B222-E141030 - Sensor systems

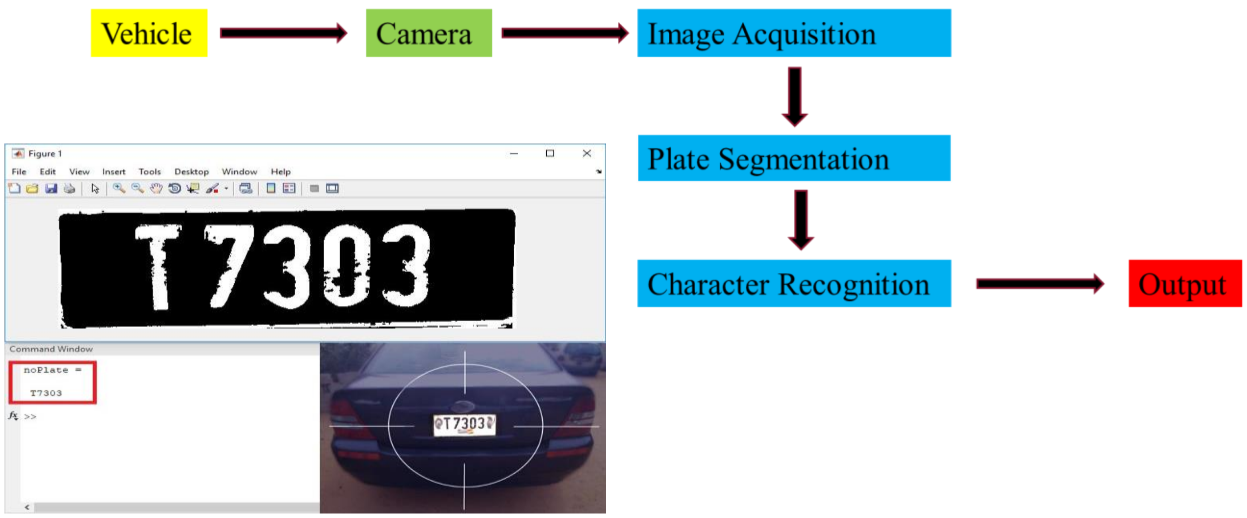
Lab 5 – Image Processing

- **Image Processing - Display and Simple Manipulation, CV classifiers.**
  Write the missing code in main1.mlx & main.mlx files in Task1 folder.

- ***Vehicle Number Plate Recognition System***

Traffic cameras capture images of numerous vehicles daily. Manually detecting the number plates of each vehicle can be a time-consuming process. Automatic number plate detection systems can overcome this problem by reducing human effort and processing time while increasing accuracy. This technology can quickly detect any car that is overspending or has violated traffic rules by recognizing its number plate. As a result, violators can be easily punished, and traffic accidents can be prevented.

1. How number plate detection works (Process Block Diagram)



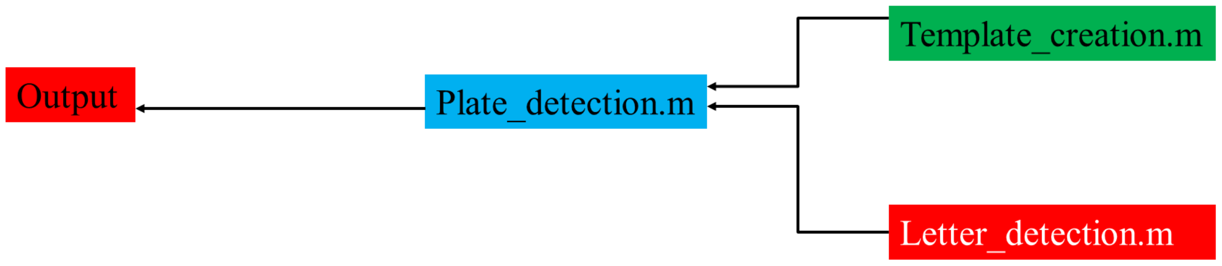
```
noPlate =

T7303

fx >>
```

The system's camera captures an image of a vehicle's license plate, which undergoes multiple algorithmic processes to convert the alpha-numeric characters in the image into a text format.

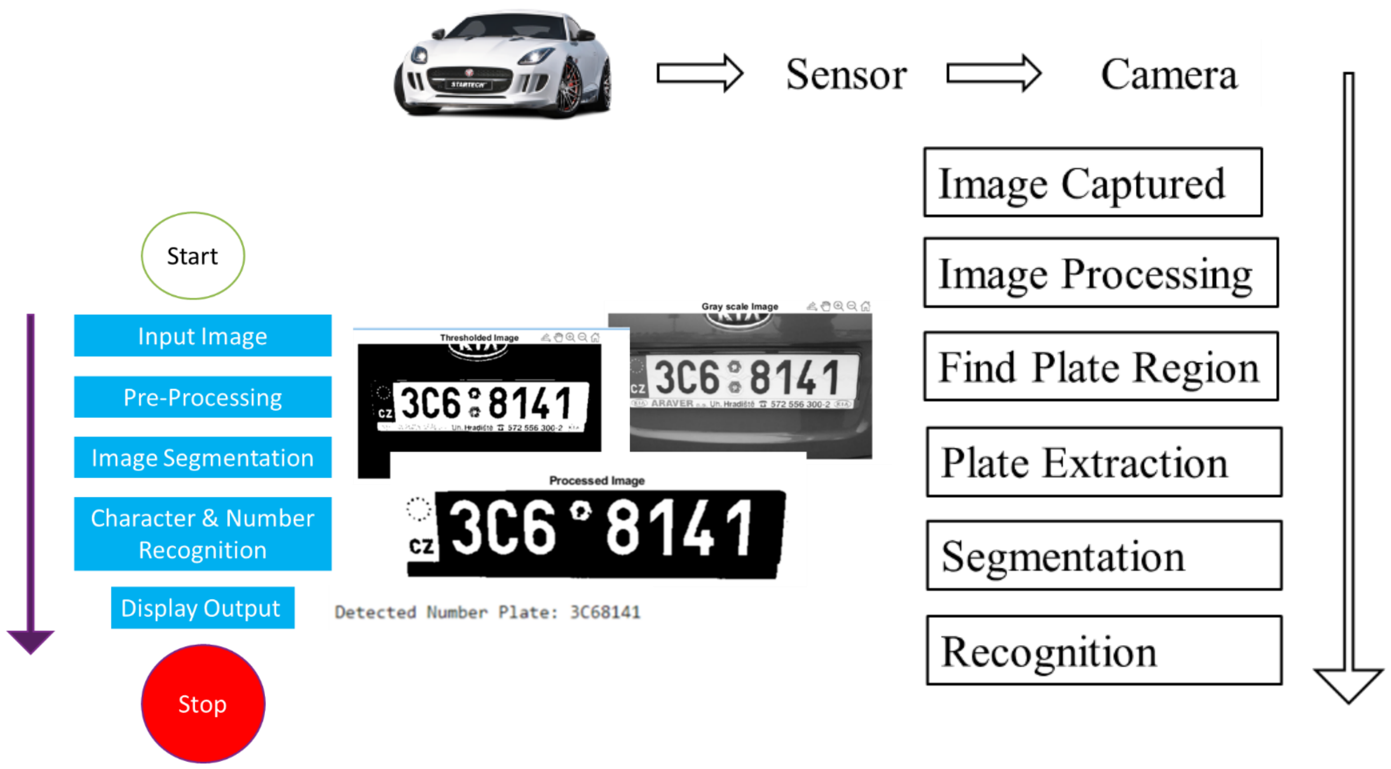
2. Programming code structure



This project consists of three programs, each saved as '.m' files:

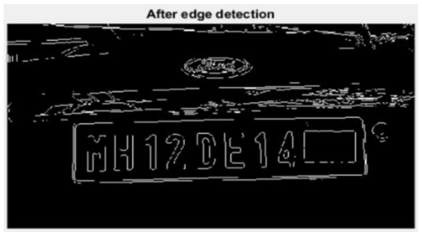
1. Template_creation.m: This program retrieves saved images of alphanumeric and creates a new template in MATLAB memory.
2. Letter_detection.m: This program reads characters from an input image and matches them with the corresponding alphanumeric with the highest match score.
3. Main.mlx: This program processes the image and utilizes the above two programs to detect the number plate.

Below you can see flow diagram and use case for this process.

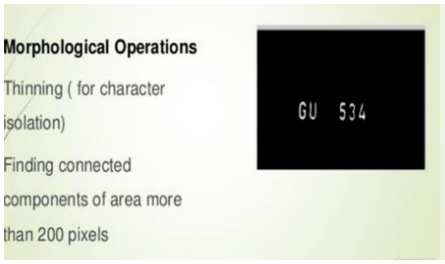


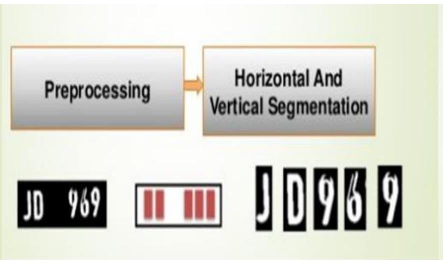
Load the image

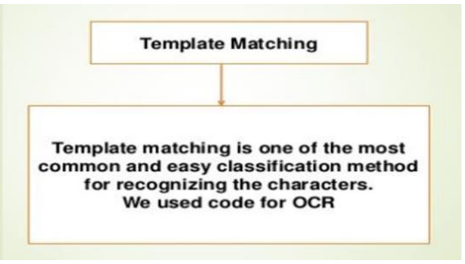
Convert the RGB to gray



Edge Enhancement


Morphological Operations
Thinning (for character isolation)
Finding connected components of area more than 200 pixels




Template Matching
Template matching is one of the most common and easy classification method for recognizing the characters.
We used code for OCR

Functions used:

- **imread()** – This command is used to open the image into the MATLAB from the target folder.
- **rgb2gray()** –This command is used to convert the RGB image into grayscale format.
- **imbinarize()** – This command is used to Binarize 2-D grayscale image or simply we can say it converts the image into black and white format.
- **edge()** – This command is used to detect the edges in the image, by using various methods like Roberts, Sobel, Prewitt and many others.
- **regionprops()** – This command is used to measure properties of image region.
- **numel()** – This command is used to calculate the number of array elements.
- **imcrop()** – This command is used to crop the image in the entered size.
- **bwareaopen()** – This command is used to remove small objects from binary image.

By using the above commands in the code, we are calling the input image and converting it into the grayscale. Then the grayscale is converted into the binary image, and the edge of the binary images is detected by the **Prewitt method.**

Tasks:

1- Read what you can find about **Morphological operations**, **OCR, and Prewitt method** and based on the code, discuss how these methods were used in this algorithm.

2- Run the code to process a plate that has another numeric shape for example https://www.europlates.eu/images/plates/syr/P00263.JPG
Discuss how you can solve this problem by processing plate images from different countries and different alphabets to recognize the plate numbers, try to modify the code to read the plate, discuss the limitations, and the solutions?

- ## *Multiple Face Detection System*

  It implements tracking multiple objects in real time using WebCam and **Kanade-Lucas-Tomasi (KLT) algorithm.**
  Automatically detects and tracks multiple faces in a webcam-acquired video stream.
    - ✓ Install the Webcam Support Package to run the code.

  It bring live images from any USB Video Class (UVC) Webcam into MATLAB.
    - ✓ Run detectAndTrackFaces.m from MATLAB.

Tasks:

1- Draw a diagram about the KLT algorithm, discuss limitations, and use cases.
2- Modify the code so it measures distance to the recognized face.

- ## *Car Detection*

Detect cars in a video by splitting the video into a sequence of images, apply image processing methods to separate the cars from the background by using Gaussian Mixture Model (GMM) as machine learning model.

The Gaussian Mixture Model (GMM) is represented by the following mathematical equation:

$$p(x) = \sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(x|\mu_k, \Sigma_k)$$

Where:

- $p(x)$ is the probability density function of the GMM.
- K is the number of Gaussian components in the mixture.
- π_k represents the weight or mixing coefficient of the kth component, satisfying $\sum_{k=1}^{K} \pi_k = 1$.
- $\mathcal{N}(x|\mu_k, \Sigma_k)$ is the Gaussian distribution of the kth component, defined by its mean μ_k and covariance matrix Σ_k.

Note: These formulas are defined in MATLAB's own function, and it performs operations according to parameters.

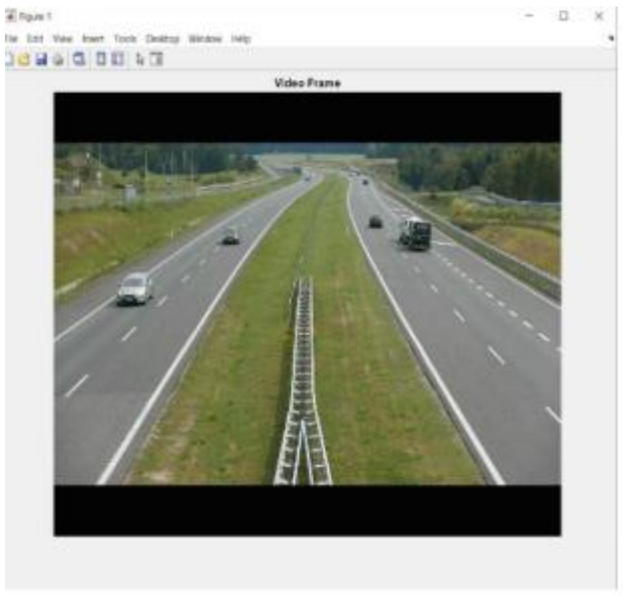
GMM can be employed as a background modeling technique in object detection systems. The main idea is to model the background of a scene using a GMM and then detect objects by identifying pixels or regions that deviate significantly from this learned background.

Initially, the GMM is trained on a set of background images or frames to learn the statistics of the background pixels. Each pixel is represented as a feature vector, typically containing color or intensity values. The GMM parameters, including the component weights, means, and covariance, are estimated using an expectation-maximization (EM) algorithm or similar techniques.

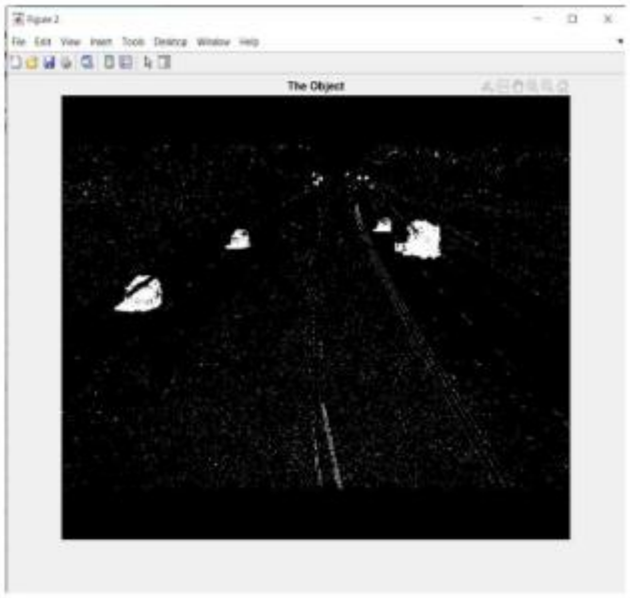
During object detection, a new frame is compared to the background model. Pixels or regions that exhibit a low likelihood under the GMM (i.e., they deviate significantly from the background) are considered as potential object candidates. These candidates can be further processed using additional techniques such as contour analysis, shape matching, or machine learning classifiers to refine the object detection results.

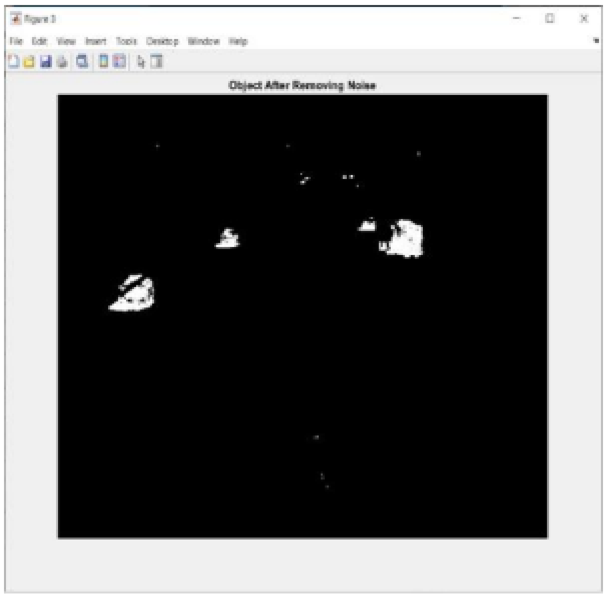
By utilizing GMM for background modeling, object detection systems can effectively distinguish foreground objects from the background, even in challenging scenarios with dynamic backgrounds, illumination changes, or moving cameras. GMM provides a probabilistic framework to model the complex distributions of the background and adapt to its variations over time, making it a valuable tool for robust object detection.

In order to address the issue of background noise interfering with the detected objects, we can employ morphological opening as a solution. When a vehicle is distant from the camera, the size of the noise and the size of the vehicle can become similar, leading to undesired results in the detection process. To overcome this limitation, we can utilize morphological opening to eliminate noise and fill gaps within the detected objects. These operations, collectively known as Morphological Operations to Remove Noise, help improve the accuracy of the object detection system.



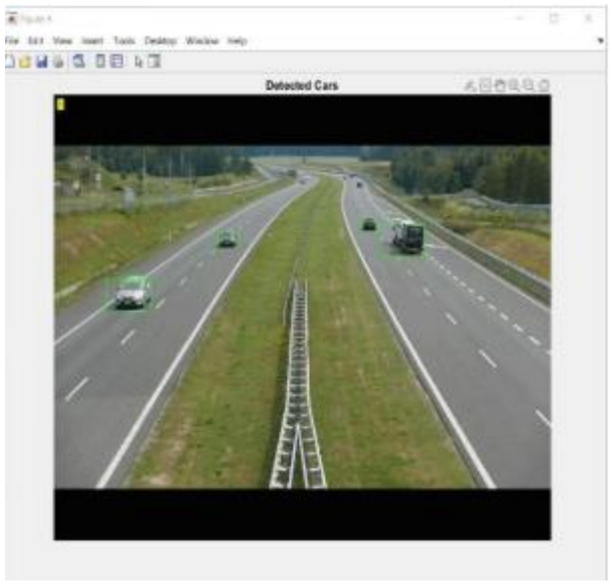
Video Frame (Fig1)





Object before noise removal (Fig2)

Object after noise removal (Fig2)



Detected Cars (Fig4)

Tasks:

1- Run the code, and read it, explain how the following processes were implemented:
   a. The location of the object.
   b. The counting of the vehicles in each frame.
   c. Suggest how we can measure the speed of the detected vehicles

REFERENCES

[1] Shengroung Gong, Chunping Liu, Yi Ji, Baojiang Zhong, Yonggang Li, Husheng Dong Advanced Image and Video Processing Using MATLAB, vol. 12, SPRINGER 2019, pp. 566 — 568.

[2] Kaewtrakulpong, P. and R. Bowden. An Improved Adaptive Background Mixture Model for Realtime Tracking with Shadow Detection. In Proc. 2nd European Workshop on Advanced Video Based Surveillance Systems, AVBS01, VIDEO BASED SURVEILLANCE SYSTEMS: Computer Vision and Distributed Processing (September 2001).

[3] Stauffer, C. and W.E.L. Grimson. Adaptive Background Mixture Models for Real-Time Tracking, Computer Vision and Pattern Recognition, IEEE Computer Society Conference on, Vol. 2 (06 August 1999), pp. 2246-252 Vol. 2.