Shape

Description automatically generated with medium confidence

**SVKM’s NMIMS**

**Mukesh Patel School of Technology Management and Engineering, Vile Parle, Mumbai – 400056**

**Department of Computer Sciences**

**Image and Video Processing : License Plate Detection and Extraction.**

Somesh Jadhwani (N022), Shreya Jain (N025)

under the guidance of Prof.Vidya Sawant.

# Problem Statement

This project seeks to identify the license plate from the vehicles present in an image and reads it to convert the license plate to text.

# Implementation

The software takes an image as an input and outputs the license plate number along with all the steps that it takes during the entire operation. The project is coded in python and uses the following libraries :

* OpenCV : Various Image operation
* MatPlotLib : Plotting the image
* NumPy : Creating and handling arrays
* SciKit-Image : Operations on image
* ImUtils : Accessory for Pytesseract
* Pytesseract : Text extraction from object

The project makes use of Image Processing concepts and uses the entire image processing pipeline that includes

* Image Acquisition
* Image Enhancement
* Image Analysis
* Image Compression
* Morphological Operation
* Image Segmentation
* Image Representation
* Object Detection

The software can be deployed for detection of speeding vehicles, penalising vehicles for not adhering to road rules, Parking Verification and also Automated Barrier for entry into protected areas.

To achieve the output, the image is first converted to a grey scale image and then it is segmented using thresholding method with the help of a gaussian , adaptive and otsu threshold. Following this , the image is resized and smoothened to remove noise. Then the image is passed through a canny edge detector following which the contours are extracted and stored in a list. We extract the top 30 contours in hopes of finding a rectangle that contains the number plate. Once the number plate has been detected, it is cropped and compressed to reduce the size and passed onto the pytesseract library that reads the text and outputs the license plate number.

# Source Code

import cv2

import imutils

import pytesseract

from matplotlib import pyplot as plt

from scipy import signal

import numpy as np

img = cv2.imread('test5.png',0)

gaussian = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C,cv2.THRESH\_BINARY, 51, 5)

adapt = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE\_THRESH\_MEAN\_C,cv2.THRESH\_BINARY, 31, 5)

ret, otsu = cv2.threshold(img, 120, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU)

plt.figure(figsize=(12,12))

plt.subplot(1,3,1)

plt.title("Image after Gaussian Thresholding")

plt.imshow(gaussian,cmap="gray")

plt.axis('off')

plt.subplot(1,3,2)

plt.title("Image after Adaptive Threshlding")

plt.imshow(adapt,cmap="gray")

plt.axis('off')

plt.subplot(1,3,3)

plt.title("Image after Otsu Threshlding")

plt.imshow(otsu,cmap="gray")

plt.axis('off')

#read and resize input image

#print("1. Gaussian filter")

#print("2. Adaptive filter")

#print("3. Otsu Filter")

#c= int(input("Make a choice"))

for i in range (0,3):

if(i==0):

cv2.imwrite('new.png',gaussian)

elif(i==1):

cv2.imwrite('new.png',adapt)

else:

cv2.imwrite('new.png',otsu)

image = cv2.imread('new.png')

img\_main = image.copy() #original image

image = imutils.resize(image, width=300 )

image\_resized = image.copy() #Resized image

#convert coloured image to gray

gray\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY) #Grey Image

#using mask for smoothening

mask=np.ones([3,3],dtype=int)

mask=mask/49

gray\_image\_s=signal.convolve2d(gray\_image,mask)

#Smooth Image

#gray\_image = cv2.bilateralFilter(gray\_image, 11, 17, 17)

#pass the smoothened image for egde detection using canny edge detection

t\_lower = 50 # Lower Threshold

t\_upper = 200 # Upper threshold

# Applying the Canny Edge filter

edged1 = cv2.Canny(gray\_image, 30, 200) #Canny Edge Detected Image

image\_canny =edged1.copy()

# Finding Contours

# Use a copy of the image e.g. edged.copy()

# since findContours alters the image

#cv2.CHAIN\_APPROX\_SIMPLE: It removes all redundant points and compresses the contour, thereby saving memory.

#cv2.RETR\_LIST: retrieves all contours

cnts,new = cv2.findContours(edged1.copy(), cv2.RETR\_LIST, cv2.CHAIN\_APPROX\_SIMPLE)

image1=image.copy()

#image2: destination image,cnts: all contours,negetive index -1:draw all countours

#3rd paramenter: colour, 4th parameter: thickness

cv2.drawContours(image1,cnts,-1,(0,255,0),3)

image\_contour = image1.copy() #Contoured Image

#sorts the contours based on area

#reverse= true gives largest contour, select contours uptill 30

# since number plate lies somewhere inthe middle of the size list

cnts = sorted(cnts, key = cv2.contourArea, reverse = True) [:30]

screenCnt = None

#screenCnt = None: Stores the number plate contour.

image2 = image.copy()

cv2.drawContours(image2,cnts,-1,(0,255,0),3)

image\_top30 = image2.copy() #top 30 countours image

i=7

#loop for finding the best contour for number plate

for c in cnts:

perimeter = cv2.arcLength(c, True)# second argument is true implies shape is closed

approx = cv2.approxPolyDP(c, 0.018 \* perimeter, True)# ture implies closed shape

#chooses the contours with four sides as this will probably be our number plate.

if len(approx) == 4:

screenCnt = approx

#This finds the coordinates of the part identified as the license plate.

x,y,w,h = cv2.boundingRect(c)

new\_img=image[y:y+h,x:x+w]

# cropped number plate image is stored

cv2.imwrite('./'+str(i)+'.png',new\_img)

i+=1

break

#This draws the contour selected to be the number plate on our original image.

#cv2.drawContours(image, [screenCnt], -1, (0, 255, 0), 3)

image\_detected\_plate = image.copy() #Image with detected license plate

Cropped\_loc = './7.png'

cropped\_image = cv2.imread(Cropped\_loc) #Cropped Image

plate = pytesseract.image\_to\_string(Cropped\_loc, lang='eng')

if(len(plate)==0 or len(plate)<10):

continue

else:

print("Number plate is:", plate)

break

plt.figure(figsize=(10,10))

plt.subplot(5,2,1)

plt.title("Original Image")

plt.imshow(img\_main,cmap="gray")

plt.axis('off')

plt.subplot(5,2,2)

plt.title("Resized Image")

plt.imshow(image\_resized,cmap="gray")

plt.axis('off')

plt.subplot(5,2,3)

plt.title("Grayscale Image")

plt.imshow(gray\_image,cmap="gray")

plt.axis('off')

plt.subplot(5,2,4)

plt.title("Smoothened Image")

plt.imshow(gray\_image\_s,cmap="gray")

plt.axis('off')

plt.subplot(5,2,5)

plt.title("Canny Edge Detector")

plt.imshow(image\_canny,cmap="gray")

plt.axis('off')

plt.subplot(5,2,6)

plt.title("Contour Detector")

plt.imshow(image\_contour,cmap="gray")

plt.axis('off')

plt.subplot(5,2,7)

plt.title("Top 30th Contours")

plt.imshow(image\_top30,cmap="gray")

plt.axis('off')

plt.subplot(5,2,8)

plt.title("Cropped")

plt.imshow(cropped\_image,cmap="gray")

plt.axis('off')

plt.subplot(5,2,9)

plt.title("Image with detected License Plate")

plt.imshow(image\_detected\_plate,cmap="gray")

plt.axis('off')

# Test Cases and Outputs



Figure 1.1 : Input Image For Test Case 1

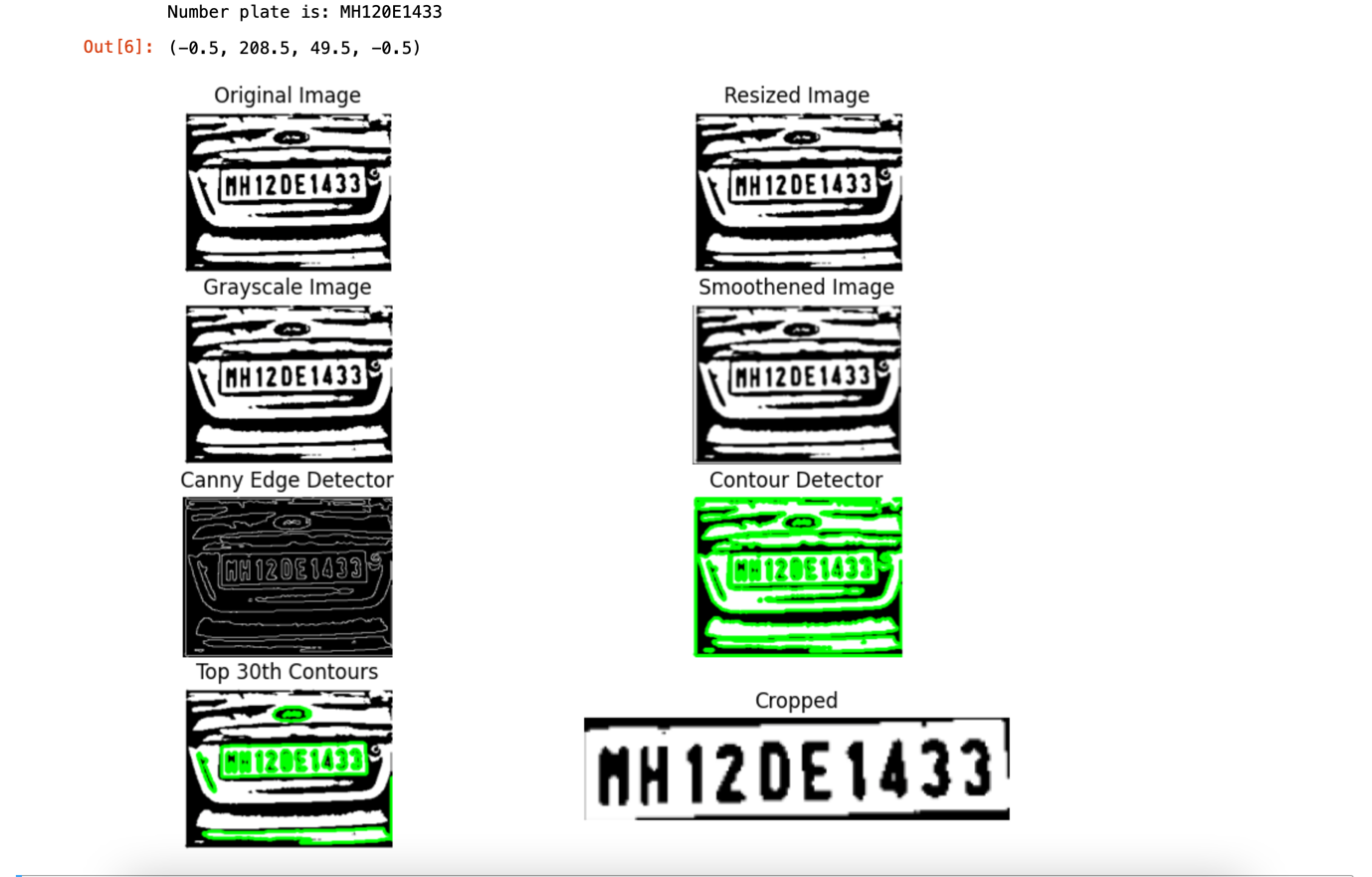


Figure 1.2 : Output for Test Case 1



Figure 2.1 : Input for Test Case 2

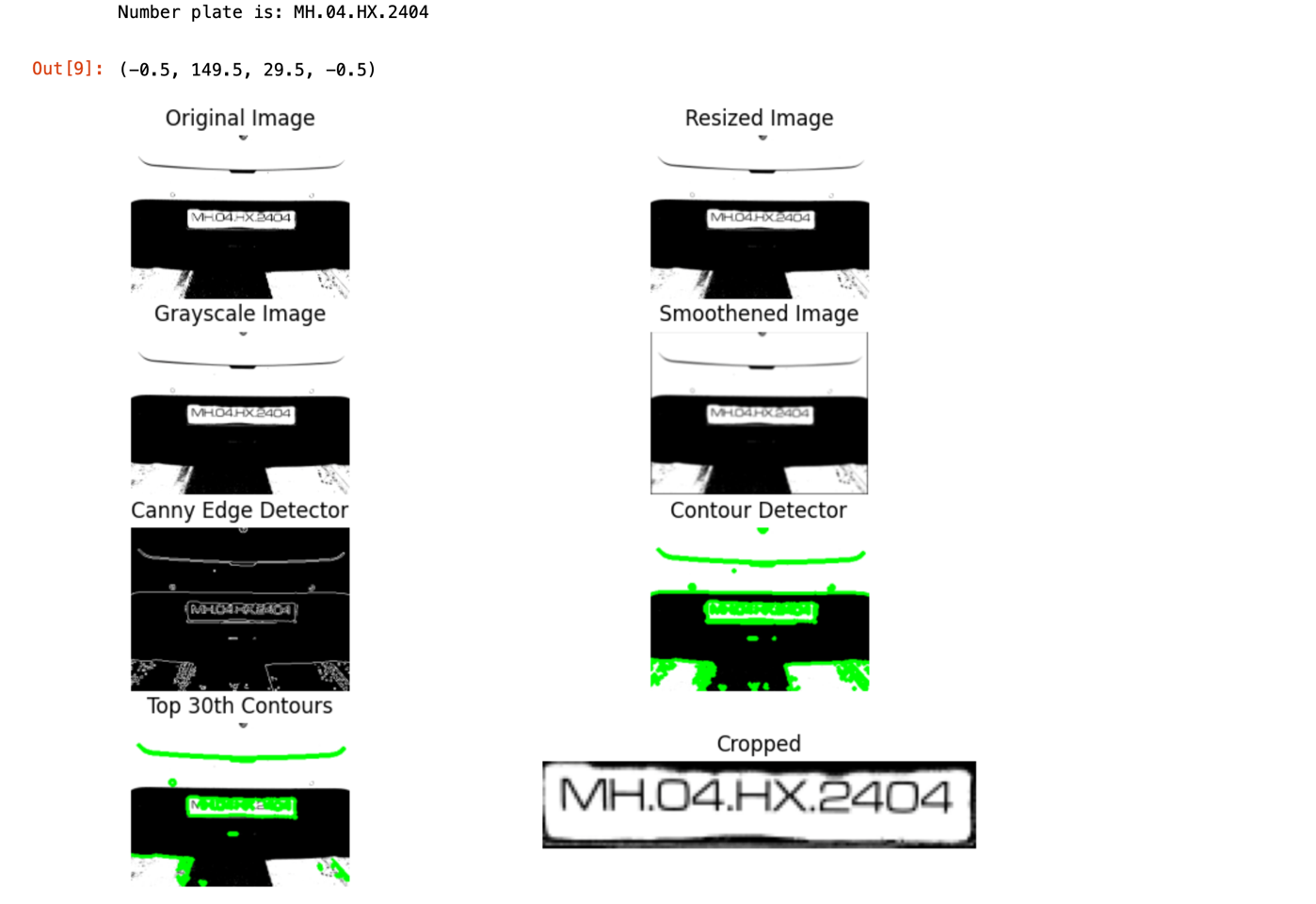


Figure 2.2 : Output for Test Case 2



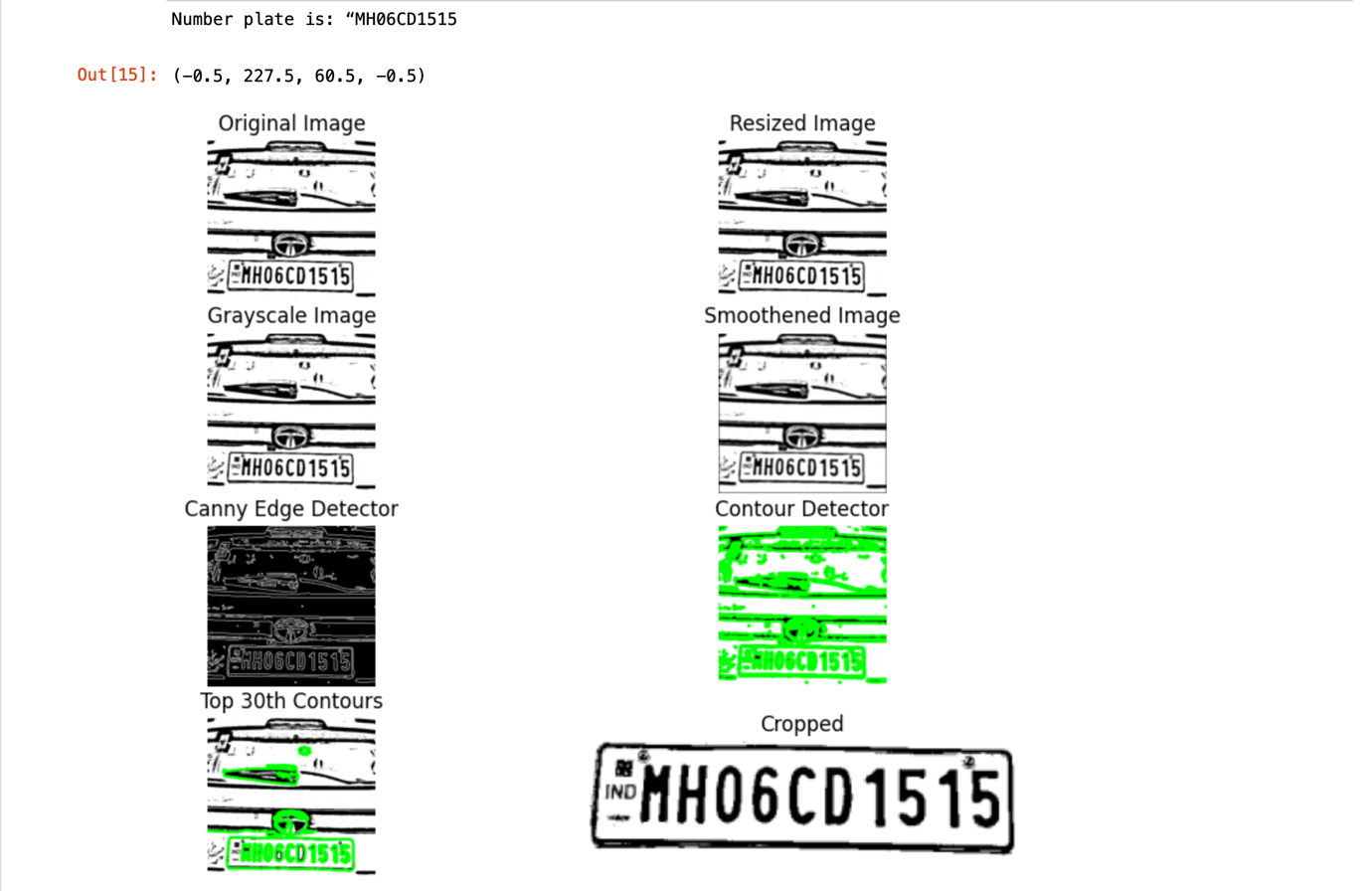


Figure 3.2 : Output for Test Case 3

Figure 3.1 : Input for Test Case 3

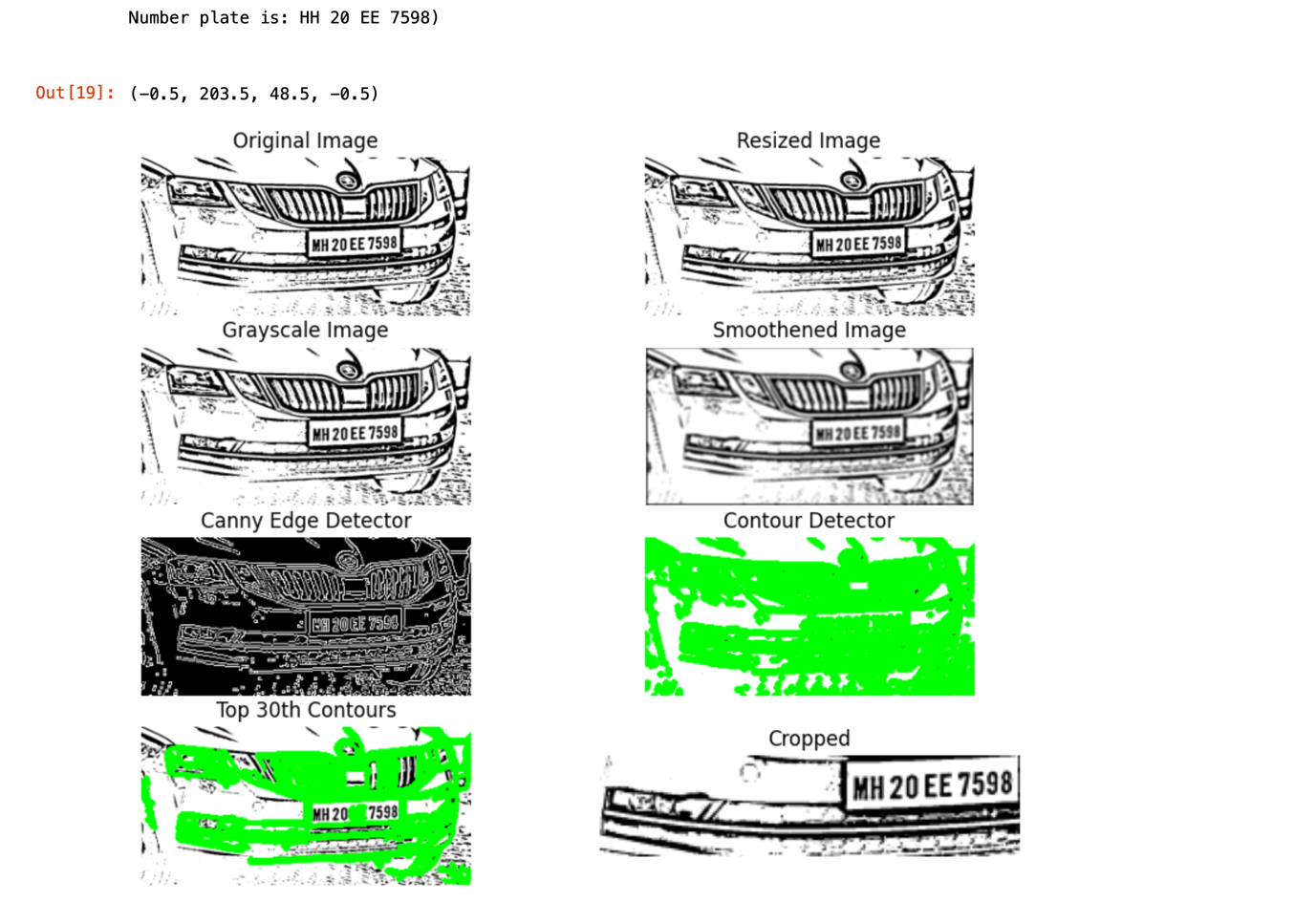


Figure 4.2 : Output for Test Case 4

Figure 4.1 : Input for Test Case 4



Figure 5.1 : Input for Test Case 5

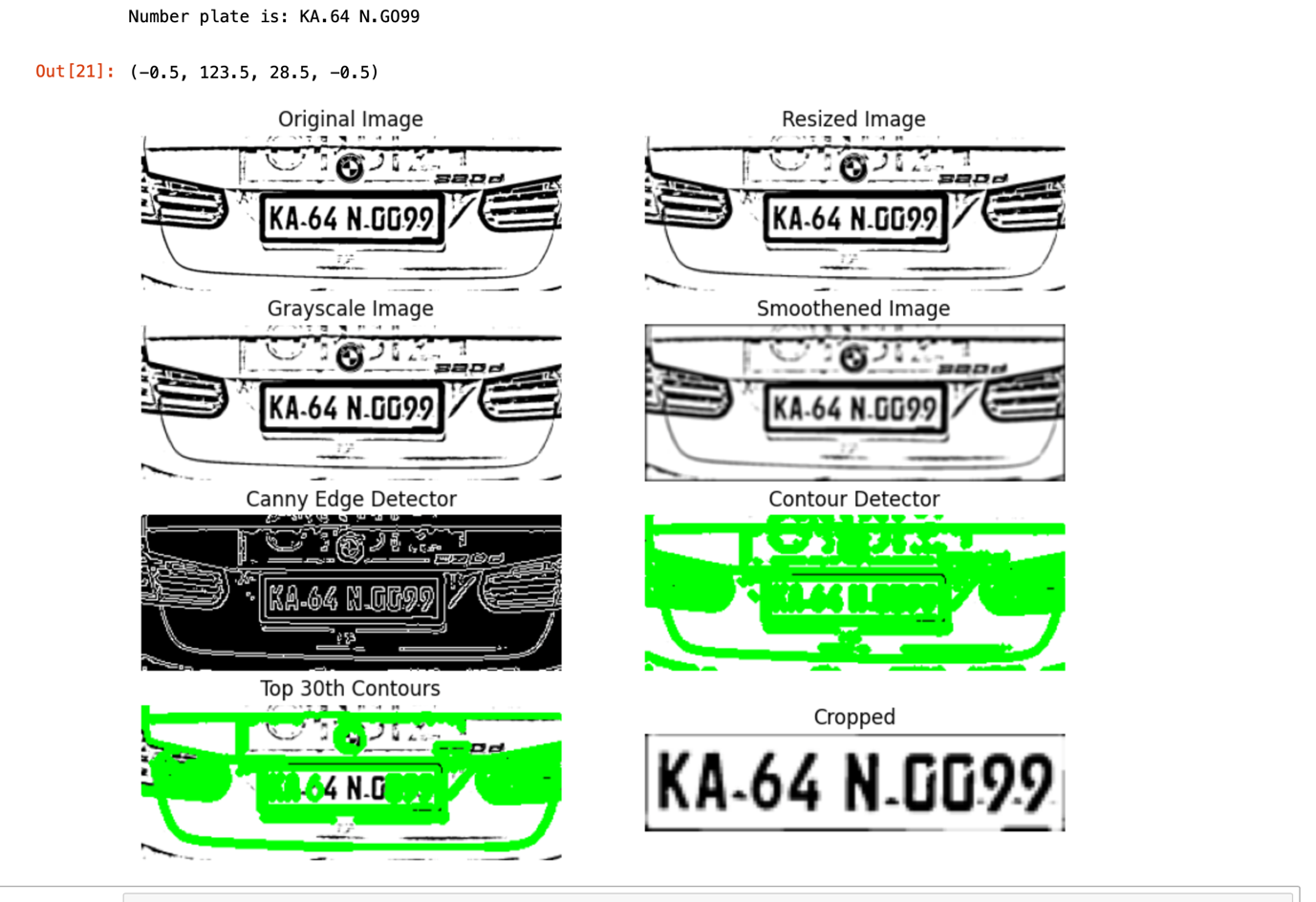


Figure 5.2 : Output for Test Case 5

# Inferences

To successfully get a license plate off an image, we had to process the image in several stages that included resizing, converting to grey scale, smoothing to remove noise, and then using a canny edge detector to generate the contours an extract the license plate from the image. While coding the project, we observed that the success rate of the algorithm improved greatly after including segmentation by thresholding. However, due to the variety of inputs, to successfully segment the image in order to have the number plate clearly visible, we had to use both global and local segmentation techniques. Since it wasn’t possible to determine the segmentation technique required for the given image beforehand, we decided to apply all three types of segmentations one at a time and process the image.This slightly increased the time complexity of the project but greatly increased the success rate and hence the trade off was deemed feasible.

# Conclusion

The software uses multiple image processing concepts to find the license plate of a vehicle and convert it to text using the Pytesseract library. The software also shows the various steps that it takes in the processing of the image in order to achieved the desired output.

As the provided test cases show ,the software works as desired and reliably extracts the license plate from the given image.