|  |  |
| --- | --- |
| Vehicle logo recognition  Detecting logos and labelling them | Abstract  With Image Processing using MATLAB. This project takes the car as an input and detects the logo and labels it as an output.  Farah Aymen Mounir 194233  AI-Group 1 |

**Table of Contents**

[**1.** **Overview** 2](#_Toc102095605)

[**2.** **Approach** 3](#_Toc102095606)

[**3.** **Code Explanation** 5](#_Toc102095607)

[**4.** **Output** 12](#_Toc102095608)

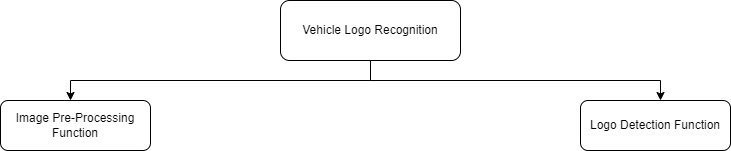
[**5.** **Images Used as Dataset** 15](#_Toc102095609)

# **Overview**

One of the necessary applications needed these days is image detection and recognition in general. With the development of image recognition, it has become one of the most used applications in the industry. For example, license plate recognition, self-driving, etc… This project aims to detect car logos where it does some pre-processing on the image to enhance it, then segment the logo from it using a specific technique, and finally recognize which logo it is by the help of a predefined data set. The whole project is composed of two functions where the first one cleans the image and binarizes it, and the other takes the image and other values as a parameter and returns the segmented logo with its label.

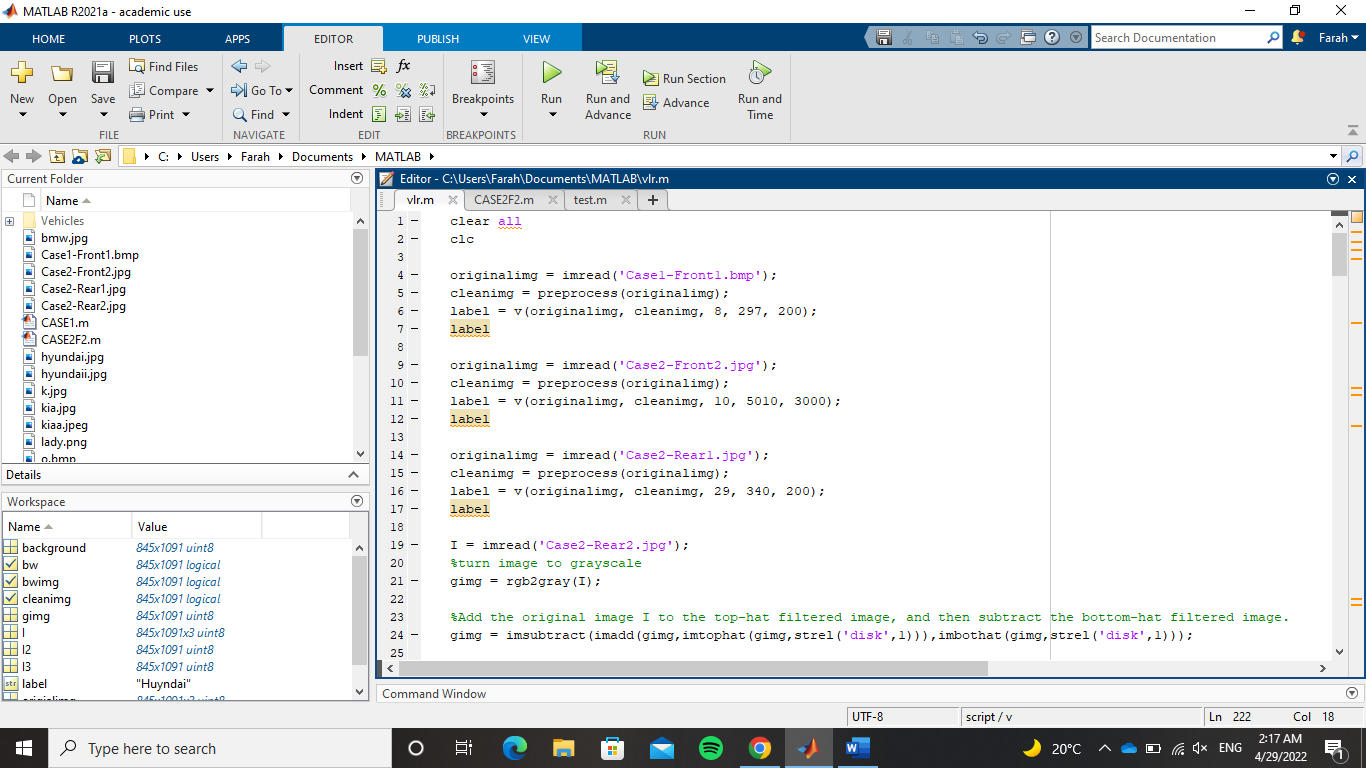
# **Approach**

The concept of this code to find generalized functions as much as possible to fit all test cases available. The project is composed of two main functions: pre-processing function, and recognition function.

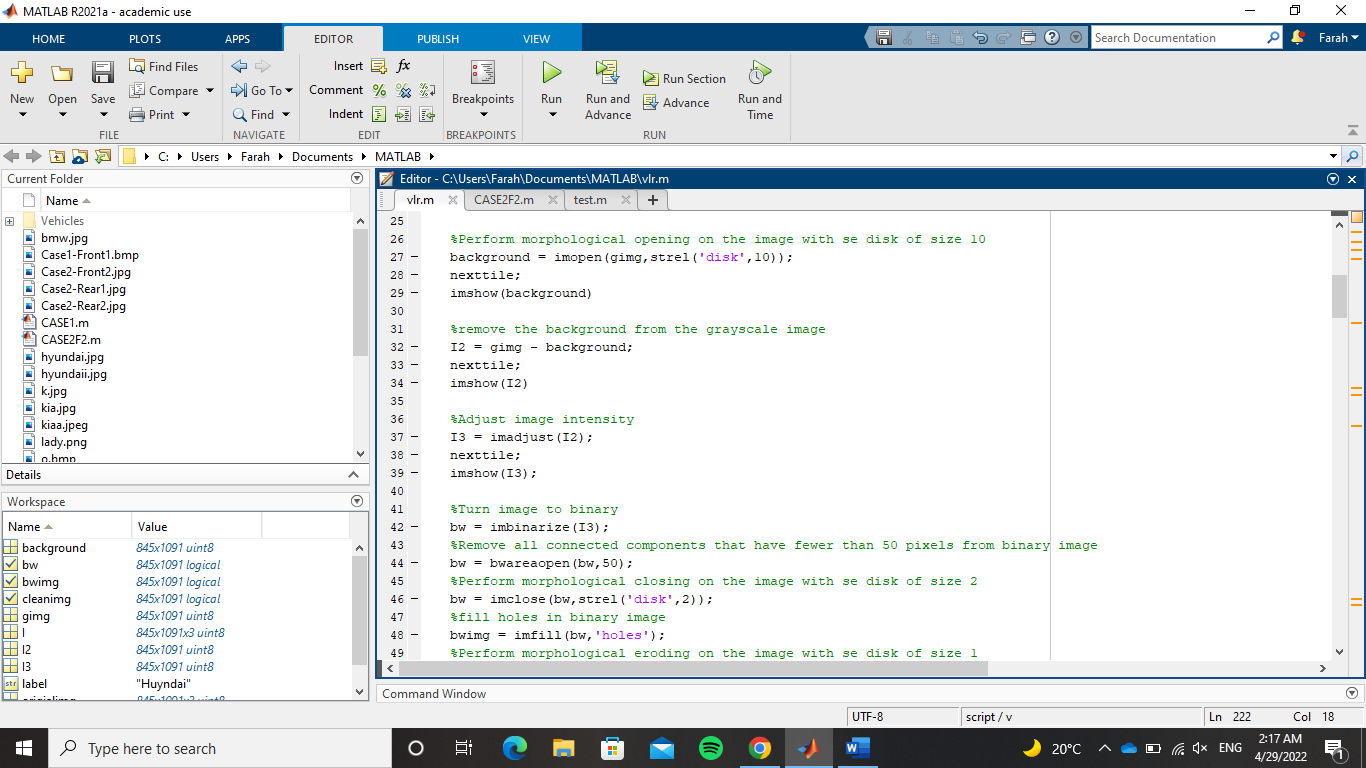


The first function’s parameters are only one parameter which is the image itself that needs to be processed and cleaned. The function takes the image to clean it so it could be ready for detecting the logo from it. First, the function converts the image into its grey scale the binarizes it so it can perform all the operations on it. Second, a structuring element is defined so it can be used in some morphological operations done on the image. This structural element is a disk of size 1. Third, it first adds the greyscale of the image to a top hat filtered version of the greyscale image, and then subtracts the bottom-hat filtered version of the greyscale image from the result of the addition. The top hat in general performs morphological opening on the image and subtracts the result from the original image. While the bottom-hat filtering performs morphological closing on the image and subtracts the original image from the result. Performing these operations of the greyscale image enhanced it so it can now be clear for segmenting and detecting the logo. The Second function takes parameters of the original image, the enhanced image, three parameters acquired by trial and error for further purposes that will be discussed later. The idea from the function is to sort all connected components in the image according to their areas so we can select the logo from all of those regions. First, the function labels all connected parts, then groups them according to their areas in one array. Then, it sorts the areas in descending order, and loops over them to remove all areas bigger than the logo. This is where the third parameter comes in; as it is the number of regions bigger than the logo. It is used in the for loop that loops on the array of areas. Then, the array of areas gets resorted in an ascending order and perform the same for loop to remove all small regions smaller than the logo. This is where the fourth parameter comes in; as it is the number of regions that has smaller areas than the logo. That is used in the for loop. Now, supposedly the only remaining part is the logo, so the function measures the coordinates of this remaining part to crop it from the original image, and now the logo is segmented. Using Fourier transform, the function is going to deduce the features of the logo to compare it with features of each image in a dataset of logos, then it chooses the closest of them to identify with it. In the end, the function returns the label of the logo. Notice that, the fourth testcase image needed specific pre-processing done but the rest could fit into the pre-processing function.

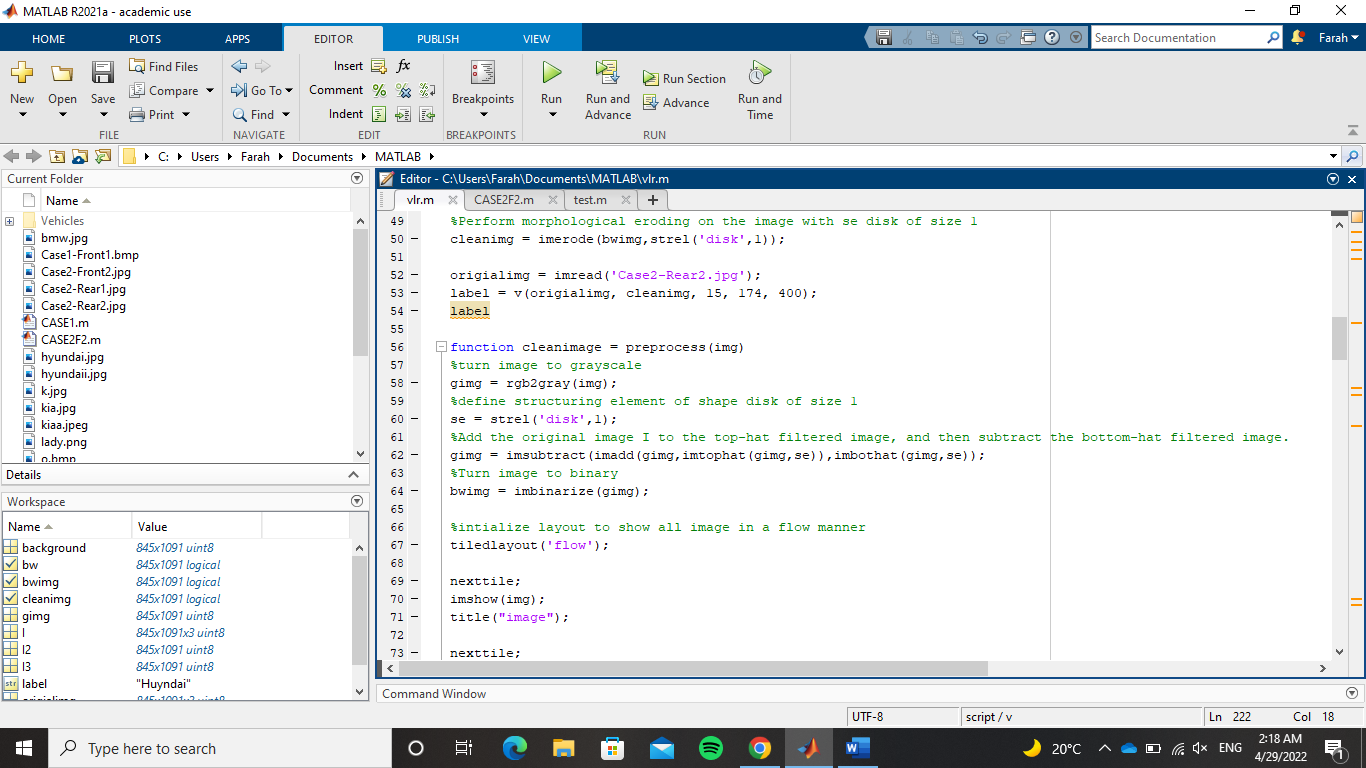
# **Code Explanation**



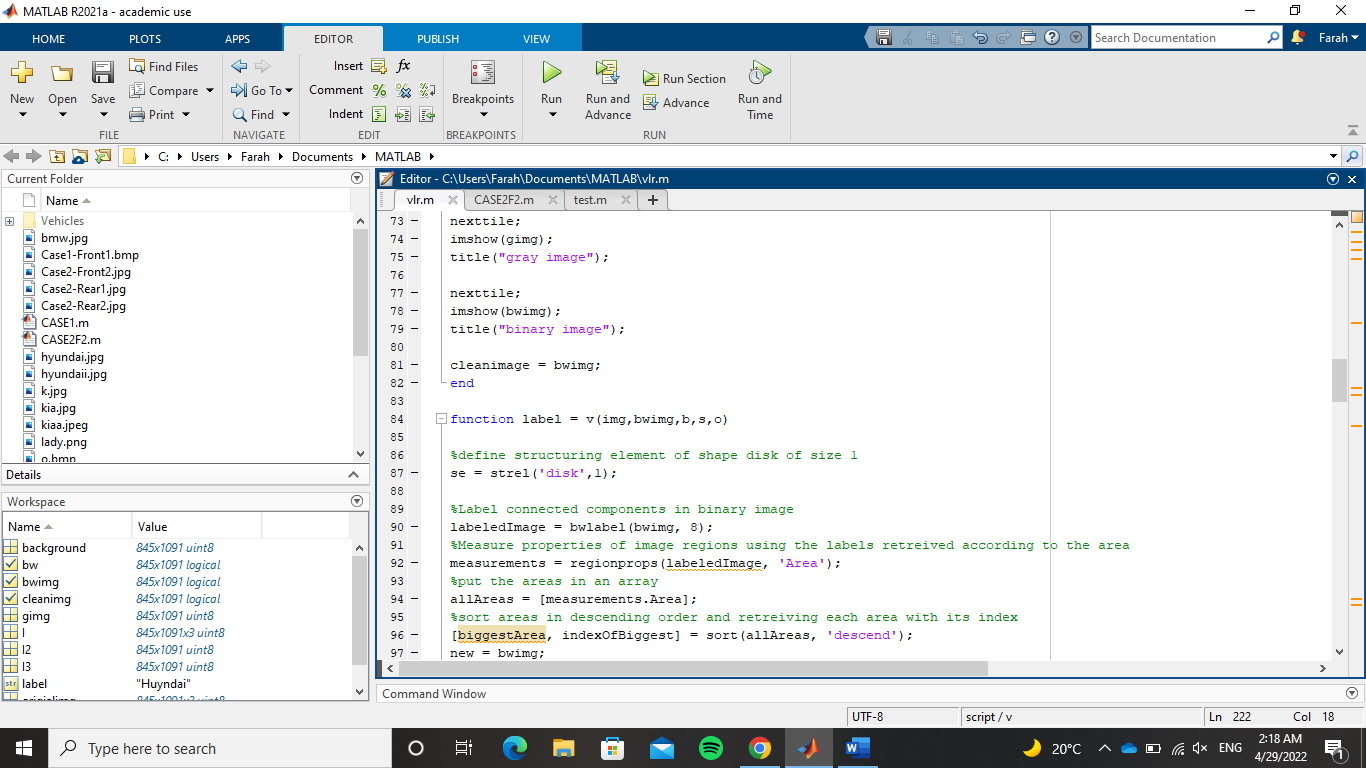
Calling the pre-processing function and the recognition function for each image giving the appropriate parameters.



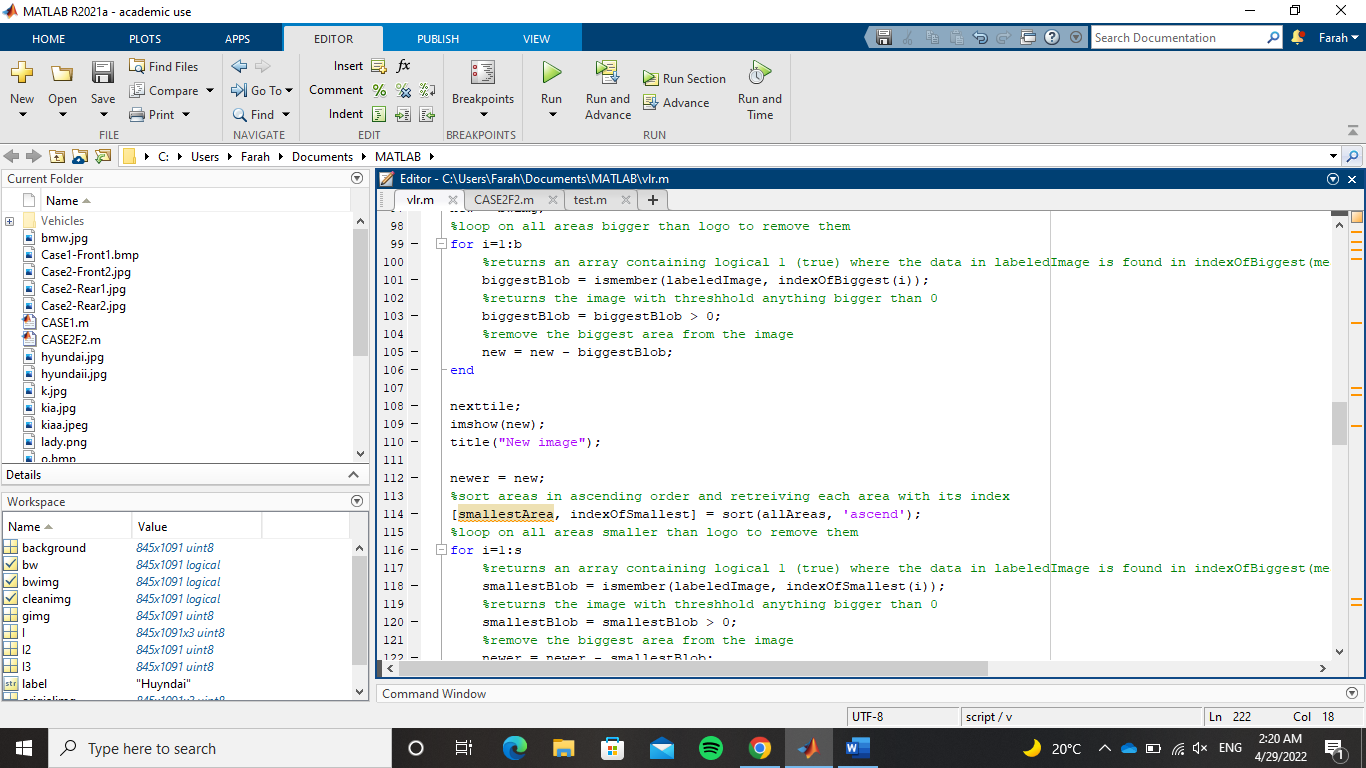
Reading the fourth test case, it needed more pre-processing to enhance it because the image had a lot of issues one of them is a faint reflection of the sun on the car hood that needed to be adjusted. First, performing a morphological opening on the image with structuring element disk of size 10, this will give us a kind of a blurred image (background) that will be subtracted from the image to give an image showing something like the edges of objects in the image. Second, adjusting the intensity in the new image to give result of brighter more visible lines. Third, turn the image into binary. Fourth, remove any components less than 50 pixels (this number is obtained by trial and error). Fifth, perform close operation to keep the elements intact. Sixth, fill any holes to make the image even more intact and avoid any data loss.



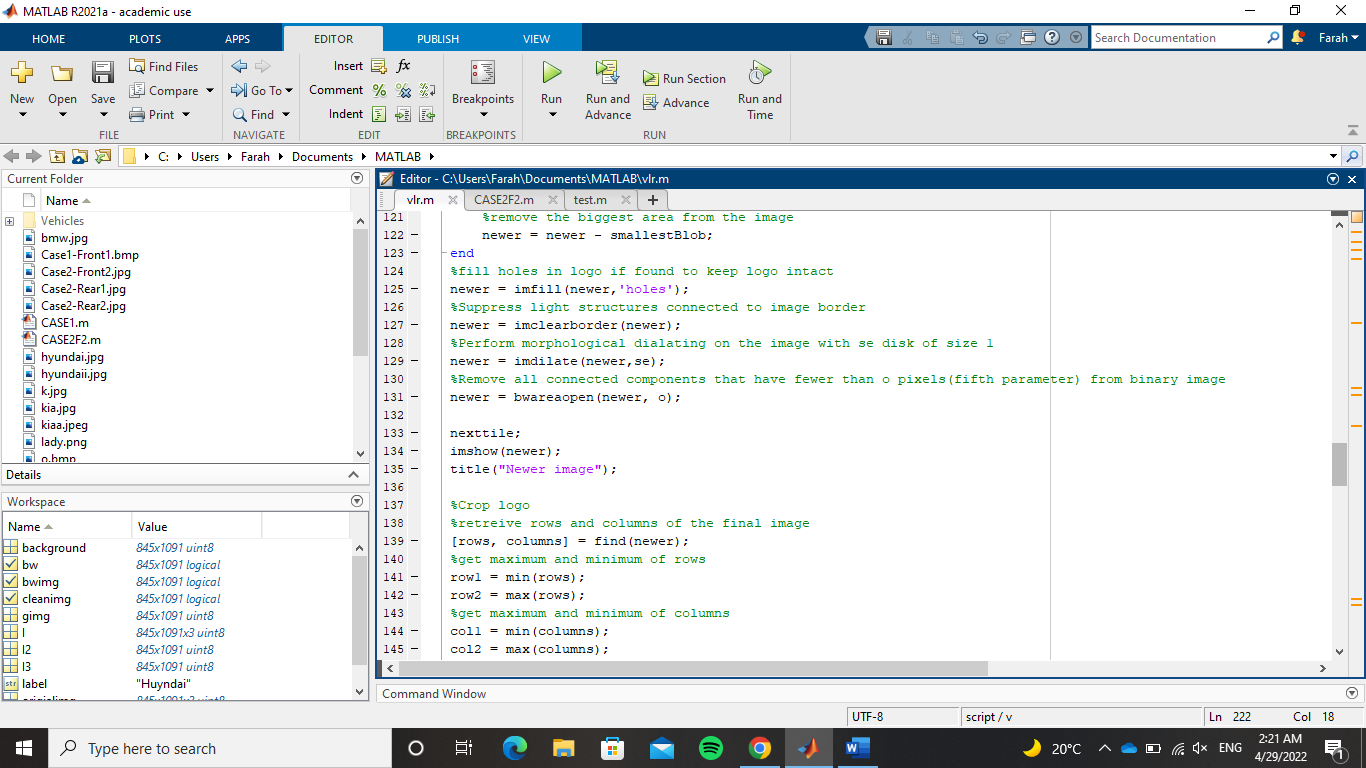
Seventh, perform erode operation to disconnect some components that have been caught together from the closing and filling. Finally, send the image to the recognition function (v) along with its other parameters. Regarding the pre-process function, it takes the image as a parameter to enhance it and returns it back. First, it turns the image to grayscale. Second, it subtracts the bottom hat filtered version of the image from the addition of the original image and the top hat filtered version of the image. With this operation, the image is much enhanced from any external factors that affected the image. Finally, it turns the image to binary.



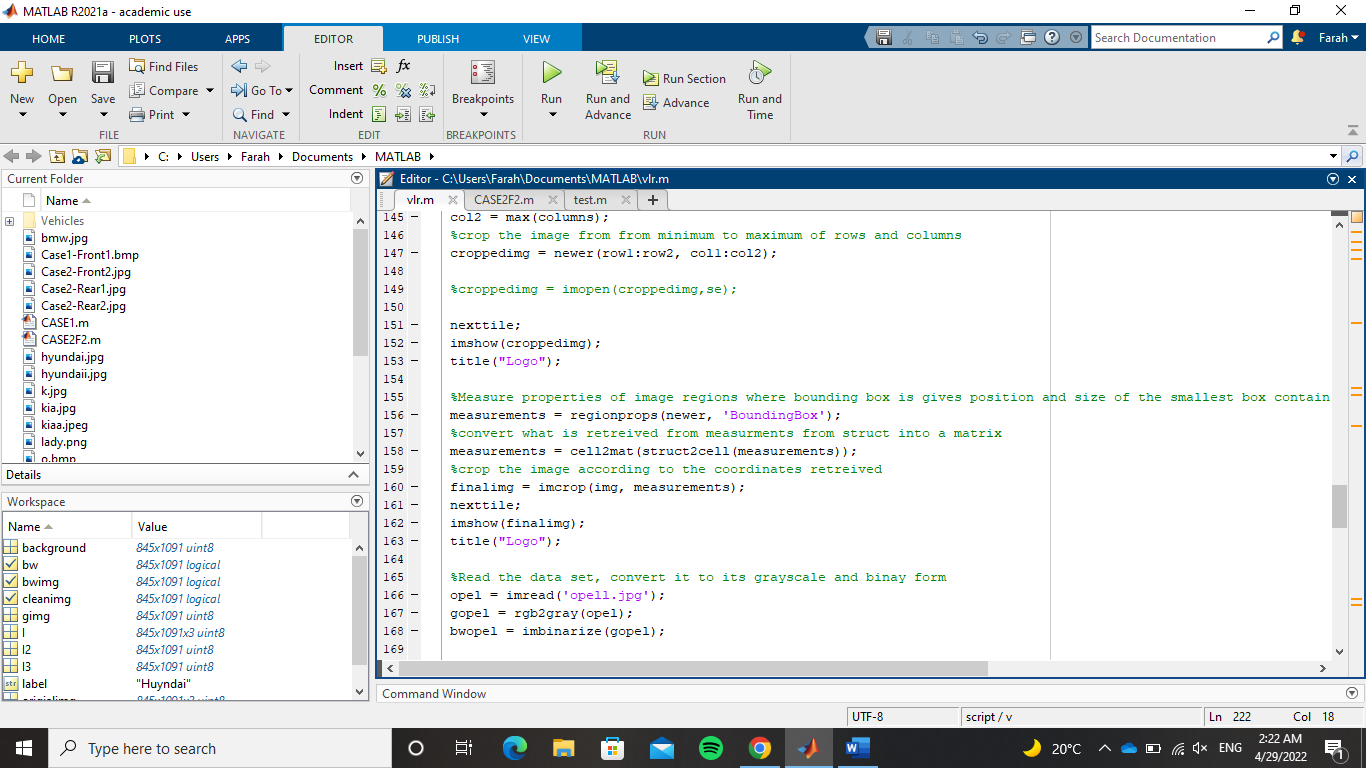
Regarding the recognition function (v), it takes five parameters: the original image, the processed image after getting cleaned from the previous function, a parameter that will be used in a for loop to remove all regions bigger than the logo, a parameter that will be used in a for loop to remove all regions smaller than the logo, and number of pixels that act as a threshold to remove any component that has pixels less than that parameter. First, a structuring element is defined to be used later in multiple morphological operations. Second, it uses the function bwlabel() to retrieve the connected components in the image. Third, using regionprops() it will measure the properties of the regions in the image based on their areas, then it will put the areas in an array called measurements. Fourth, it will sort this array of areas in descending order.



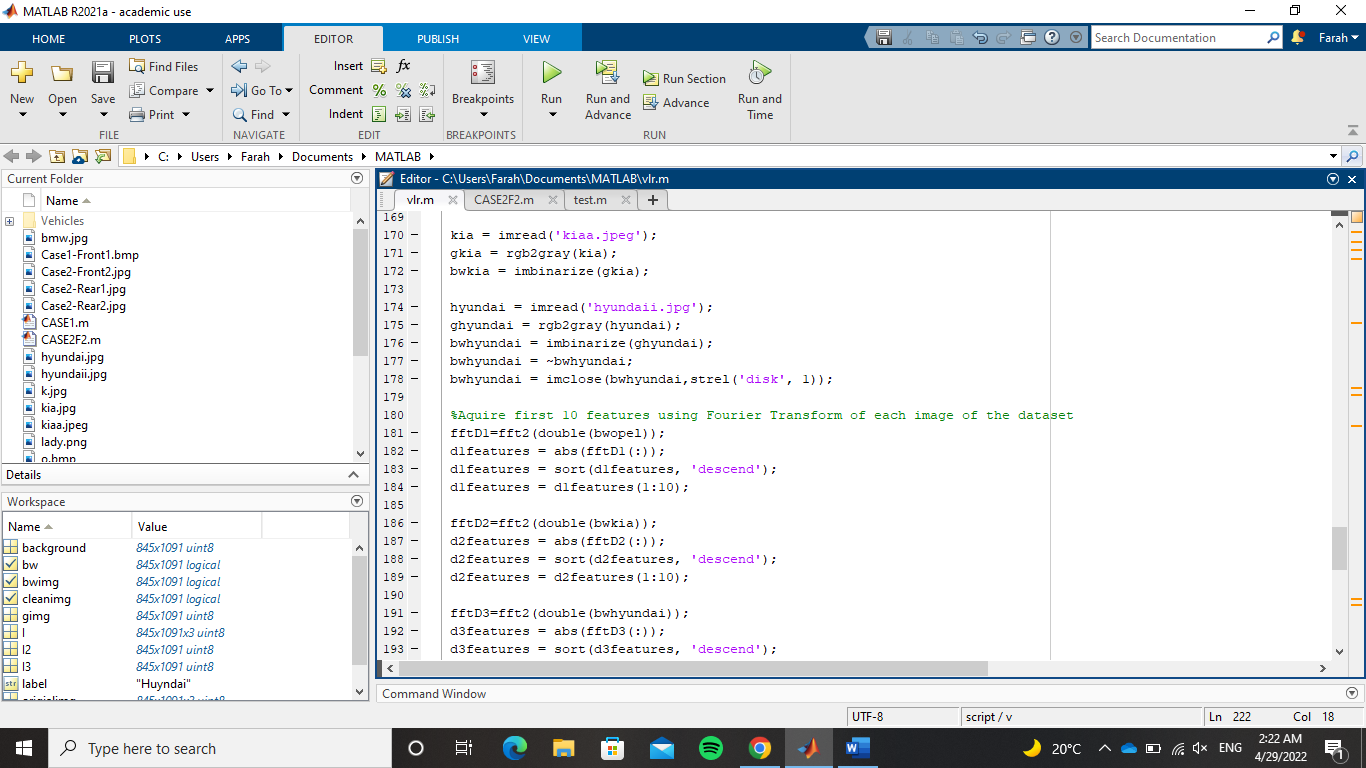
Fifth, it will loop on that array of areas using the third parameter as the limit of the interval. It will first determine the biggest area in the array that hasn’t been removed yet from the image using ismember (). This function will check if this region was found in the image. Then, it returns the image with anything bigger than 0. Sixth, it removes this big area from the image and so on until the for loop ends. Same thing goes for the other for loop. The only difference is that it re-sorts the areas in ascending order and loops using the fourth parameter as interval till it removes all regions smaller than the logo.



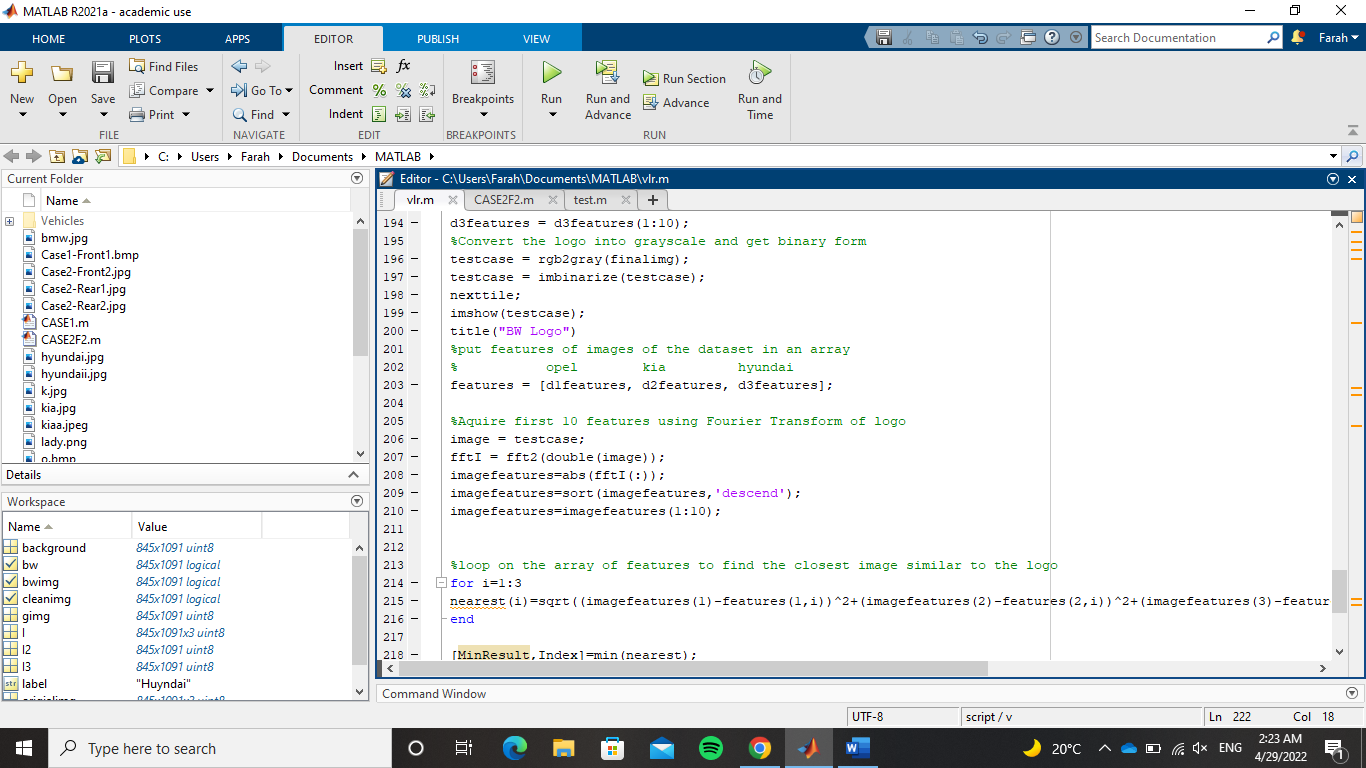
Then, it fills holes to keep the logo intact because the logo might have not been one piece from the beginning so any disconnected parts may have been removed while removing the other regions from the image. Then, it clears elements at the boarder of the image, to make sure nothing is left. They may have the same area as the logo so that’s why they weren’t removed in the for loops. Next, it removes any components that has pixels less than the fifth parameter. Now, the logo stands alone in the image. So, we crop it. This is achieved by getting the rows and columns of the image and asking the function to start at the most minimum row it first encounters and stops at the last row that has a value and not equal to 0. Same goes for the columns.



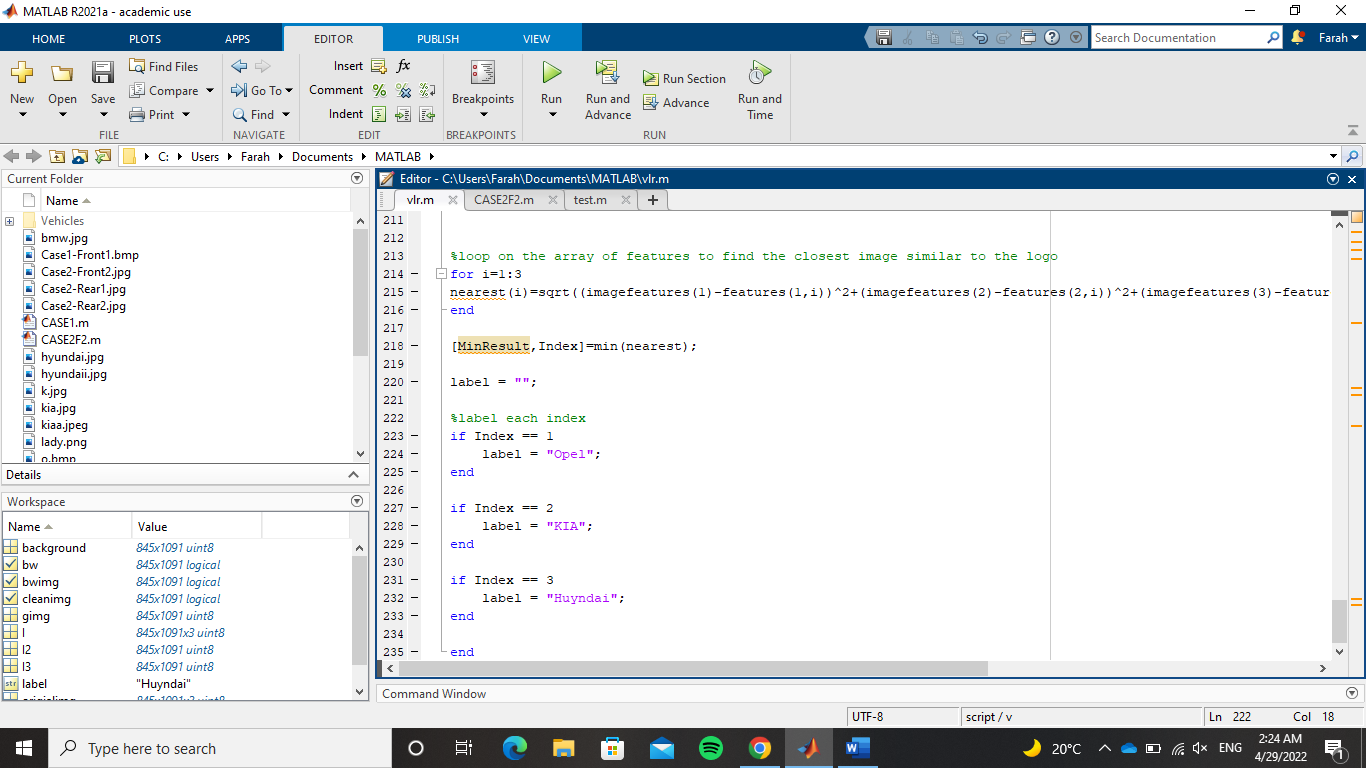
Now that only the logo exists in the image, its coordinates in the image can be easily retrieved using regionprops() to measure the properties of the image. Since that the function returns a struct and the crop function takes an array as an argument, first we will need to convert the stuct into a matrix which is achieved using the function cell2mat(). Next, it will crop the image using the function imcrop(). Now that the segmentation stage is done, the recognition phase begins. First, the function defines a set of images to test upon them which of them is the closest to the selected image, and transform them into greyscale then binarize them.



Using Fourier Transform, we gather the features of each image, order them in descending order, and select the first 10.

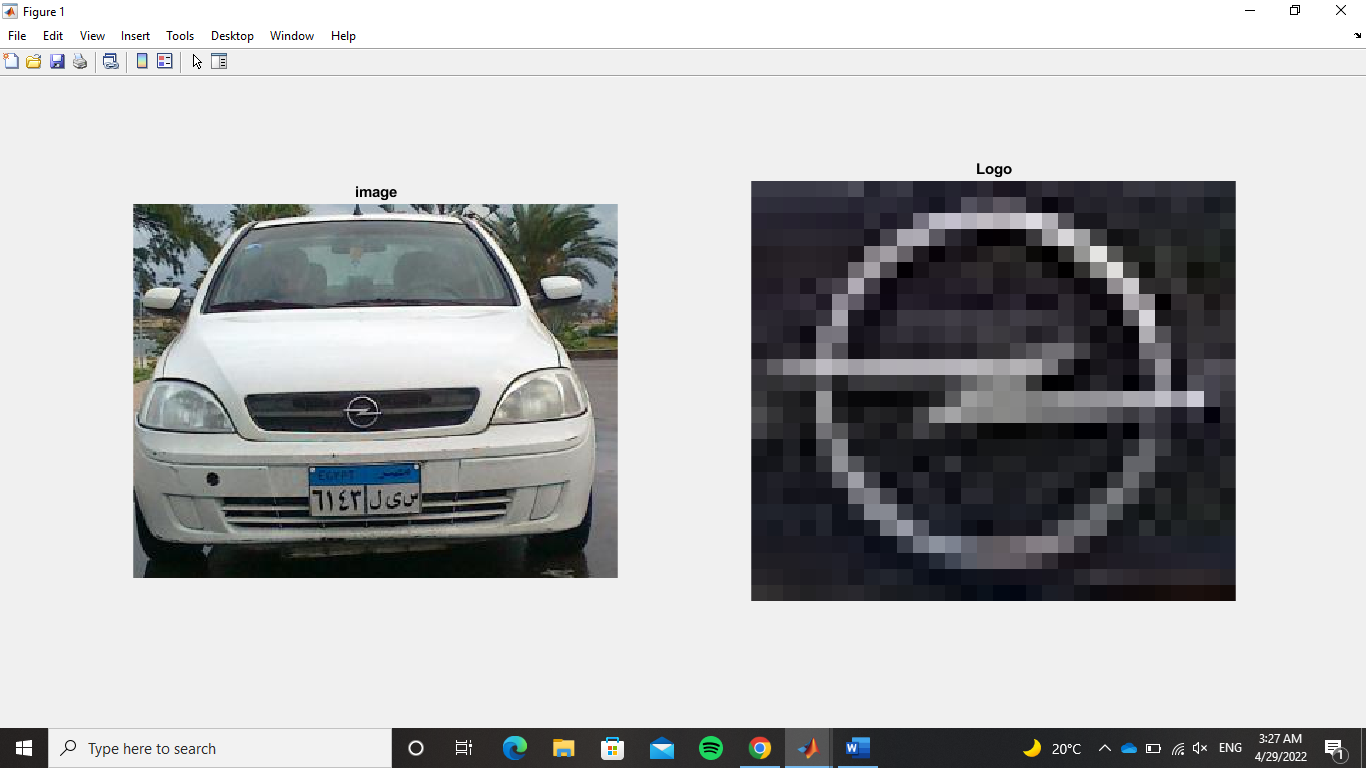


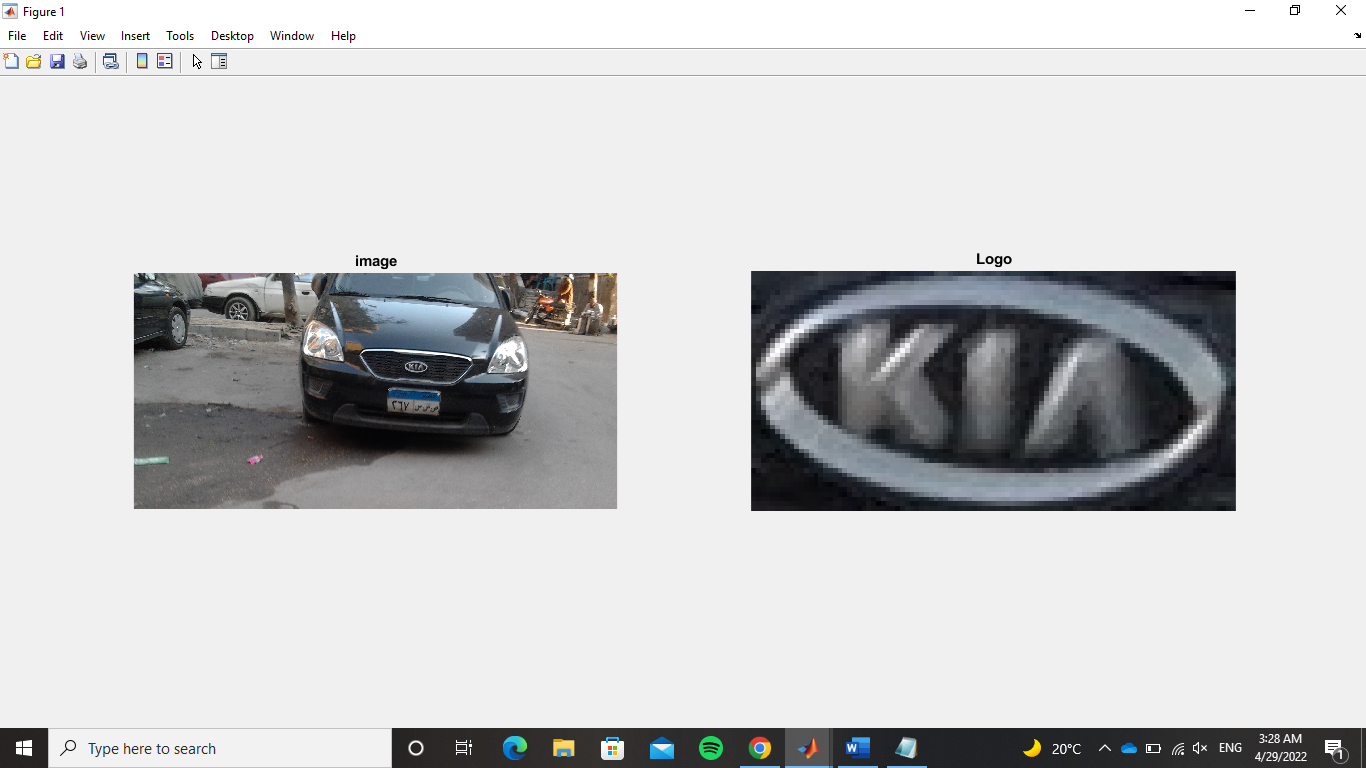
Then, we add those features in an array. After this, we get the features of the image we would like to know its logo. In a for loop, we compare the features of each image with the image of the logo.

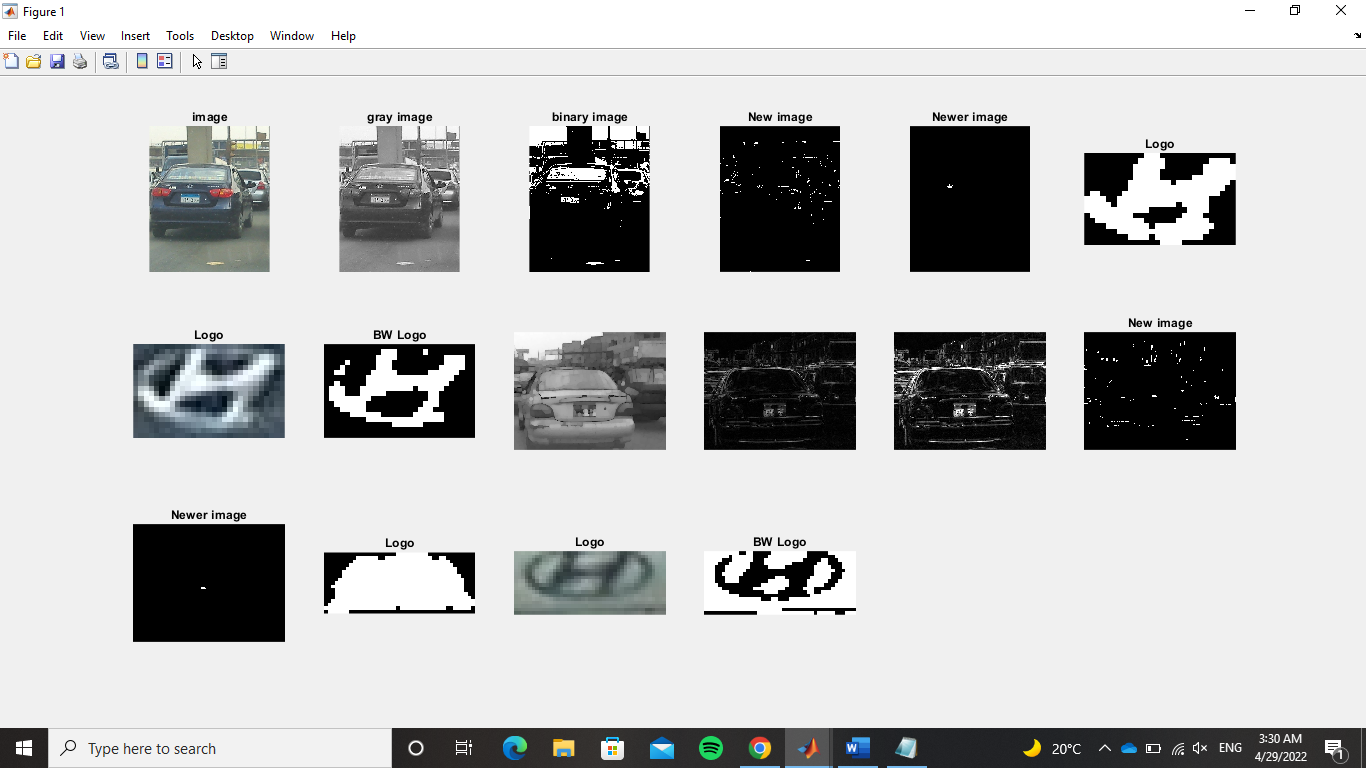


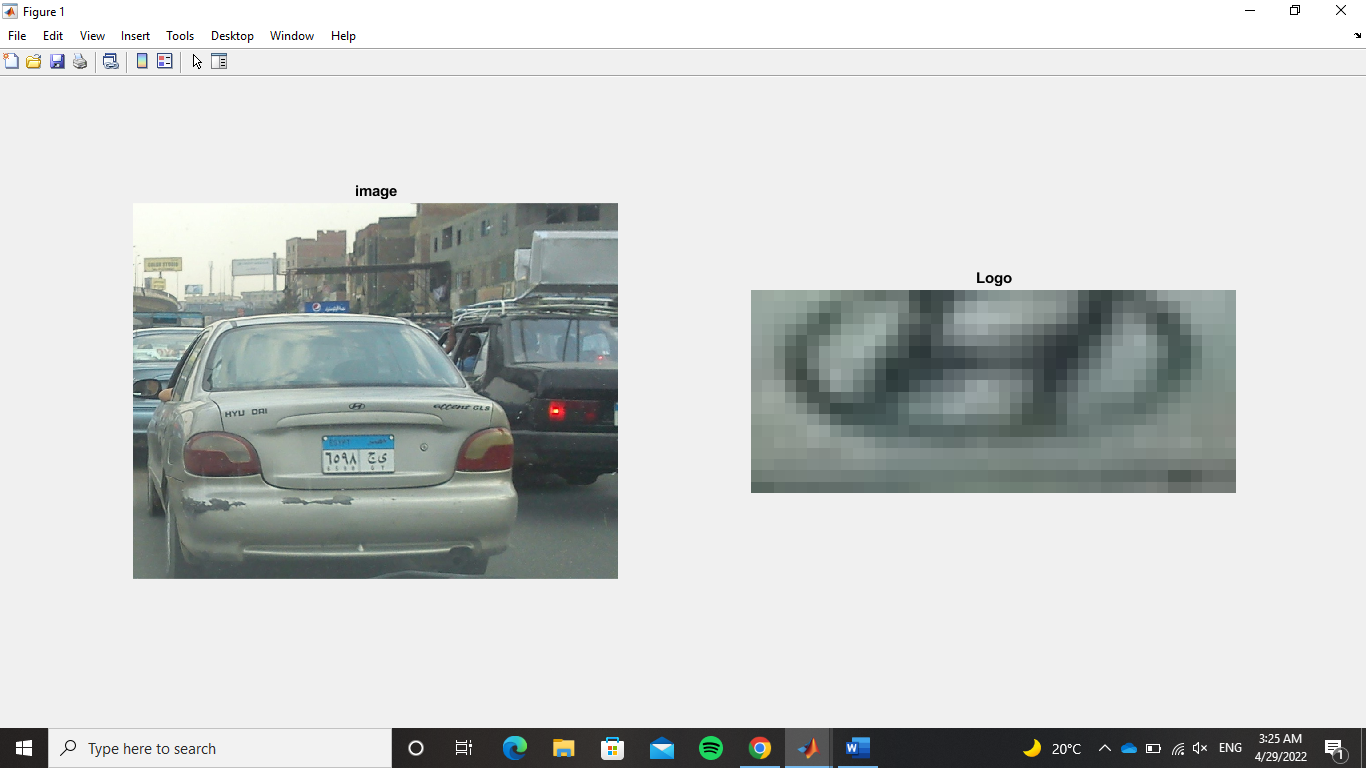
We choose the smallest Value as it indicates that it is the closest in similarity. Using if conditions we determine that each index indicates a certain logo. Finally, the function returns this label.

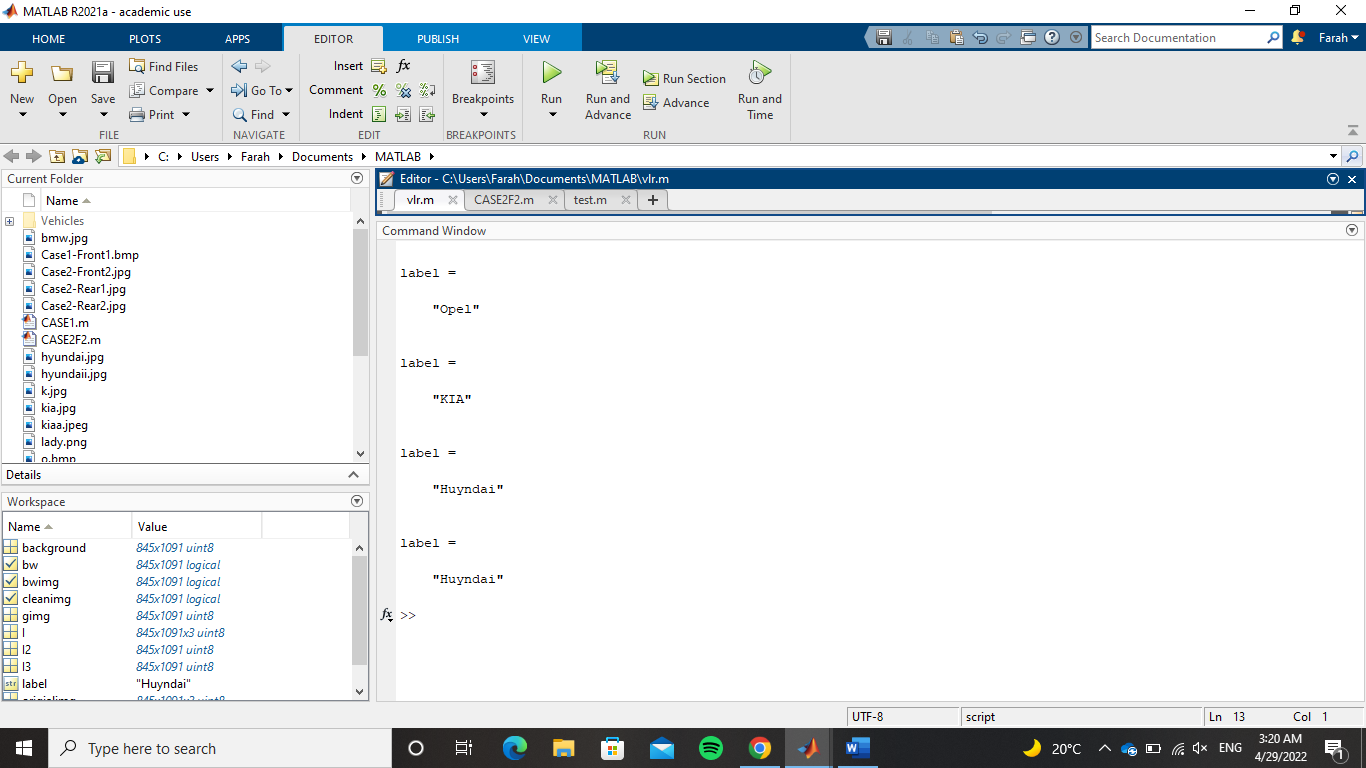
# **Output**











The first label is for the first test case, the second label is for the second test case, the third label is for the third test case, and the fourth label is for the fourth test case

# **Images Used as Dataset**

****

****

