BRIEF EXPLANATORY DOCUMENT

VEHICLE SPEED ESTIMATION USING YOLO AND DEEPSORT

PROJECT OVERVIEW

GitHub Repository: (https://github.com/itheaks/HackTech_Showcasing)
Submitted by: **AMIT KUMAR SINGH (aksmlibts@gmail.com)**

INTRODUCTION

In the context of the HackTech competition, this project presents a comprehensive solution for real-time vehicle speed estimation using advanced computer vision techniques. Leveraging the power of YOLO (You Only Look Once) for object detection and DeepSort for object tracking, the project goes beyond simple detection to provide accurate speed estimations for tracked vehicles.

KEY COMPONENTS

1. CODE ORGANIZATION

The project's GitHub repository, [HackTech_Final] (https://github.com/itheaks/HackTech_Final), showcases a well-organized codebase. The "Just for Reference" folder contains an IPython notebook, providing an insightful guide for users and reviewers to navigate through the code with ease. This thoughtful organization enhances the project's accessibility and encourages collaboration.

2. OBJECT DETECTION AND TRACKING

OBJECT DETECTION WITH YOLO

The project leverages YOLO, an advanced object detection system known for its efficiency in real-time applications. YOLO's unique capability to predict multiple bounding boxes and class probabilities in a single pass excels in identifying vehicles and diverse objects within video streams.

OBJECT TRACKING WITH DEEPSORT

DeepSort is employed to enhance tracking precision. seamlessly integrating with YOLO's detections, DeepSort associates identified objects across frames, ensuring a reliable tracking mechanism. This synergistic approach guarantees the system's robust performance in tracking vehicles through complex scenarios.

3. SPEED ESTIMATION LOGIC

The project goes beyond mere detection and tracking by incorporating a dedicated script for speed estimation. The speed estimation logic is a crucial aspect, enabling the system to quantify the movement of tracked vehicles. This functionality is vital for applications such as traffic monitoring, surveillance, and intelligent transportation systems.

4. IMPLEMENTATION HIGHLIGHTS

IMPORT STATEMENTS

The script begins by importing essential libraries, including `hydra` for configuration management, `torch` for deep learning capabilities, `cv2` for computer vision tasks, and `numpy` for numerical operations. These libraries lay the foundation for the project's functionality.

CONFIGURATION AND INITIALIZATION

Hydra is utilized for configuration management, allowing users to customize parameters through command-line arguments or configuration files. The DeepSort tracker is initialized with configurations loaded from a YAML file, providing flexibility in adapting to different scenarios.

UTILITY FUNCTIONS

Several utility functions enhance the script's functionality. Functions for converting bounding box coordinates, computing colors for labels, drawing borders, and calculating the direction between two points contribute to the overall efficiency of the system.

OBJECT COUNTING

The script features object counting capabilities, enabling the system to determine the number of vehicles entering and leaving a specified region. This functionality enhances the system's utility in traffic analysis and management.

OUTPUT DISPLAY

The script's output is not just limited to raw data. It provides visually informative frames with annotated bounding boxes, object labels, trails, object count, and average speed. This user-friendly output is designed for easy interpretation and integration into broader applications.

5. SPEED ESTIMATION PROCESS

EUCLIDEAN DISTANCE CALCULATION

The heart of the speed estimation process lies in the `estimatespeed` function. This function calculates the Euclidean distance between the center points of an object in consecutive frames. This distance becomes a crucial metric for determining the object's movement.

PIXELS PER METER (PPM)

A constant `ppm` (pixels per meter) is introduced to convert the calculated distance from pixels to meters, laying the groundwork for consistent unit conversion.

TIME CONSTANT

The speed estimation process involves defining a time constant (`time_constant`). This constant is set to convert the distance traveled to speed in kilometers per hour, taking into account the frame rate of the video stream.

SPEED CALCULATION

Using the established constants, the speed of the object is calculated through a simple formula: `speed = distance / time`. This step transforms the raw distance data into a meaningful speed metric.

ROUNDING AND RETURN

The final step involves rounding the calculated speed to an integer value, providing a practical and interpretable output. The speed is then returned for further integration into the system.

6. DATASETS AND LABELS

DATASETS

For training and testing purposes, the project utilizes the COCO datasets, which are widely recognized in the computer vision community. The datasets include a diverse set of images capturing various scenarios and objects.

LABELS

A comprehensive list of labels is employed for object categorization. These labels cover a wide range of objects, including vehicles (cars, trucks, buses), pedestrians, and various other entities commonly encountered in urban environments.

CONCLUSION

In conclusion, the project successfully combines the strengths of YOLO and DeepSort to create a sophisticated vehicle speed estimation system. The script's modular design, clear organization, and integration of speed estimation make it a robust solution for real-world applications. This project lays the foundation for advanced applications in traffic management, surveillance, and intelligent transportation systems.

7. REFERENCES

Given videos:
https://drive.google.com/drive/folders/1-b3UaecDVr3imt0hEuTQCRwEhX9A_7I-
My Code:
https://github.com/itheaks/HackTech_Showcasing
Use case:
https://colab.research.google.com/drive/182WSXPsMPwhhnefaxzE978-81uKAr0xU?usp=sharing
Output Videos:
https://drive.google.com/drive/folders/1rnJCupHWDbtNLckp58xRB4BlDk7RmZe0?usp=sharing

THANK YOU