Scilab case study project on

Vehicle Number Plate Recognition

Vishesh Vinod Munghate

Visvesvaraya National Institute of Technology Nagpur

Image Processing

August 28, 2024

Abstract

This case study demonstrates how image processing techniques in Scilab can be used for vehicle number plate recognition. It outlines a process that includes converting RGB images to grayscale, applying noise reduction and edge detection, and using morphological operations to isolate the number plate. The recognition process employs bounding box segmentation and template matching to identify characters. Challenges such as image variability and character distortion are addressed, showcasing an effective method for automated vehicle identification using Scilab.

1. Introduction

Vehicle number plate recognition plays a critical role in automated systems for vehicle identification, traffic monitoring, and security. This case study explores the use of Scilab, an open-source software for scientific computing, to implement an effective number plate recognition system.

The process begins with converting RGB images of vehicles to grayscale and filtering them to remove noise. Key techniques such as mathematical morphology and edge detection are applied to isolate the number plate region. Following this, the study uses bounding boxes to segment individual characters on the plate and template matching to identify these characters accurately.

Challenges such as varying image quality, character distortion, and distinguishing between character and non-character regions are addressed. This case study illustrates how Scilab can be utilized to build a robust vehicle number plate recognition system, offering insights into practical image processing and computer vision techniques.

2. Problem Statement

- **Problem:** Given an RGB image of a car's front or rear showing its number plate, develop a Scilab-based program to accurately extract and display the vehicle's number.

- **Method of Approach:**

1. Extraction of Number Plate Location:
   - Convert the RGB image to grayscale.
   - Filter the image to remove noise.
   - Apply mathematical morphology to detect the number plate region.
   - Use the Canny edge detection operator to compute the threshold value.
   - Obtain a dilated image, then apply the `imfill` function to fill holes, resulting in a clear binary image.
   - Multiply this binary image with the grayscale image to isolate the number plate.

2. Segmentation:
   - Utilize bounding boxes to segment the number plate into individual characters.
   - Crop the bounding boxes to generate images of each character.

3. Recognition of Characters by Template Matching:
   - Compare the cropped character images against a complete alphanumeric database using template matching.
   - Use the correlation coefficient to measure how well each image matches the template, with the template size being 42x24 pixels.

- **Problems Faced:**

1. Variations in color, intensity, and contrast across different images affect thresholding and filtering parameters, requiring adjustments, especially for white and black cars.

2. Preprocessing steps like filtering, erosion, and dilation can distort characters, impacting recognition accuracy during template matching.
3. Bounding boxes may also encompass non-character features, complicating the process of ensuring that only character bounding boxes are used for template matching, which can lead to incorrect results.

3. Basic concepts related to the topic

3.1 Convolution

Convolution is a fundamental mathematical operation in image processing. It involves multiplying two arrays of numbers, generally of different sizes but the same dimensionality, to produce a third array of the same dimensionality. In image processing, one array is typically a grayscale image, and the second, smaller array is the kernel (or filter). Convolution is used to implement operators that produce output pixel values as linear combinations of input pixel values.

3.2 Filtering

In image processing, filters are used to either suppress high frequencies (smoothing the image) or enhance low frequencies (detecting or enhancing edges). This is achieved through convolution. Discrete convolution involves shifting the kernel over the image and multiplying its values with the corresponding pixel values. For a square kernel of size M × M, the output image is calculated using the formula:

$$g(i, j)=\sum_{m=-\frac{M}{2}}^{\frac{M}{2}} \sum_{n-=\frac{M}{2}}^{\frac{M}{2}} h(m,n)f(i-m,j-n)$$

3.3 Morphology

Morphological operations process binary images using a structuring element to combine them with set operators (intersection, union, complement). These operations focus on the shape of objects within the image, as defined by the structuring element.

3.3.1 Dilation

The basic effect of this operator on a binary image is to gradually enlarge the boundaries of regions of foreground pixels (i.e. white pixels, typically). Thus areas of foreground pixels grow in size while holes within those regions become smaller The dilation operator takes two pieces of data as inputs. The first is the image which is to be dilated. The second is a (usually small) set of coordinate points known as structuring element (also known as a kernel). It is this structuring element that determines the precise effect of the dilation on the input image.

3.3.2 Erosion

The basic effect of the operator on a binary image is to erode away the boundaries of regions of foreground pixels(i.e. white pixels, typically). Thus areas of foreground pixels shrink in size, and holes within those areas become larger. The erosion operator takes two pieces of data as inputs. The first is the image which is to be eroded. The second is a (usually small) set of coordinate points known as structuring element (also known as a kernel). It is this structuring element that determines the precise effect of the erosion. Larger structuring elements produced enhance effect of dilation or erosion.

3.4 Canny Edge Detection

The Canny edge detection algorithm is designed to identify edges optimally. It involves multiple stages:

1. Smoothing: The image is smoothed using Gaussian convolution.
2. Edge Detection: A 2-D first derivative operator highlights regions with high spatial gradients.
3. Non-Maximal Suppression: Edges are thinmed by zeroing pixels not on the ridge top.
4. Hysteresis Thresholding: Tracking begins at ridge points with a gradient higher than a primary threshold (T1) and continues until the gradient falls below a secondary threshold (T2). This process helps to connect edges and reduce noise.

3.5 Template Matching

Matching is an operation to determine the similarity between two entities (a template/reference signal/entity and a target signal/entity). For 2D images, template matching uses a reference image (the template), which can be a sample of a real image or, for some applications, a synthetized prototype of the pattern. The pattern is usually smaller than the image. The problem is to find if and where there is an occurrence (or at least a similar enough occurrence) of the template in the target image. The correlation approach uses the correlation coefficient as a measure of similarity between the reference (template) for each location (x,y) in the target image. The result will be maximum for locations where the template have correspondence (pixel by pixel) to the sub image located at (x,y).

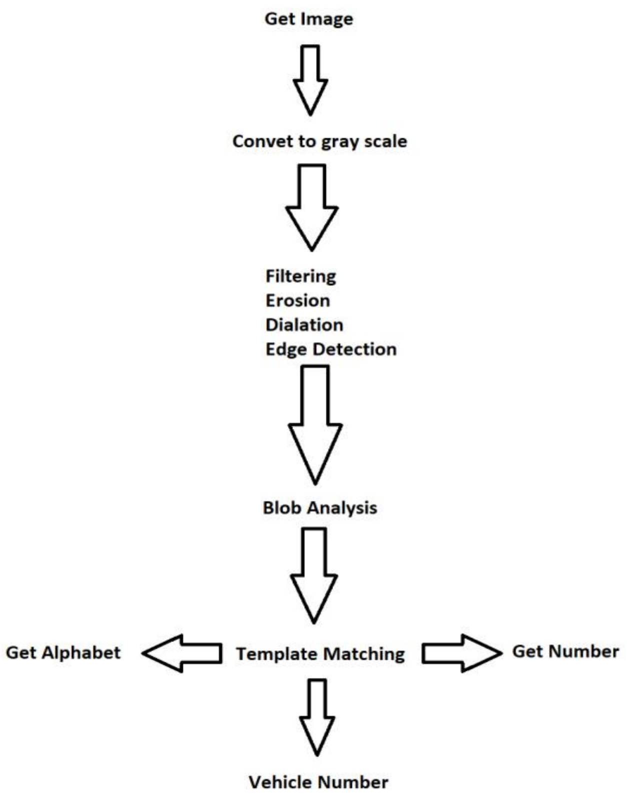
3.6 Blob Analysis

In computer vision, blob detection methods are aimed at detecting regions in a digital image that differ in properties, such as brightness or colour, compared to surrounding regions. Informally, a blob is a region of an image in which some properties are constant or approximately constant; all the points in a blob can be considered in some sense to be similar to each other. The most common method for blob detection is convolution.

3.7 Thresholding

An image processing method that creates a bitonal (aka binary) image based on setting a threshold value on the pixel intensity of the original image. The input to a thresholding operation is typically a grayscale or colour image. In the simplest implementation, the output is 3 a binary image representing the segmentation. Black pixels correspond to background and white pixels correspond to foreground (or vice versa). In simple implementations, the segmentation is determined by a single parameter known as the intensity threshold. In a single pass, each pixel in the image is compared with this threshold. If the pixel's intensity is higher than the threshold, the pixel is set to, say, white in the output. If it is less than the threshold, it is set to black.

4. Flowchart



5. Software/Hardware used

Operating System: Windows 10 Home

Scilab Version: 2024.1.0

Toolbox Used: Image Processing and Computer Vision (IPCV) toolbox

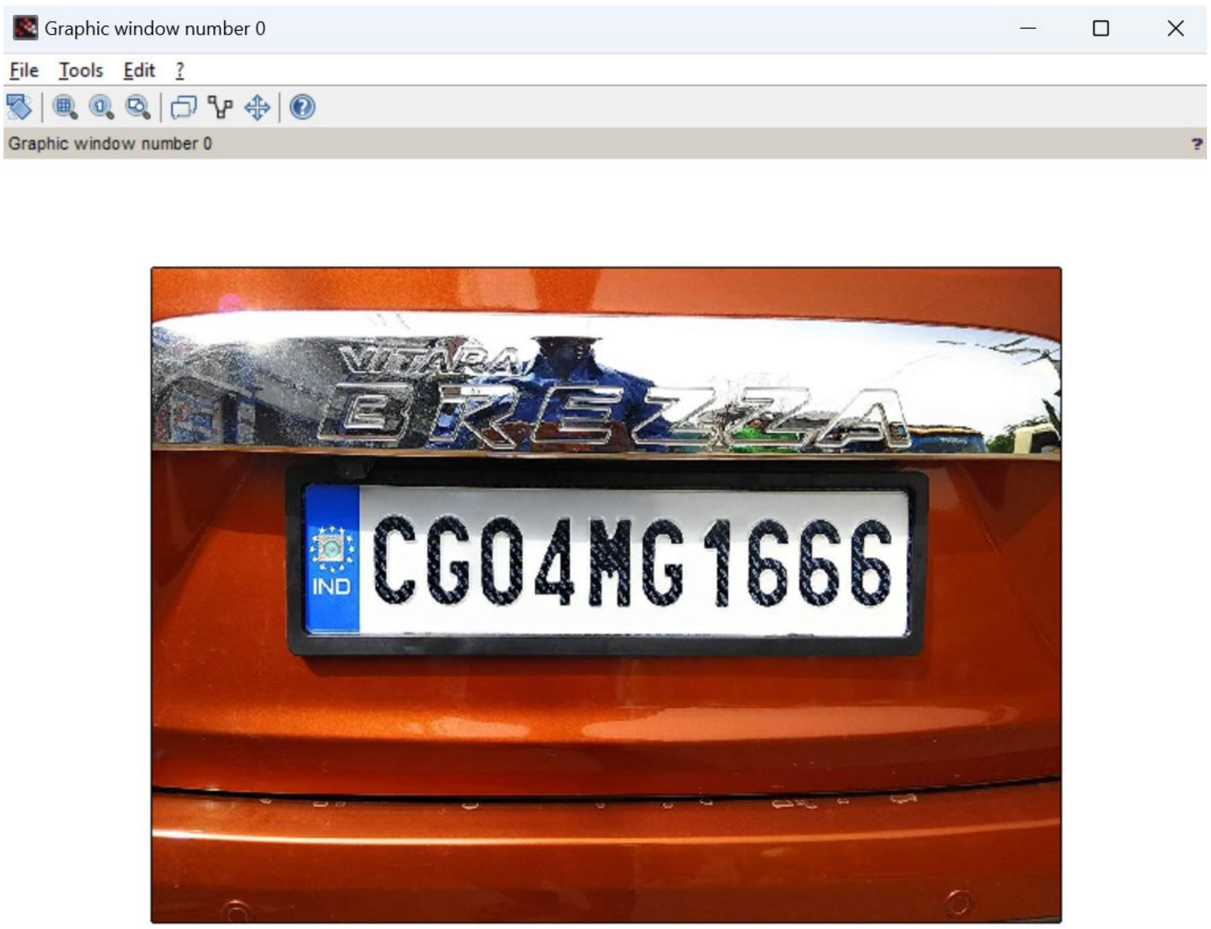
Hardware: MacBook Air 2019 with Windows 10 Home installed via Boot Camp

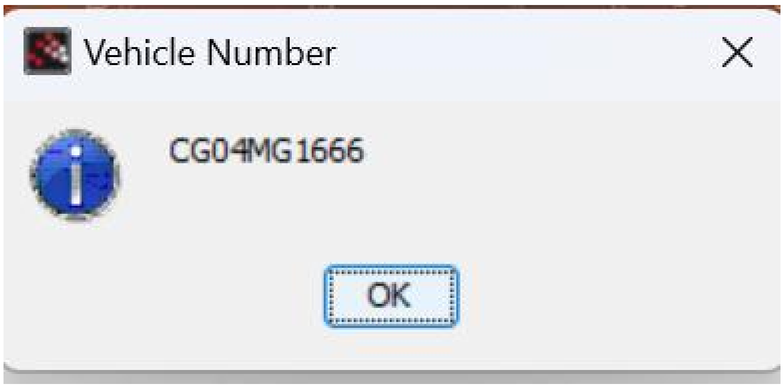
6. Procedure of execution

1. There are three code files involved: `main.sce`, `getletter.sci`, and `getnumber.sci`.
2. Set the current directory to the "Code files" folder.
3. Open the `main.sce` file in Scilab’s Scinotes editor.
4. Press the play button to execute the code.
5. A dialogue box will appear. Choose an image from the "Sample image" folder.
6. A message box will pop up, displaying the vehicle number extracted from the image.

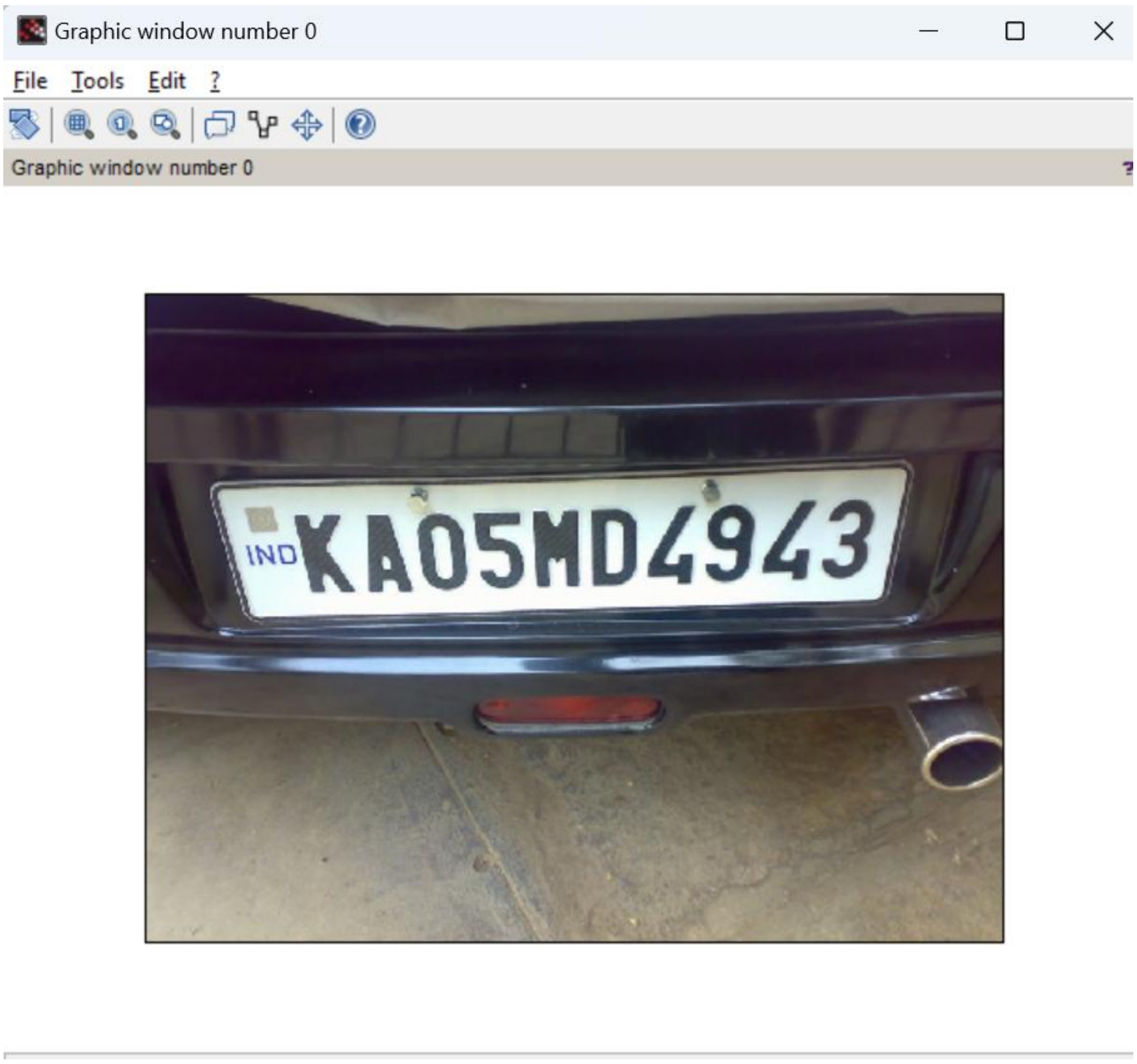
7. Result

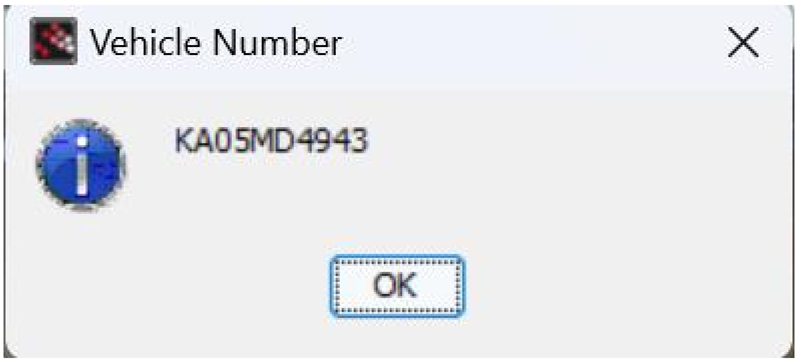
The Scilab-based program developed for vehicle number plate recognition was tested on a set of provided images. The program successfully recognized and extracted the vehicle number in each case, with the outputs matching the expected results exactly. This consistent accuracy across all test images demonstrates the robustness and reliability of the approach used in the implementation.




Vehicle Number

```
CG04MG1666
```


Graphic window number 0
File
Tools
Edit
?
Graphic window number 0
IND
KA05MD4943


Vehicle Number
OK

KA05MD4943

8. References

1. Gonzales, R. C. (n.d.). Digital Image Processing. Prentice Hall.

2. The Hypermedia Image Processing Resource. (n.d.). Retrieved from https://homepages.inf.ed.ac.uk/rbf/HIPR2/hipr_top.htm

3. Qadir, R. (2020). Vehicle number plate recognition. MATLAB Central File Exchange. Retrieved May 4, 2020, from https://www.mathworks.com/matlabcentral/fileexchange/40426-vehicle-number-plate-recognition

4. R. Bhat and B. Mehandia, "Recognition of Vehicle Number Plate Using MATLAB," International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering, vol. 2, no. 8, pp. 1899-1903, Aug. 2014. [Online]. Available: https://ijireeice.com/wp-content/uploads/2013/03/IJIREEICE3G-a-ragini-RECOGNITION-OF-VEHICLE-NUMBER-PLATE-USING-MATLAB.pdf. Accessed: Aug. 28, 2024.