MOSAIC PS2

Team- ULTRONIX

Image Segmentation Part-

CHALLENGES FACED-

1.Determining the threshold type

2.Handle different lightning conditions

3.Removing Noise in the background

4.Handling rotated Number Plate

Firstly we converted the RGB image to Grayscale image. We found that there are shadows in License Number Plate images which makes it very much difficult to threshold the image properly .So we applied a signals and image processing technique called homomorphic filtering.Homomorphic filtering is used for correcting non-uniform illumination .We can represent an image in terms of a product of illumination and reflectance.

Homomorphic filtering tries and splits up these components and we filter them individually using high pass and low pass filter.

This basically uses fast fourier transform to separate multiplicative components of the image in the frequency domain.

And then these components are filtered using high pass and low pass filters.

Original Image

After Homomorphic transform

Now to further enhance the quality of image we applied another image processing technique called histogram equalization.

Below we implemented another technique called histogram equalization for making either under exposed or over exposedimages well exposed.

```
#images well exposed


equalized = cv2.equalizeHist(gray)
#kernel = cv2.getStructuringElement
```

We also tried adaptive histogram equalization (CLAHE) but it made the images very darker so we stick to normal histogram equalization.

Now since the image has become quite dark so we increased contrast and brightness of image using function given below

```python
def apply_brightness_contrast(input_img, brightness = 0, contrast = 0):

    if brightness != 0:
        if brightness > 0:
            shadow = brightness
            highlight = 255
        else:
            shadow = 0
            highlight = 255 + brightness
        alpha_b = (highlight - shadow)/255
        gamma_b = shadow

        buf = cv2.addWeighted(input_img, alpha_b, input_img, 0, gamma_b)
    else:
        buf = input_img.copy()

    if contrast != 0:
        f = 131*(contrast + 127)/(127*(131-contrast))
        alpha_c = f
        gamma_c = 127*(1-f)

        buf = cv2.addWeighted(buf, alpha_c, buf, 0, gamma_c)

    return buf
```

Initial Image

After Histogram Equalization

After homomorphic filteringIncreasing contrast and brightness

Now finally we get the image which can easily be thresholded using opencv

Now comes a question whether to choose adaptive thresholding or normal thresholding or some otsu thresholding

We tried all and we found that adaptive thresholding is not working for all scenarios sometimes it masks the noise in the background also and also sometime when the characters are too close than adaptive thresholding merges two or more characters.

So we decided to use simple thresholding with a very low threshold as in previous preprocessing techniques we already made the characters very much darker so using low threshold masks only very dark areas of image.

Now for segmenting characters we used open cv's connected components with stats function

We separated noisy components from the character components using different limits for area , position , aspect ratio, height, width.

Like it will consider only those components whose area value lies between particular limits etc.

```
def connectedcomp(thresh):
    connectivity = 4

    # Perform the operation
    output = cv2.connectedComponentsWithStats(thresh, connectivity, cv2.CV_32S)
    numLabels, labels, stats, centroids=output
    mask = np.zeros(thresh.shape, dtype="uint8")
    new_mask=mask.copy()
    #cv2.imshow("winname",thresh)
    #cv2.waitKey(0)
```

Complete function can be found in full code

Now we drawn all the characters component on black background

Now comes how to handle rotation first we dilated image using a kernel having more value in x direction than y direction to connect characters horizontally. Then we used minarearect function in open cv to get angle of the rotated plate.

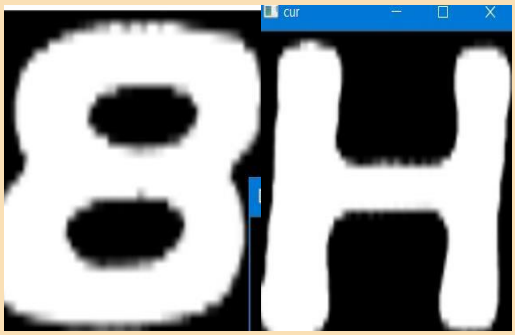
```
largestContour = contours[0]
minAreaRect = cv2.minAreaRect(largestContour)

# Determine the angle. Convert it to the value that was origina
angle = minAreaRect[-1]
print(angle)
if angle>45:
    return 90-angle
else :
    return -angle
```

This is how we got the angle and then we used open cv warpaffine method to rotate image

```
#this function takes angle as argument and rotates the image
def rotateImage(cvImage, angle: float):
    newImage = cvImage.copy()
    (h, w) = newImage.shape[:2]
    center = (w // 2, h // 2)
    M = cv2.getRotationMatrix2D(center, angle, 1.0)
    newImage = cv2.warpAffine(newImage, M, (w, h), flags=cv2.INTER_CUBIC, borderMode=cv2.BORDER_REPLICATE)
    return newImage
```

Now we extracted the characters using countours and found bounding box using bound rect function

```python
def extract(image):
    newimg=image.copy()
    kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (1,7))
    dilate = cv2.dilate(newimg, kernel, iterations=2)
    #cv2.imshow("j",dilate)
    #cv2.waitKey(0)
    cnts,_ = cv2.findContours(newimg, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    #print(np.mean(cnts[0],axis=0))
    cnts.sort(key=lambda x:np.mean(x,axis=0)[0][0])
    ls=[]
    for cntr in cnts:
        if(cv2.contourArea(cntr)>=1000):
            x,y,w,h = cv2.boundingRect(cntr)
            cropped=image[y:y+h,x:x+w]
            cropped=cv2.resize(cropped,(256,256))
            kernel = np.ones((3,3), np.uint8)
            cropped=cv2.erode(cropped, kernel,iterations=1)
            ls.append(cropped)
    return(ls)
```



Some extracted characters

Model Making Part-

Firstly we tried our own model which is based on resnet architecture we trained it using the given dataset of characters but the model was not giving good results.

As we are allowed to used pretrained models so we tried tesseract model it performed better than previously trained model but sometimes it was misclassifying the characters. At last we decided to use EasyOCR to identify the car number and it worked pretty well.It takes image as the input and returns output in the format of the list, where each item of the list represents bounding box, text and confidence level.

For plate extraction from car images we used YOLOv3 pretrained model. From the coordinates of the bounding boxes we got of number plates present in the image we extracted those parts for final segmentation.

This model is trained over 3000 images of vehicles in different viewpoint and lighting conditions.

And we also tried to detect number plates on video but it was quite slow but working fine.

Link for more information about Histogram Equalization and Homomorphic filtering-

https://drive.google.com/file/d/1wupnCfsfZxWvq33vdmmwT5YHWWMz1U6z/view?usp=sharing Link for YOLOv3 research paper -

https://www.irjet.net/archives/V7/i3/IRJET-V7I3756.pdf

Link for YOLOv3 pretrained model - https://drive.google.com/drive/folders/14e051ocMTDwH7EHP4Y7l-RkFa-LUe-vR

Link for Easyocr – https://pypi.org/project/easyocr

FOR ACCURACY METRIC WE PERFORMED SEGMENTATION OF 19 DIFFERENT IMAGES GIVEN TO US IN ROUND 2
(images can be found in final_test folder)

We got the following result for the 19 images—

1.ok
2.one extra noise considered as 1 but all classified
3.one misclassification
4.ok
5.ok
6.ok
7.ok
8.ok
9.ok
10.ok
11.ok
12.ok
13- one misclassification
14 -missing 4 char because of too noise
15- too much noise not able to segment
16.ok
17.one misclassification of M with H and 1 is missing at very corner
18. two misclassifications G with 6 and M with H
19.ok