Personal Project Report

Project Title: Object Detection and Tracking for Surveillance Using YOLOv3

ABSTRACT

Problem Statement:

In urban areas with high traffic density, real-time surveillance and monitoring pose significant challenges due to dynamic object movement, occlusions, and the need for fast, accurate detection. Traditional methods are either slow or inaccurate. A robust system capable of object detection and tracking is essential for modern surveillance needs.

Objectives:

- Create a custom dataset with 8 relevant object classes.
- Train a YOLOv3 model for real-time object detection.
- Integrate Deep SORT algorithm for object tracking.
- Apply the system in traffic surveillance scenarios for tasks such as vehicle velocity estimation and tracking distances between objects.

Methodology:

We used Google OpenImagesV6 dataset and LabelImg for dataset creation and annotation. A custom YOLOv3 model was trained using the Darknet framework on Google Colab. Detected objects were tracked using the Deep SORT algorithm, which utilizes motion and appearance features. YOLO weights were converted to TensorFlow format for compatibility.

Key Results:

- Accurate real-time object detection for 8 classes: Person, Motorbike, Traffic light, Car, Bus, Truck, Bicycle, Umbrella.
- Effective object tracking using Deep SORT with unique IDs and minimal ID switches.
- Demonstrated application in traffic surveillance scenarios.

GitHub Link for Code: https://github.com/sandipanrakshit34/Object-Detection-and-Tracking-for-Surveillance

Scope of the Project: The system is limited to 8 classes and performs best in urban surveillance scenarios. It can be extended to include velocity tracking, density mapping, and alert systems.

Review of Object Detection Algorithms: YOLO, SSD, Faster-RCNN, and RetinaNet are popular one-stage and two-stage object detectors. YOLOv3 is known for its real-time performance and good accuracy tradeoff.

Object Tracking Techniques: Simple Online Realtime Tracker (SORT) and Deep SORT are widely used. Deep SORT enhances SORT with visual appearance features, reducing ID switches.

Gaps in Existing Methods

- Speed vs. accuracy tradeoff
- Identity switches in tracking
- Limited real-time deployment due to computational needs

Addressing the Gaps

This project integrates YOLOv3 and Deep SORT for a balanced solution providing both speed and tracking robustness. Conversion to TensorFlow enables Deep SORT integration.

Dataset Creation and Labeling

- Dataset: Google OpenImagesV6
- Classes: Person, Car, Bus, Truck, Bicycle, Motorbike, Umbrella, Traffic Light
- Tool: LabelImg for bounding box annotation

YOLOv3 Model Training

- Framework: Darknet
- Config adjustments:
  - classes=8, filters=(8+5)*3 = 39
  - max_batches=16000
  - batch=64, subdivisions=16
- Training on Google Colab with pre-trained convolutional weights

Object Detection using YOLOv3

- Real-time detection
- Detection output: bounding boxes, confidence scores, class labels

Object Tracking with Deep SORT

- Conversion of weights to TensorFlow using custom script
- Deep SORT integrates Kalman filter, Hungarian algorithm, appearance features
- Assigns unique IDs, tracks object movement across frames

Dataset Creation and Labeling :

- We used images from Google's OpenImagesV6 dataset, publicly available online. It is a very big dataset with more than 600 different categories of an object. The dataset contains the bounding box, segmentation, or relationship annotations for these objects.
- We collected images of 8 classes. "Person", "Motorbike", "Traffic light", "Car", "Bus", "Truck", "Bicycle" and "Umbrella".
- We used LebelImg to label each and every Image for collecting dimensions of anchor boxes.

Object Detection :

- Object Detection is a common Computer Vision problem which deals with identifying and locating object of in the image.(Object Recognition recognise what kind of object they are : class labels)
- Interpreting the object localisation can be done in various ways, including creating a bounding box around the object or marking every pixel in the image which contains the object (called segmentation).
- With the need of real time object detection, many one-step object detection architectures have been proposed, like YOLO, YOLOv2, YOLOv3, SSD, RetinaNet etc. which try to combine the detection and classification step.
- In our project we will use YOLOv3 for training our object detection model.

YOLO v3 :

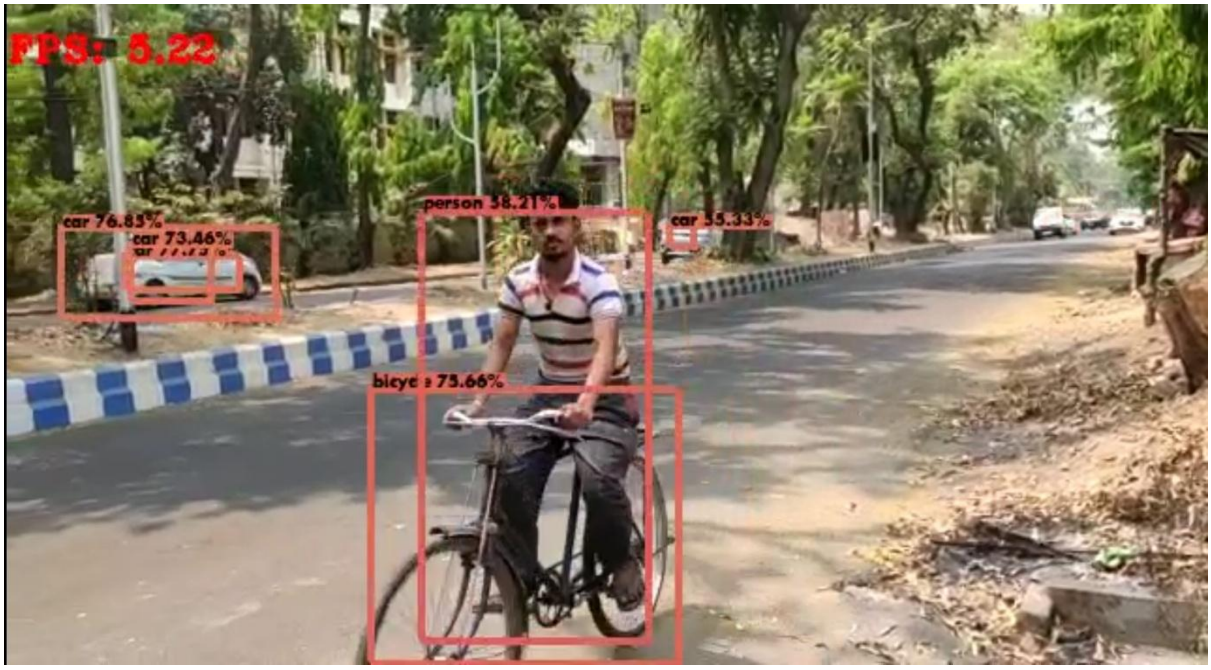
We only look once (YOLO) is an object detection system targeted for real-time processing.YOLO is able to perform object detection and recognition at the same time.It is a detector applying a single neural network which -

* Predict bounding boxes
* Multilabel classification

Grid cell:

- YOLO divides the input image into an S×S grid. Each grid cell predicts only one object.
- For example, the yellow grid cell below tries to predict the "person" object whose center (the blue dot) falls inside the grid cell.Each grid cell predicts a fixed number of boundary boxes. In this example,the yellow grid cell makes two boundary box predictions(blue boxes) to locate where the person is.

Object Detection Output



Object Tracking :

It is the process of locating moving objects over time in videos. It involves-

- Taking an initial set of object detection
- Creating unique ID for each of the detection
- Tracking the object over time
- Maintaining the ID assignment

Difference between Object Detection and Tracking :

- Object detection is simply about identifying and locating all known objects in a scene. Object tracking is about locking onto a particular moving object(s) in real-time.
- Object detection can occur on still photos while object tracking needs video feed. Object detection can be used as object tracking if we run object detection every frame per second.
- Running object detection as tracking can be computationally expensive and it can only track known objects. Thus object detection requires a form of classification and localization.

DeepSORT :

The most popular and one of the most widely used, elegant object tracking framework is Deep SORT, an extension to SORT (Simple Real time Tracker). Improves the matching procedures and reduces the number of identity switches by adding visual appearance descriptor or appearance features. It obtains higher accuracy with the use of

1) Motion measurement
2) Appearance features

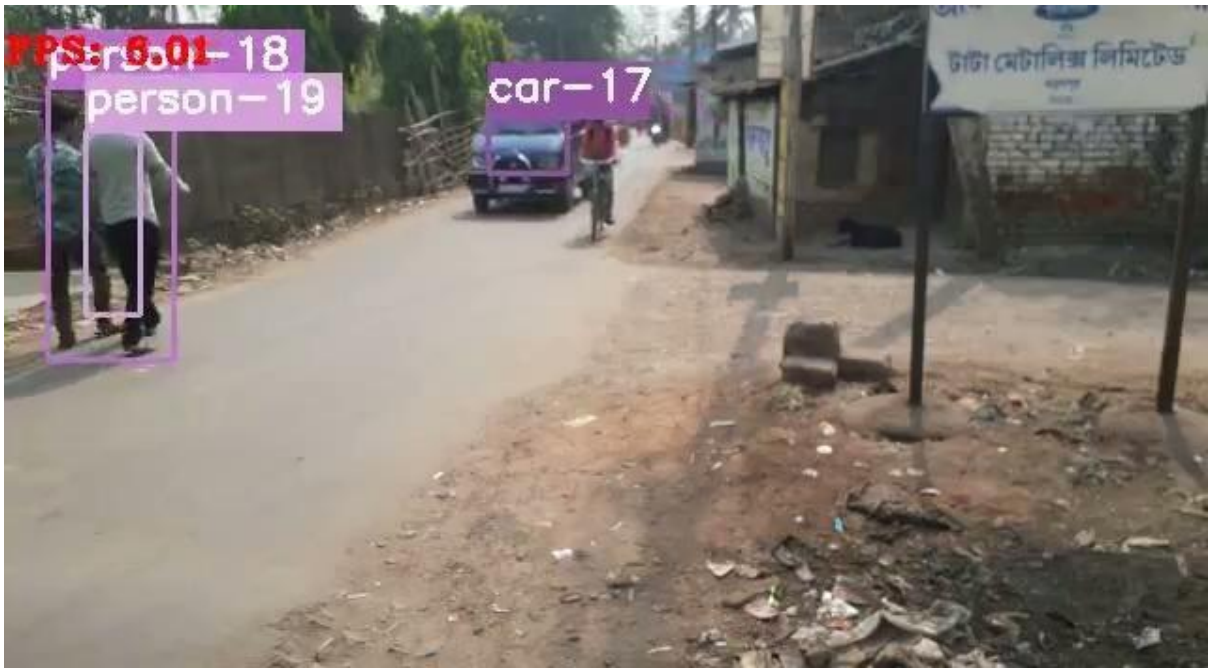
It applies Kalman Filtering, Hungarian method, Mean shift,Optical Flow,Feature Vector

Converting YOLO v3 weights to tensorflow Model :

- We will use our previously trained YOLOV3 weight for tracking objects.So we need to copy our trained yolov3_custom.weight to weight folder and also copy obj.names file to labels folder
- To apply this weight to Deep Sort first we need to convert the yolov3_custom.weight to Tensorflow model to work with Deep Sort.
- For this purpose we write a script load_weights.py and we convert weights to yolov3_custom.tf format.

Now using the tensorflow model and with the help of Deep Sort we are able to track the objects which are previously detected by YOLOV3.

Object Tracking Output



Application of Object Detection and Tracking :

- Video surveillance
- Pedestrian detection
- Anomaly detection
- People Counting
- Self driving cars
- Face detection
- Security
- Manufacturing Industry

Tools Used

- Google Colab, Darknet, TensorFlow, OpenCV, LabelImg

Snapshots of Detection and Tracking

- Detected object with bounding box and class label
- Continuous tracking with unique ID

Applications Demonstrated

- Vehicle counting
- Person tracking
- Distance estimation
- Zone entry/exit monitoring

Challenges :

- Speed for real-time detection
- Limited data
- Class imbalance
- Illumination
- Multiple spatial scales and aspect ratios
- Positioning
- Rotation
- Dual priorities: object classification and localization
- Occlusion
- Mirroring

Future Work :

We will further try to improve this project by adding extra feature like -

1. Counting number of vehicle or person entered in a particular frame
2. We can use Density Map
3. In or Out of a Specific Zone

Conclusion

This project demonstrates the effectiveness of combining YOLOv3 and Deep SORT for real-time surveillance in traffic-heavy environments. It provides a practical framework for expanding to broader smart city and public safety applications.

REFERENCES

1. Joseph Redmon et al., "YOLOv3: An Incremental Improvement," arXiv, 2018.
2. Wojke et al., "Simple Online and Realtime Tracking with a Deep Association Metric," 2017.
3. Google OpenImages Dataset V6, https://storage.googleapis.com/openimages/web/index.html
4. Darknet Framework: https://github.com/AlexeyAB/darknet
5. Deep SORT GitHub Repository: https://github.com/nwojke/deep_sort