In [123]:

```
import cv2
import imutils
import numpy as np
import pytesseract
from PIL import Image
import matplotlib.pyplot as plt
pytesseract.pytesseract.tesseract_cmd=r"C:\Program Files\Tesseract-OCR\tesseract.exe"
```

In [124]:

```
Image.open('IMG_20201218_130821.jpg')
```

Out[124]:

1. Locating number plate (phase-1)

In [125]:

```
img=cv2.imread('IMG_20201218_130821nn.jpg',cv2.IMREAD_COLOR)
img=cv2.resize(img,(1200,800))
```

In [127]:

```
cv2.imwrite('Resized.jpg',img)
Image.open('Resized.jpg')
```

Out[127]:

In [128]:

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) #transforming the whole image into an image th
```

In [129]:

```
gray=cv2.bilateralFilter(gray,3,15,15)    #to blur out the noise , noise in our case is any
```

In [130]:

```
cv2.imwrite('grayed.jpg',gray)
Image.open('grayed.jpg')
```

Out[130]:

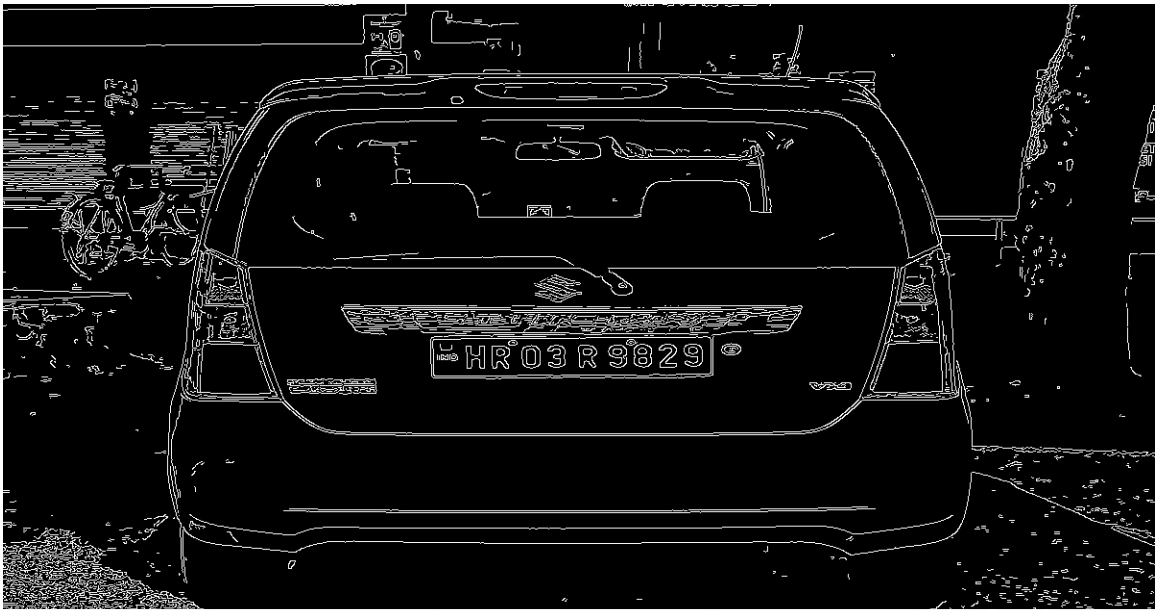
In [131]:

```
edged=cv2.Canny(gray,70,100)
# performing edge detection
# (sourceimage,minthresh,maxthresh)
# here only those edges having intensity gradient more than min
# threshold will be shown
```

In [132]:

```
cv2.imwrite('edged.jpg',edged)
Image.open('edged.jpg')
```

Out[132]:



In [133]:

```
#finding the contours- locating the number plate
contours= cv2.findContours(edged.copy(),cv2.RETR_TREE,
                           cv2.CHAIN_APPROX_SIMPLE)
contours=imutils.grab_contours(contours)
contours=sorted(contours,key=cv2.contourArea,reverse=True)[:10]
screenCnt=None
```

In [134]:

```
for c in contours:
    #approx contour
    peri=cv2.arcLength(c,True)
    approx=cv2.approxPolyDP(c,0.02*peri,True)
    #if our approx contour has four points, then we can assume that we have found our scree
    if len(approx)==4:
        screenCnt=approx
        break
```

In [135]:

```
if screenCnt is None:
    detected=0
    print('no contour detected')
else:
    detected=1
if detected==1:
    cv2.drawContours(img,[screenCnt],-1,(0,0,255),3)
```

In [136]:

```
cv2.imwrite('contour_im.jpg',img)
Image.open('contour_im.jpg')
```

Out[136]:

2. Masking and character segmentation (phase -2)

Removing the rest of the area by masking other part expect the detected number plate

In [137]:

```
mask= np.zeros(gray.shape,np.uint8)
new_img=cv2.drawContours(mask,[screenCnt],0,255,-1)
new_img=cv2.bitwise_and(img,img,mask=mask)
```

In [138]:

```
cv2.imwrite('masked.jpg',new_img)
Image.open('masked.jpg')
```

Out[138]:

In [139]:

```
#cropping the plate
(x,y)=np.where(mask==255)
(topx,topy)=(np.min(x),np.min(y))
(bottomx,bottomy)=(np.max(x),np.max(y))
Cropped=gray[topx:bottomx+1,topy:bottomy+1]
```

In [121]:

```
cv2.imshow('cropped plate',Cropped)
cv2.waitKey(0)
```

Out[121]:

```
-1
```

In [140]:

```
cv2.imwrite('cropped.jpg',Cropped)
Image.open('cropped.jpg')
```

Out[140]:

preparing cropped image before character recog phase

In [141]:

```
Cropped=cv2.resize(Cropped, None, fx=2,fy=2)
```

In [142]:

```
cv2.imwrite('cropped2.jpg',Cropped)
Image.open('cropped2.jpg')
```

Out[142]:

In [143]:

```
Cropped=cv2.adaptiveThreshold(Cropped,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY,
```

In [144]:

```
cv2.imwrite('cropped3.jpg',Cropped)
Image.open('cropped3.jpg')
```

Out[144]:

3. Character recognition (phase-3)

In [145]:

```
text=pytesseract.image_to_string(Cropped,config='--psm 11',lang='eng')
print('detected license plate number is:',text)
```

```
detected license plate number is: ~HROZRS8239
```