**DETECTING THE COUNT OF CARS USING**

**IMAGE PROCESSING**

GADUPUTI ABHISHEK SAI RAM

RONANKI BHANU SAMPATH

Y. VENKATA SAI KRISHNA

**OBJECTIVE:**

To obtain the count of cars, an application of image processing using MATLAB.

**THEORY:**

An image is represented by its dimensions (height and width) based on the number of pixels. For example, if the dimensions of an image are h x w, the total number of pixels in the image is (h.w).

So, the pixel is a point on an image that takes on a specific shade, opacity or color. It is usually represented in one of the following:

* Grayscale - A pixel is an integer with a value between 0 to 255 (0 - complete black & 255 - complete white).
* RGB - A pixel is made up of 3 integers between 0 to 255 (the integers represent the intensity of red, green, and blue).
* RGBA - It is an extension of RGB with an added alpha field, representing the image’s opacity.

Image processing requires fixed sequences of operations that are performed at each pixel of an image. The image processor performs the first sequence of operations on the image, pixel by pixel. Once this is fully done, it will begin to perform the second operation, and so on. The output value of these operations can be computed at any pixel of the image.

The image processing system usually treats all images as 2D signals by applying certain predetermined signal processing methods.

We will be using MATLAB software to implement the project.

**METHODOLOGY:**

The steps performed to get the final count is as follows,

1. **Load the image** using the “imread” function.
2. Convert the **image to grayscale** using the rgb2gray function.

Purpose:

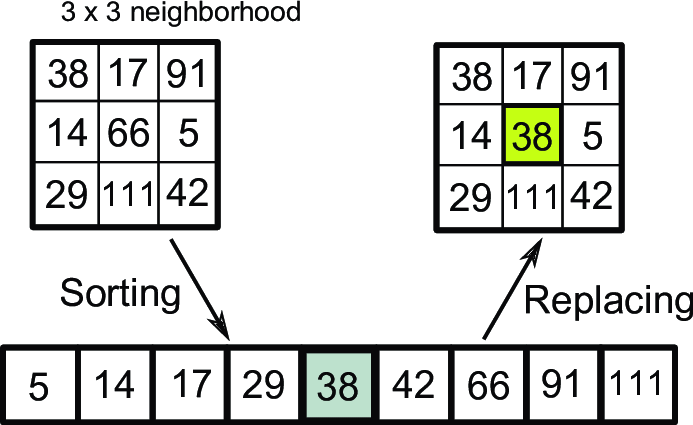
It helps in simplifying algorithms and as well eliminates the complexities related to computational requirements. For example, color information doesn't help us identify important edges or other features, complexity, speed, etc.

1. Apply a **median filter to reduce noise** using the medfilt2 function.

About:

The median filter is a non-linear digital filtering technique used to remove noise from an image. By moving through the image pixel by pixel, replacing each value with the median value of neighboring pixels.

Example of how it works:



1. Use the edge function **to detect edges** in the image.

About:

It’s a multi-stage algorithm used to detect a wide range of edges in an image. Among the edge detection methods developed so far, the Canny edge detection algorithm is one of the most strictly defined methods that provide good and reliable detection.

Gradient: A directional change in the intensity or color of an image.

The Canny method applies two thresholds to the gradient: a high threshold for an edge with low sensitivity and a low threshold for an edge with high sensitivity. Edge starts with the low sensitivity result and then grows it to include connected edge pixels from the high sensitivity result. This helps fill in gaps in the detected edges.

1. Apply a Hough transform using the Hough function **to detect lines in the image.**
2. Use the “hough peaks” function is used **to find the most prominent lines**.
3. Use the “hough lines” function is used **to extract the endpoints of the lines**.
4. **Use the slope of the lines** to determine if they correspond to a car.
5. **Count** the number of cars detected.

**CODE:**

% To load the image

img = imread('C:\Users\sam\Desktop\1140-cars-on-highway.jpg');

% Convert to grayscale

gray = rgb2gray(img);

% Apply median filter

gray = medfilt2(gray);

% Detect edges

edges = edge(gray, 'Canny');

% Apply Hough transform

[H,theta,rho] = hough(edges);

% Find peaks

peaks = houghpeaks(H, 10, 'threshold', ceil(0.3\*max(H(:))));

% Extract lines

lines = houghlines(edges, theta, rho, peaks, 'FillGap', 20, 'MinLength', 40);

% Calculate slope of each line

slopes = zeros(length(lines), 1);

for k = 1:length(lines)

endpoints = [lines(k).point1; lines(k).point2];

slope = (endpoints(2,2) - endpoints(1,2)) / (endpoints(2,1) - endpoints(1,1));

slopes(k) = slope;

end

% Determine if the line corresponds to a car

cars = 0;

for k = 1:length(lines)

if abs(slopes(k)) > 0.5 && abs(slopes(k)) < 2

cars = cars + 1;

end

end

% Display result

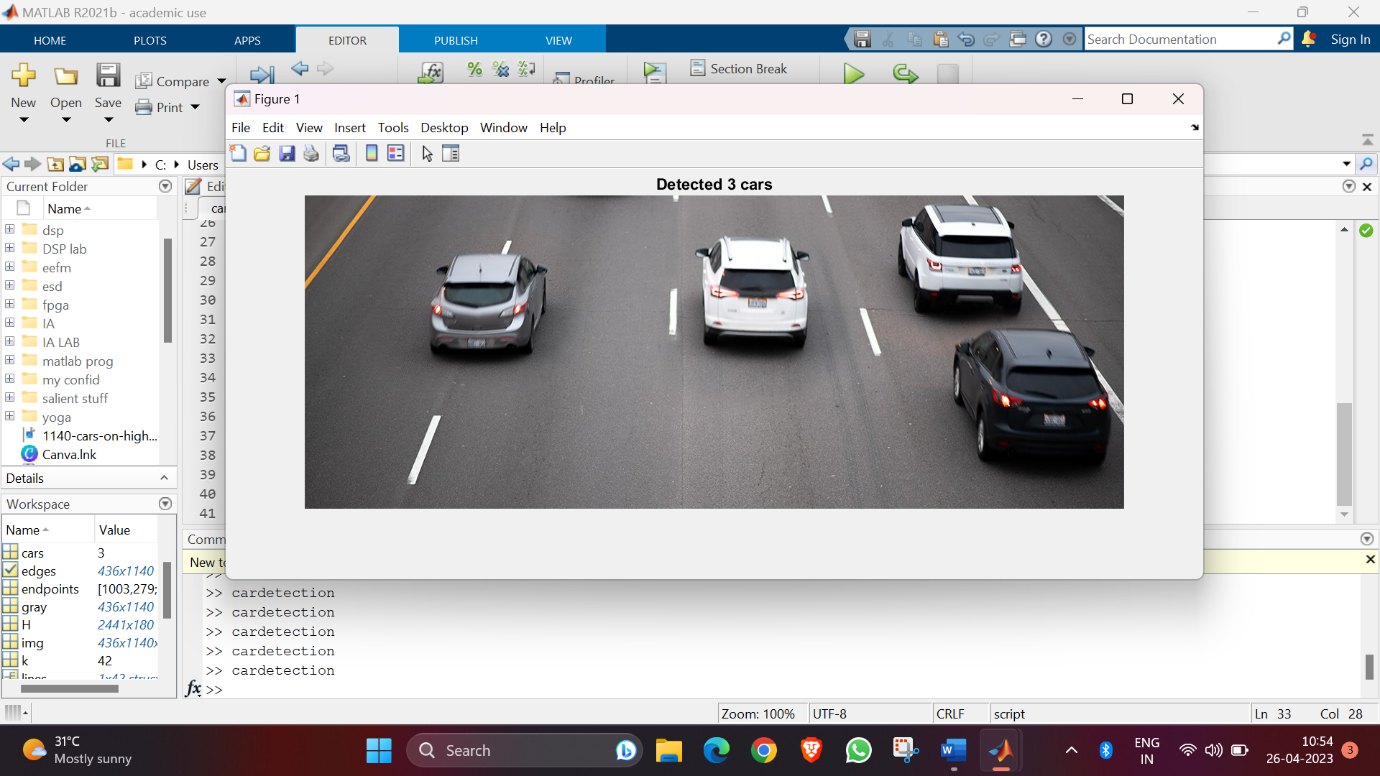
imshow(img);

title(sprintf('Detected %d cars', cars));

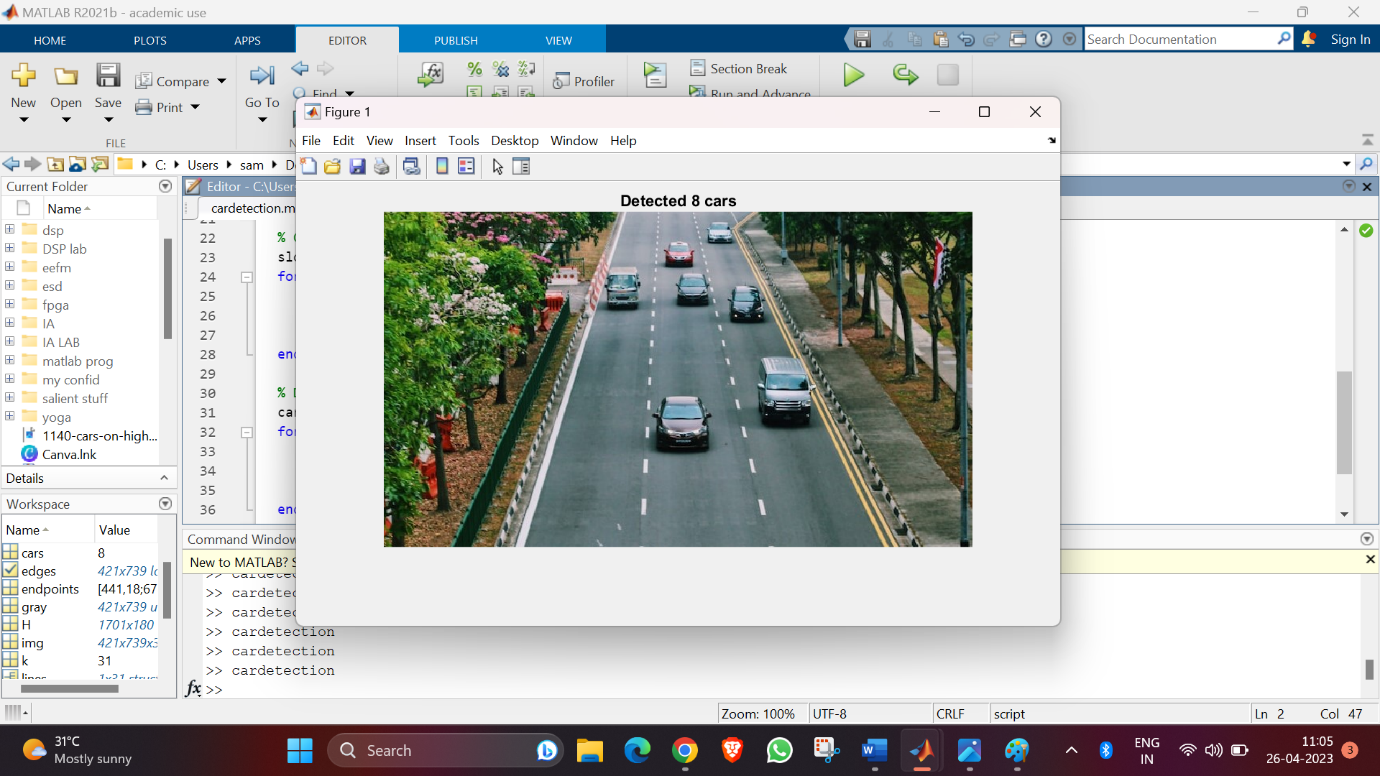
**RESULTS:**

3 sample outputs are shown below to appreciate the insight into this technique’s reliability,

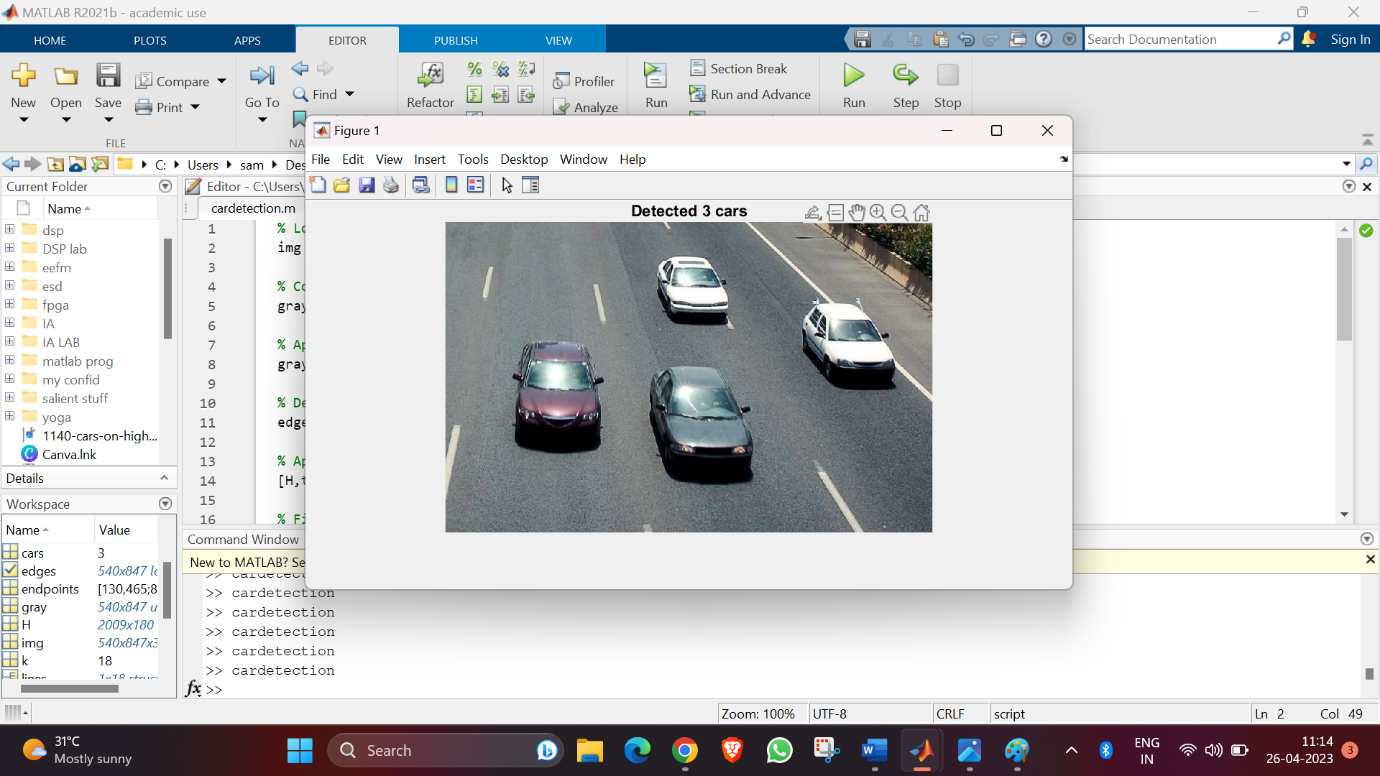
Sample1:



Sample2:



Sample3:



**CONCLUSION:**

The count of cars, an application of image processing has been successfully carried out.

We have to keep in mind that the parameters of the Hough transform and the edge detection can be adjusted for different images and are the most reliable means for carrying out simple operations of this kind where an accuracy of 100% isn’t a major concern

Also, the small deviation/error in the output is caused as the code assumes that the cars are roughly aligned with the image axis. For more complex images, more advanced techniques may be required.

**REFERENCES:**

1. <https://www.geeksforgeeks.org/digital-image-processing-algorithms-using-matlab/>

2. <https://www.section.io/engineering-education/image-processing-using-matlab/>

**THANK YOU**