LICENSE PLATE RECOGNITION

By Yehia SHORIM

Image Processing Project Documentation

1) Introduction and Project Overview

This project implements a **License Plate Recognition (LPR)** system using classical image processing techniques.

During our research and project implementation we found out that LPR systems vary widely in terms of techniques, performance, and complexity. While license plate recoginition may seem basic and easy to implement it is **highly detailed and surprisingly challenging**, requiring careful consideration of preprocessing, segmentation, feature extraction, and character recognition especially in real word since it is a critical tool used in all countries all over the world. Almost all governments use it for traffic monitoring, parking management, toll collection, law enforcement and much more.

In our projects , our goal is not to achieve a real word functional license plate recognition system, but to **develop something similar to the real word idea while mainly implementing image processing techinques we studied in our course.**

The LPR system we developed detects and reads European license plates from images, segmenting individual characters and matching them against preprocessed templates using **Histogram of Oriented Gradients (HOG)** features.

➔ Key Project objectives:

- Preprocess and normalize templates and test plates.
- Segment each character accurately, even in noisy or blurred images.
- Resolve ambiguities between visually similar characters

2) Project dataset

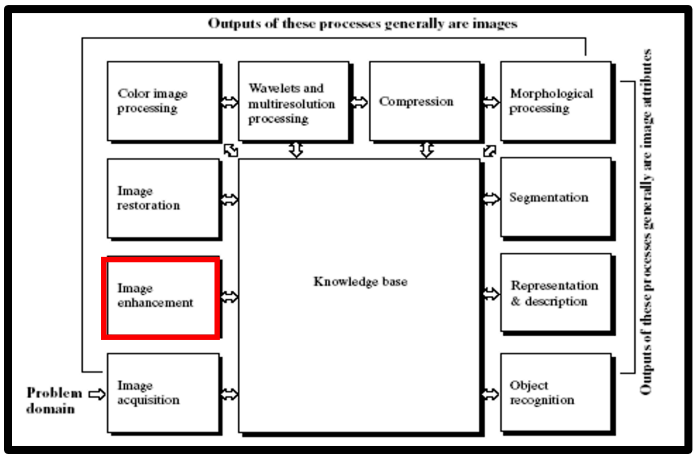
➔ We used for our project cropped European license plates , Here is a sample of the plates we used:

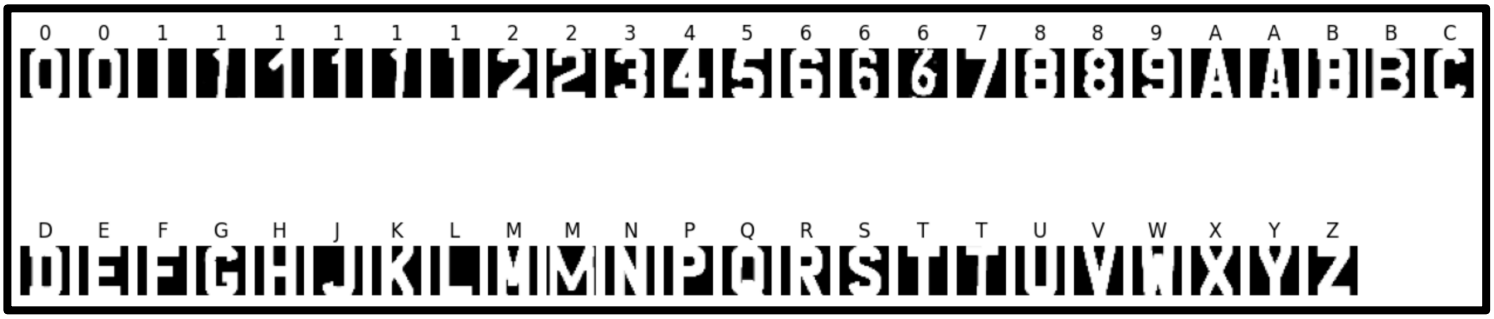
3) WorkFlow and Project Pipeline

Our LPR pipeline consists of four main stages:

Stage 1: Preprocessing

- **Templates Processing and enhancement :**
  - Convert to grayscale and denoise using Gaussian blur.
  - Apply **adaptive thresholding + Otsu's method** for binarization.
  - Morphological operations (opening + closing) to remove noise and small gaps.
  - Crop tightly to character bounding boxes and resize to **32x32** with padding.
  - Apply **dilation** to normalize stroke width.

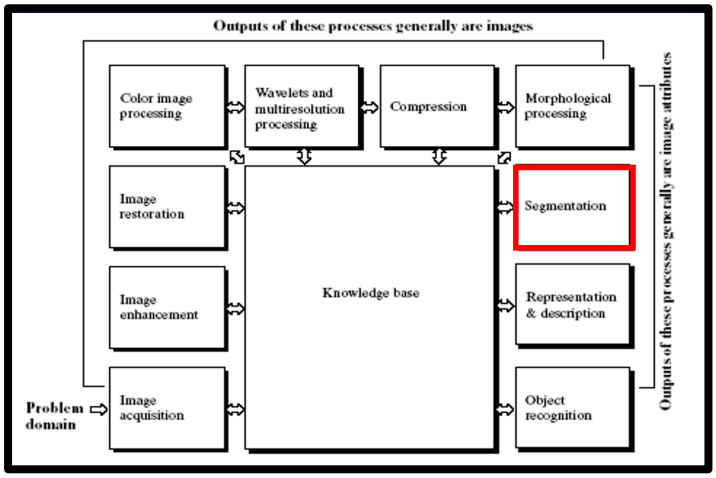




- **Test Plates Processing and enhancement:**
  - Convert to grayscale and enhance contrast using **CLAHE**.
  - Apply sharpness correction if image is detected as **blurry** (is_blurry function).
  - Binarization using adaptive threshold + Otsu.
  - Morphological cleanup for better segmentation.

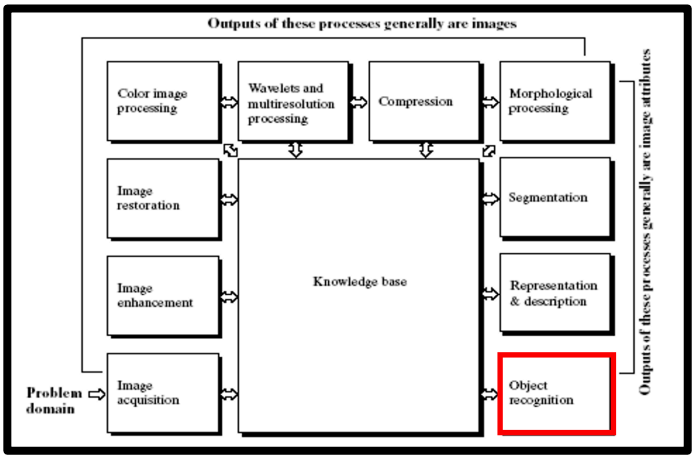
Stage 2: Character Segmentation

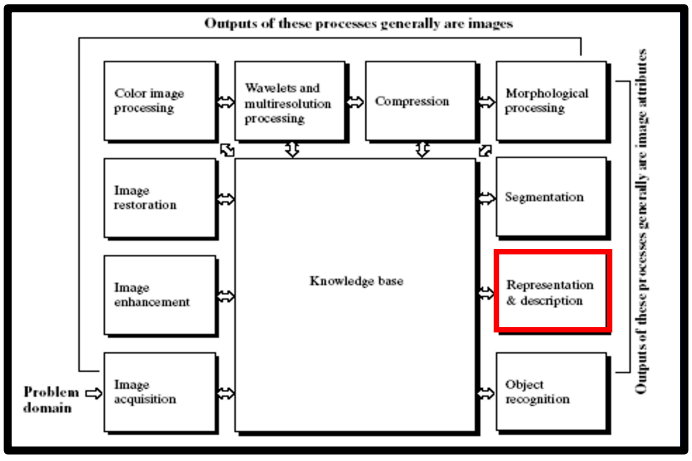
- Use **connected components** to identify candidate characters.
- Filter characters based on:
  - Height and width relative to plate size.
  - Aspect ratio constraints.
  - Area proportion relative to plate.
- Resize and normalize each segmented character to **32x32**, with dilation.



Stage 3: Feature Extraction & Matching

- Extract **HOG features** for both templates and segmented characters.
- Compare using **Chi-square distance** to find the best matching template.
- Apply **topology-based refinement** for ambiguous characters:
  - Ambiguous sets handled:
    - 8 ↔ B
    - C ↔ G
    - 5 ↔ 6
    - 1 ↔ L
  - Use **Euler number, connected components, number of holes, and aspect ratio** to resolve ambiguities.





Stage 4: Result Visualization and Saving

- Draw bounding boxes and predicted characters on plate images.
- Display processed plates (optional in Colab).
- Save results to a **CSV file** with columns: filename, recognized_text.

➔ Here is a sample of our outputs:

97WW755.png → 97WW755

07KE10361.png → 07KE10351

04WH3753.png → 04WH3753

4) Challenges Faced

During the implementation of the License Plate Recognition (LPR) project, we faced several challenges that made us experiment various techniques ; we found the following solutions:

1. **Segmentation of Characters**
    - Characters on plates are often **touching, skewed, or unevenly spaced**, making it difficult to isolate each character reliably.
    - We experimented with **morphological operations, connected components, and bounding box filtering** to ensure each character could be extracted cleanly.

2. **Character Confusions**
    - Certain characters share similar shapes, leading to misclassifications.
        - Examples: 8 vs B, C vs G, 5 vs 6, 1 vs L.
    - To address this, we implemented **topological checks** using:
        - **Connected Components (C)** – counting separate regions.
        - **Holes (H)** – detecting inner white regions inside characters.
        - **Euler Number (E)** – combining the number of connected components and holes to distinguish visually similar characters.

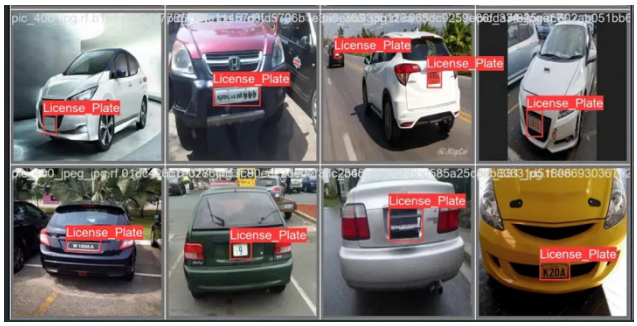
3. **Blurry or Low-Quality Images**
    - Some test plates were captured under poor lighting or motion blur.
    - We implemented an **is_blurry check** and applied **adaptive sharpening and CLAHE contrast enhancement** to improve recognition accuracy.

4. **Template Consistency**
    - Templates from different sources had varying sizes, stroke widths, and noise levels.
    - We designed a **preprocessing pipeline for templates** including resizing, padding, binarization, and stroke normalization to ensure a **consistent knowledge base** for HOG feature extraction.

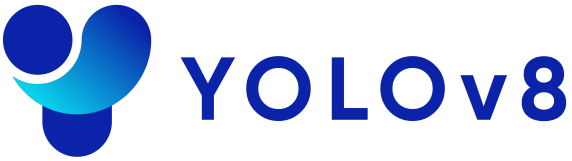
5) Future Improvements

While the current LPR system achieves accurate recognition on our dataset, several parts canbe enhanced in the future:

1. **Real-Time Recognition**
   - Our ultimate goal was to implement a **real-time LPR system** capable of processing live video streams from cameras.
   - We started experimenting with real-time pipelines, testing frame-by-frame detection and recognition.



2. **Deep Learning-Based Detection**
   - Initially, we intended to use **YOLOv8** for license plate detection and character segmentation, which would allow the system to detect plates directly from images or video with high speed and robustness.
   - Due to **course limitations we were not allowed to do it so we can improve what we have now and implement real time detection using YoloV8**



3. **Country Generalization**
   - Handling **plates from multiple countries**, **with many fonts**
   - Combining our HOG + topological features pipeline with a deep learning-based detector could create a hybrid system with both **high accuracy and speed**.