DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING

COLLEGE OF E&ME, NUST, RAWALPINDI

Digital Image Processing

Lab Final

Student Name: **Muhammad Hamza Tariq – 371070**

Degree/ Syndicate: **43 CE B**

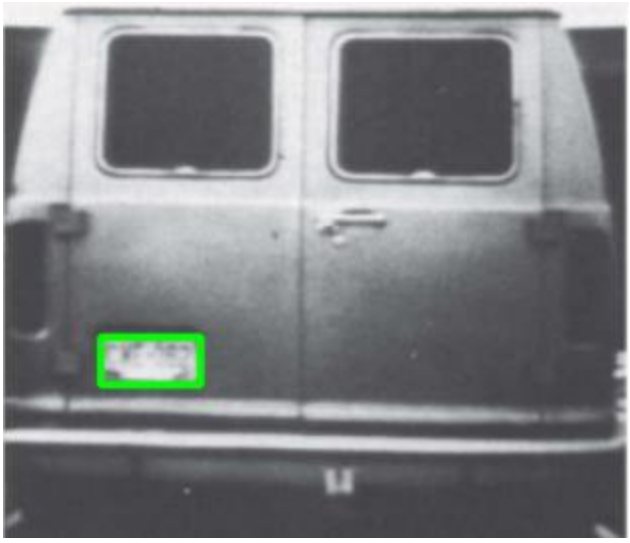
Code:

```
import cv2
import numpy as np

image = cv2.imread('Lab\lab_final\images\lab final.tif')
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
blurred = cv2.GaussianBlur(gray, (5, 5), 0)
edges = cv2.Canny(blurred, 30, 150)
rect_kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (15, 3))
dilated = cv2.dilate(edges, rect_kernel, iterations=1)
eroded = cv2.erode(dilated, rect_kernel, iterations=1)
contours, _ = cv2.findContours(eroded, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

possible_plates = []
for contour in contours:
    x, y, w, h = cv2.boundingRect(contour)
    aspect_ratio = w / float(h)
    if 2 <= aspect_ratio <= 5:
        if w > 30 and h > 10:
            possible_plates.append((x, y, w, h))
possible_plates = sorted(possible_plates, key=lambda x: -x[2]*x[3])

if possible_plates:
    (x, y, w, h) = possible_plates[0]
    cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)
cv2.imshow('image', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



Pre processing:

```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
blurred = cv2.GaussianBlur(gray, (5, 5), 0)
```

Detecting edgs:

```
edges = cv2.Canny(blurred, 30, 150)
```

rectangular structuring element

```
rect_kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (15, 3))
```

Apply morphological operations

```
dilated = cv2.dilate(edges, rect_kernel, iterations=1)
eroded = cv2.erode(dilated, rect_kernel, iterations=1)
```

Finding the rectangular shapes:

```
contours, _ = cv2.findContours(eroded, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

Finding the rectangle that has similar aspect ratio:

```
possible_plates = []
for contour in contours:
    x, y, w, h = cv2.boundingRect(contour)
```

```
    aspect_ratio = w / float(h)
    if 2 <= aspect_ratio <= 5:
        if w > 30 and h > 10:
            possible_plates.append((x, y, w, h))
```

Finding the brightest plate

```
possible_plates = sorted(possible_plates, key=lambda x: -x[2]*x[3])
```

Mark the plate

```
if possible_plates:
    (x, y, w, h) = possible_plates[0]
    cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)
```

Display:

```
cv2.imshow('image', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```