  43.60M  32.4MB/s    in 1.3s

2024-01-17 09:53:29 (32.4 MB/s) - ‘demo.mp4’ saved [45714742/45714742]
```

PaddleDetection MOT

❖ Coding Step 3: Download a sample video

PaddleDetection MOT

❖ Coding Step 4: Inference with PPYOLOe and ByteTrack

```
1 # Inference with PPYOLOe
2 !python tools/infer_mot.py -c configs/mot/bytetrack/bytetrack_ppyoloe.yml --video_file=demo.mp4
```

```
[01/17 09:53:33] ppdet.utils.download INFO: Downloading ppyoloe_crn_l_36e_640x640_mot17half.pdparam
100% 207958/207958 [00:39<00:00, 5244.19KB/s]
[01/17 09:54:17] ppdet.utils.checkpoint INFO: Finish resuming model weights: /root/.cache/paddle/we
[01/17 09:56:16] ppdet.data.source.mot INFO: Length of the video: 750 frames.
[01/17 09:56:16] ppdet.engine.tracker INFO: Starting tracking video demo.mp4
100% 750/750 [01:54<00:00,  6.55it/s]
ffmpeg version 4.4.2-0ubuntu0.22.04.1 Copyright (c) 2000-2021 the FFmpeg developers
  built with gcc 11 (Ubuntu 11.2.0-19ubuntu1)
  configuration: --prefix=/usr --extra-version=0ubuntu0.22.04.1 --toolchain=hardened --libdir=/usr/
  libavutil      56. 70.100 / 56. 70.100
  libavcodec     58.134.100 / 58.134.100
  libavformat    58. 76.100 / 58. 76.100
  libavdevice    58. 13.100 / 58. 13.100
  libavfilter     7.110.100 /  7.110.100
  libswscale      5.  9.100 /  5.  9.100
  libswresample   3.  9.100 /  3.  9.100
  libpostproc    55.  9.100 / 55.  9.100
Input #0, image2, from 'output/ppyoloe/mot_outputs/demo/%05d.jpg':
  Duration: 00:00:30.00, start: 0.000000, bitrate: N/A
  Stream #0:0: Video: mjpeg (Baseline), yuvj420p(pc, bt470bg/unknown/unknown), 1920x1080 [SAR 1:1 D
```

```
# Inference with PPYOLOe
!python tools/infer_mot.py -c
configs/mot/bytetrack/bytetra
ck_ppyoloe.yml --
video_file=demo.mp4 --
scaled=True --save_videos --
output_dir=output/ppyoloe
```

PaddleDetection MOT

❖ Coding Step 4: Inference with YOLOX and ByteTrack

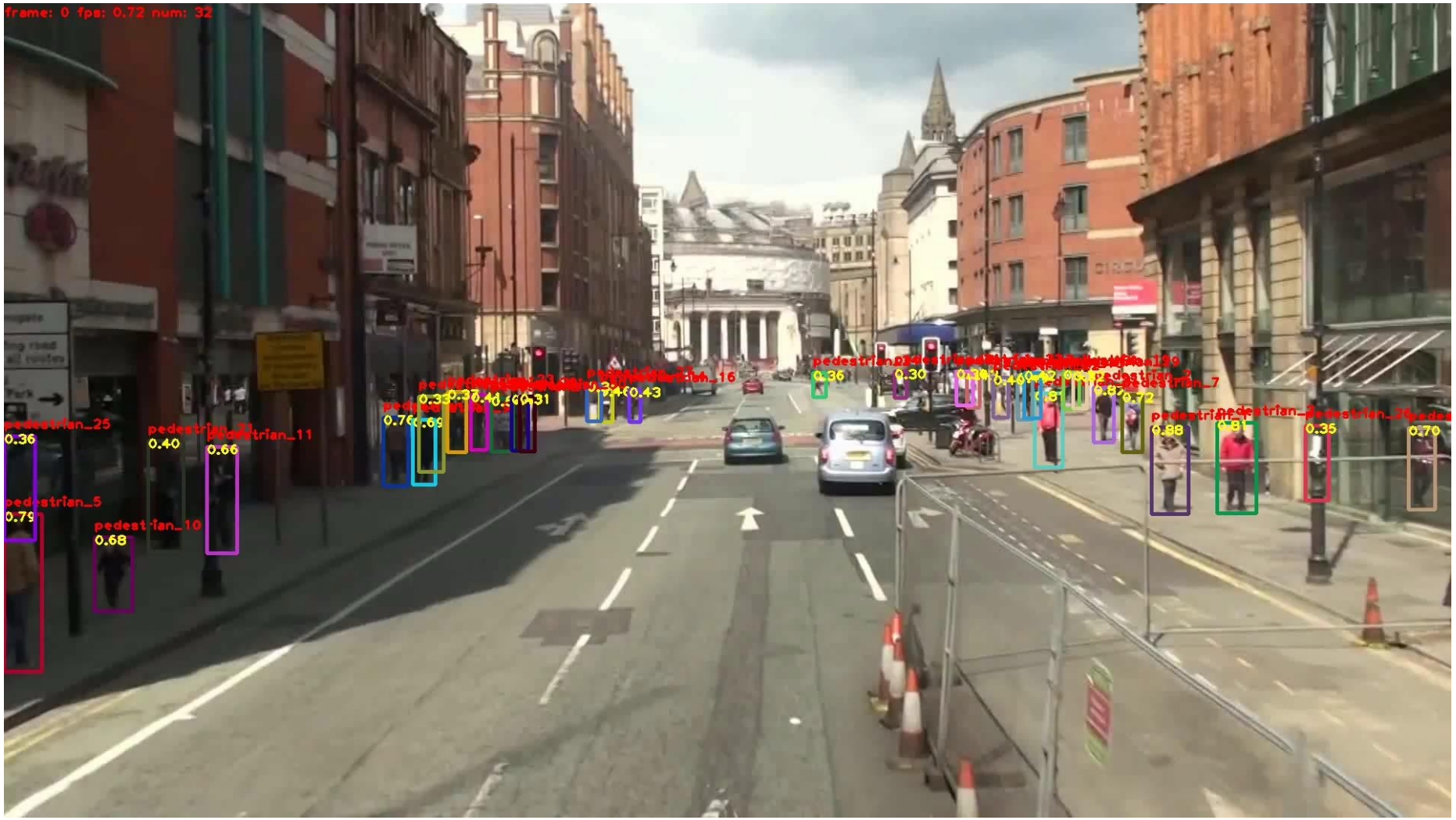
```
1 # Inference with YOLOX
2 !python tools/infer_mot.py -c configs/mot/bytetrack/bytetrack_yolox.yml --video_file=demo.mp4
```

```
[01/17 10:00:42] ppdet.utils.download INFO: Downloading yolox_x_24e_800x1440_mix_det.pdparams fr
100% 387185/387185 [00:51<00:00, 7459.55KB/s]
[01/17 10:01:38] ppdet.utils.checkpoint INFO: Finish resuming model weights: /root/.cache/paddle,
[01/17 10:03:40] ppdet.data.source.mot INFO: Length of the video: 750 frames.
[01/17 10:03:40] ppdet.engine.tracker INFO: Starting tracking video demo.mp4
100% 750/750 [03:15<00:00,  3.84it/s]
ffmpeg version 4.4.2-0ubuntu0.22.04.1 Copyright (c) 2000-2021 the FFmpeg developers
  built with gcc 11 (Ubuntu 11.2.0-19ubuntu1)
  configuration: --prefix=/usr --extra-version=0ubuntu0.22.04.1 --toolchain=hardened --libdir=/u
  libavutil      56. 70.100 / 56. 70.100
  libavcodec     58.134.100 / 58.134.100
  libavformat    58. 76.100 / 58. 76.100
  libavdevice    58. 13.100 / 58. 13.100
  libavfilter     7.110.100 /  7.110.100
  libswscale      5.  9.100 /  5.  9.100
  libswresample   3.  9.100 /  3.  9.100
  libpostproc    55.  9.100 / 55.  9.100
Input #0, image2, from 'output/yolox/mot_outputs/demo/%05d.jpg':
  Duration: 00:00:30.00, start: 0.000000, bitrate: N/A
  Stream #0:0: Video: mjpeg (Baseline), yuvj420p(pc, bt470bg/unknown/unknown), 1920x1080 [SAR 1:
Stream mapping:
  Stream #0:0 -> #0:0 (mjpeg (native) -> h264 (libx264))
Press [q] to stop, [?] for help
[libx264 @ 0x57829483d280] using SAR=1/1
[libx264 @ 0x57829483d280] using cpu capabilities: MMX2 SSE2Fast SSSE3 SSE4.2 AVX FMA3 BMI2 AVX2
[libx264 @ 0x57829483d280] profile High, level 4.0, 4:2:0, 8-bit
[libx264 @ 0x57829483d280] 264 - core 163 r3060 5db6aa6 - H.264/MPEG-4 AVC codec - Copyleft 2003-
Output #0, mp4, to 'output/yolox/mot_outputs/demo/../demo_vis.mp4':
```

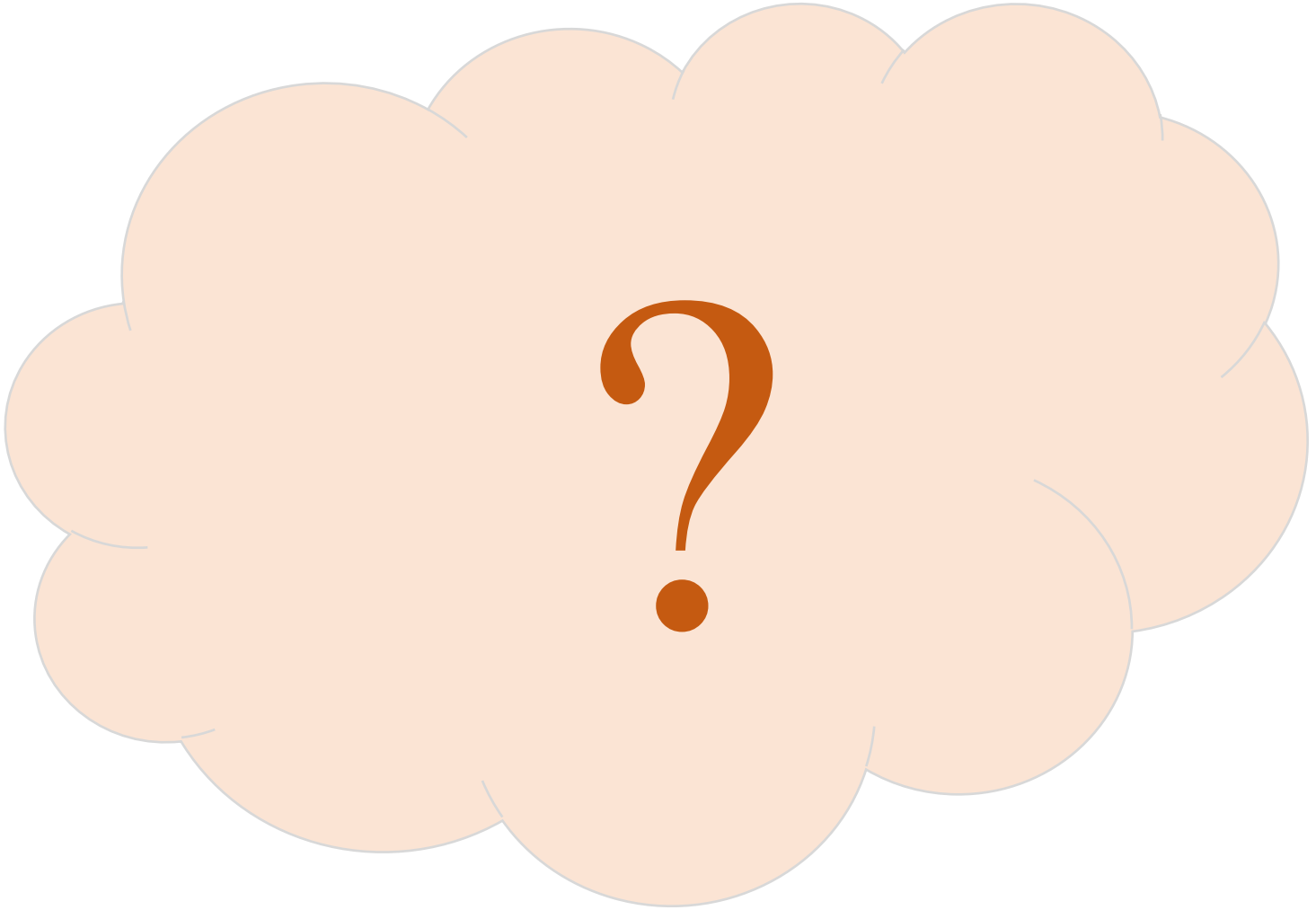
```
# Inference with YOLOX
!python tools/infer_mot.py -c
configs/mot/bytetrack/bytetra
ck_yolox.yml --
video_file=demo.mp4 --
scaled=True --save_videos --
output_dir=output/yolox
```

PaddleDetection MOT

❖ Coding Step 4: Inference with YOLOX and ByteTrack



Question




Thank you!