LICENSE PLATE DETECTION

INTRODUCTION:

This project presents a complete pipeline for detecting vehicles, identifying their license plates, extracting the alphanumeric characters, and visualizing the results. The solution is built using advanced computer vision tools like YOLOv8 for object detection, EasyOCR for optical character recognition (OCR), and the SORT algorithm for multi-object tracking

OBJECTIVE:

The goal of this project is to automatically

- Detect moving vehicles in a video stream.
- Detect and crop the number plates from these vehicles.
- Recognize the characters from the license plates using OCR.
- Track vehicles consistently across frames.
- Visualize results in an annotated video.
- Interpolate missing detection data to ensure continuity.

TOOLS AND LIBRARIES USED:

Library	Purpose
YOLOv8 (Ultralytics)	For vehicle and number plate detection
OpenCV	For video reading, writing, and image processing
EasyOCR	For reading license plate numbers
SORT (Simple Online Realtime Tracker)	For tracking vehicle IDs across frames
Supervision	For visual overlays and annotations
NumPy, Pandas	For data handling and manipulation
SciPy	For interpolating missing frame data

WORKFLOW OVERVIEW:

- Preprocessing**:** Video frames are enhanced using contrast and sharpening techniques
- Detection: Vehicles and number plates are detected using YOLOv8.
- Tracking: Each vehicle is assigned a unique ID using SORT.
- OCR Recognition: License plate images are cropped, thresholded, and recognized using EasyOCR.
- Post-Processing: OCR text is validated and formatted for standard plate structure.
- Data Interpolation: Bounding boxes for missing detections are filled in using linear interpolation.
- Visualization: Final output video shows bounding boxes, plate crops, and recognized text.

MODULES AND CODE EXPLANATION:

Installation & Initialization:

Used pip to install required packages: !pip install ultralytics easyocr filterpy supervision

Enhancement Module:

- Converts each frame to LAB color space.
- Applies CLAHE (contrast enhancement).
- Sharpens frame using kernel.
- Saves the enhanced video for improved detection and OCR accuracy.

Functions Module:

- EasyOCR is initialized for English.
- Misread characters are corrected using mappings (O ↔ 0, etc.).
- We have defined some functions :
  - read_license_plate: Reads & validates plate number.
  - format_license: Corrects OCR output based on known patterns.
  - get_car: Matches plate with a vehicle using bounding box containment.
  - write_csv: Saves structured output to CSV.

Detection & Tracking:

Loads two YOLO models:

- yolov8n.pt for vehicles.
- Custom trained number_plate_best.pt for license plates.

Each frame is processed to:

- Detect and track vehicles.
- Detect number plates.
- Crop plate and recognize number using OCR.
- Link plates to vehicles via bounding box overlap.
- Save results to dictionary and export to CSV

Handling Missing Plates Module:

- Identifies missing frames for each car_id.
- Uses linear interpolation (via scipy.interpolate.interp1d) to fill bounding box gaps.
- Adds interpolated results to a new CSV (test_interpolated_results.csv).

Output Module:

- Loads the video and interpolated CSV.
- Draws:
  - Green border around each car.
  - Red box around license plates.
  - Cropped license plate above the vehicle.
  - Plate number text in white box.
- Writes final annotated video (cars_output.mp4).

OUTPUT FILES:

File Name	Description
cars.mp4	Enhanced video for better OCR performance
test_results.csv	Raw detection + OCR output
test_interpolated_results.csv	Smoothed data after interpolating missing frames
Cars_output.mp4	Final output video with visualization

SAMPLE OUTPUT:

Each car appears with:

- ❖ Green tracking box
- ❖ Red plate detection box
- ❖ Cropped plate image above the car
- ❖ Recognized license number in readable text

CHALLENGES FACED:

Issue	Solution
OCR misreading characters	Character mapping (e.g., 'O' → '0')
Missing detection in frames	Bounding box interpolation
Low contrast and blur	Frame enhancement with CLAHE and sharpening
OCR misalignment with vehicles	Bounding box containment logic

CONCLUSION:

This project showcases a complete computer vision pipeline for real-time number plate detection and recognition using open-source tools. By combining detection, tracking, and OCR with post-processing and visualization, the system is robust and suitable for applications like traffic monitoring, law enforcement, and automated toll systems.

FUTURE IMPROVEMENTS (OPTIMIZED FOR INDIAN LICENSE PLATE DETECTION):

- ❖ Train a YOLO model specifically on Indian number plates
  → Improves accuracy by learning region-specific fonts, sizes, and formats.
- ❖ Integrate multilingual OCR (Hindi, English)
  → Supports plates with regional scripts and improves robustness in different states.
- ❖ Improve OCR preprocessing
  → Add adaptive thresholding, denoising, and edge detection to boost EasyOCR accuracy.
- ❖ Use lightweight models for real-time deployment
  → Convert the system to use YOLOv8n or YOLOv5s + TFLite for edge devices.
- ❖ Add confidence-based filtering
  → Automatically discard low-confidence detections and reduce false positives.
- ❖ Support for night-time and low-light detection
  → Use IR-enhanced preprocessing or train with night-time datasets.
- ❖ Deploy as a web app or API service
  → Make the system accessible for smart traffic systems or police use.

KEY FINDINGS:

- ❖ YOLOv8 is effective for real-time detection of both vehicles and number plates, even on standard video inputs.
- ❖ EasyOCR can recognize Indian license plates, but accuracy improves significantly with preprocessing (sharpening + thresholding).
- ❖ Character misclassification is common (e.g., 'O' ↔ '0'), but can be corrected using mapping dictionaries and format validation.

- SORT tracking maintains vehicle identity across frames, making multi-frame analysis possible.
- Missing detection frames can be filled using interpolation, improving continuity and data reliability.
- Final visualization gives a complete understanding of detections and recognition results, making the output user-friendly.