Faculty of Engineering

Computer & Systems Eng. Dept.

Digital image processing

License Plate Recognition

Name: Hoda Abdelbasit Mohammed ID: 43863

Heba Allah Essam Eldein ID: 43862

Enas Hoseny Sa'aed ID: 43740

Presented to: Eng. Aly Osama

Submission Date: 3/5/2018

Our code is based on a paper where we tried as hard as we can to apply it on the code

First we get the **pixel intensity** where the intensity = 0.3 * red layer + 0.59 * green layer + 0.11 * blue layer ,as we did this through the function which called (pixel_intensity(x, y))

```
def pixel_intensity(x, y):
    if y >= yimage or x>=ximage:
        return 0
    pixel =image[y][x]
    return (0.3* pixel[2])+0.59*pixel[1] + 0.11*pixel[0]
```

we calculate some **children** of the contours by counting it

- if it's an important contour and itslevel is the same level of other contours , if not then we discard it and still count the same level of the others to this contour

possible char is a function to determine whether this contour is a possible char or not ... it has many steps as :

- we get the signs of the rectangle through cv2.boundingRect method and it returns width, height, x and y (x,y are the top-left coordinate of the rectangle)
- then after that we get the area of this rectangle (width * height) and the aspect ratio which is width divided by length
- the contour is a possible character if the aspect ratio greater than or equal 0.25 (like I letter its width is too small compared to its height) and less than or equal 1(like W letter its width is approximately equal to the height) and the area too must be less than 100

- the other condition to tell whether this is a character or not is to determine the connectivity of this contour using cv2.isContourConvex method

Here is the function mentioned above :

```
def possible_char(contour):
    x, y, width, height = cv2.boundingRect(contour)
    aspect_ratio = (width*1.0)/(height*1.0)
    area= (width*1.0)*(height*1.0)
    if aspect_ratio < 0.25 or aspect_ratio > 1 or (area <= 100):
        return False
    return True  and not cv2.isContourConvex(contour)
```

If the contour **is a child** then it at least has one parent:

-we **get** the first **parent** of the contour by looping through all the hierarchy and it stops when it does not have more than 0 parent, or it is not a possible character then we return this parent .

```
def get_parent(i, hierarchy):
    pt = hierarchy[i][3]
    while possible_char(contours[pt]) and pt > 0:
        pt= hierarchy[pt][3]
    return pt
```

-we also count the **number of same level contours** of this contour : if it has parent then loop on it as long as it is not -1 then count the child of this parent (through this way we can get the same level contours of any contour).

```
def count_siblings(i, hierarchy, contour):
    count = 0
    p = hierarchy[i][0]
    while p > 0:
        if possible_char(contours[p]):
            count += cc(p, hierarchy, contour)
        p = hierarchy[p][0]
    return count
```

There is also **include box** method which takes the index , hierarchy and the contour . this method determine if this is a necessary box or not by returning false if the contour is a child and if the contour is less than or equal 2 and true other wise.

```
def include_box(i, hierarchy, contour):
    if is_c(i, hierarchy) and cc(get_parent(i, hierarchy), hierarchy, contour) <= 2:
        return False
    return True
```

These are all the functions needed .

In the main code-since we add it into the function that called test- first we read the image and get the edges using canny in cv2 and setting the min , max threshold with proper values .

```
def test():
     global contours
     global OUT
     txt=""
     global image, yimage, ximage
     image = cv2.imread(filename)
     yimage = len(image)
     ximage = len(image[0])
     edges=cv2.Canny(image,400, 1000)
```

Then we need to find the contours and the hierarchy of the edges. for each contour we find the bounding rectangle parameter (x,y,width ,height) and if this contours is a possible char and of an acceptable box then we take it with us as follow :

```
  _,contours, hierarchy = cv2.findContours(edges.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)
  hierarchy = hierarchy[0]

ind the bounding rect for each one in the contours
  for i, contourr in enumerate(contours):
      x, y, width, height = cv2.boundingRect(contourr)
      if possible_char(contourr) and include_box(i, hierarchy, contourr):
          OUT.append([contourr, [x, y, width, height]])

  final_image = edges.copy()
  final_image.fill(255)
```

Then for each box we get the foreground intensity. To get the foreground intensity we calculate the average intensities of the edge pixels.

```
for i, (contourr, box) in enumerate(OUT):

    foreg_intensity = 0.0
    for p in contourr:
        foreg_intensity += pixel_intensity(p[0][0], p[0][1])

    foreg_intensity /= len(contourr)

    x, y, width, height = box
```

After that we loop on the box in the 2 dimensions and if the pixel intensity of this point greater than foreground intensity then put the new image is set with 255 else it is set with 0 .

```
for xx in range(x, x + width):
    for yy in range(y, y + height):
        if pixel_intensity(xx, yy) > foreg_intensity:
            final_image[yy][xx] = 255
        else:
            final_image[yy][xx] = 0
```

Then blur the final image for the ocr operation and we write this new image with name ‘out.png’ to allow to read it in tesseract

the function that called tesseract take the image and if there is any letter in that image , it will transform it into text.

```
final_image = cv2.blur(final_image, (3, 3))
cv2.imwrite("out.png", final_image)
OUT=[]
new_image='out.png'
file=PIL.Image.open(new_image)
txt = pytesseract.image_to_string(file)
label1.configure(text=txt)
```

Here we come to the GUI part .. so

- first we specialize a canvas to show the output image before the process of tesseract function .

```
window=tk.Tk()

window.title('License Plate Recognition')
canvas=Canvas(window,width=1000,height=500)
canvas.config(scrollregion=[0,0,400,400])
canvas.pack()
```

-Then there is a button (Select input image) to let you select the input image , then the path of selected image is stored in a variale which called (filename) ,since it the same variable that used in imread function to read the input image at the first step of the whole process .

```
btn2=Button(window,text='Select Input Image',command=PhotoCallBack)
btn2.pack()
```

-There is also anoter button to let you show the output image after applying all the processes mentioned above , and it will be showed into the canvas we mentioned before .

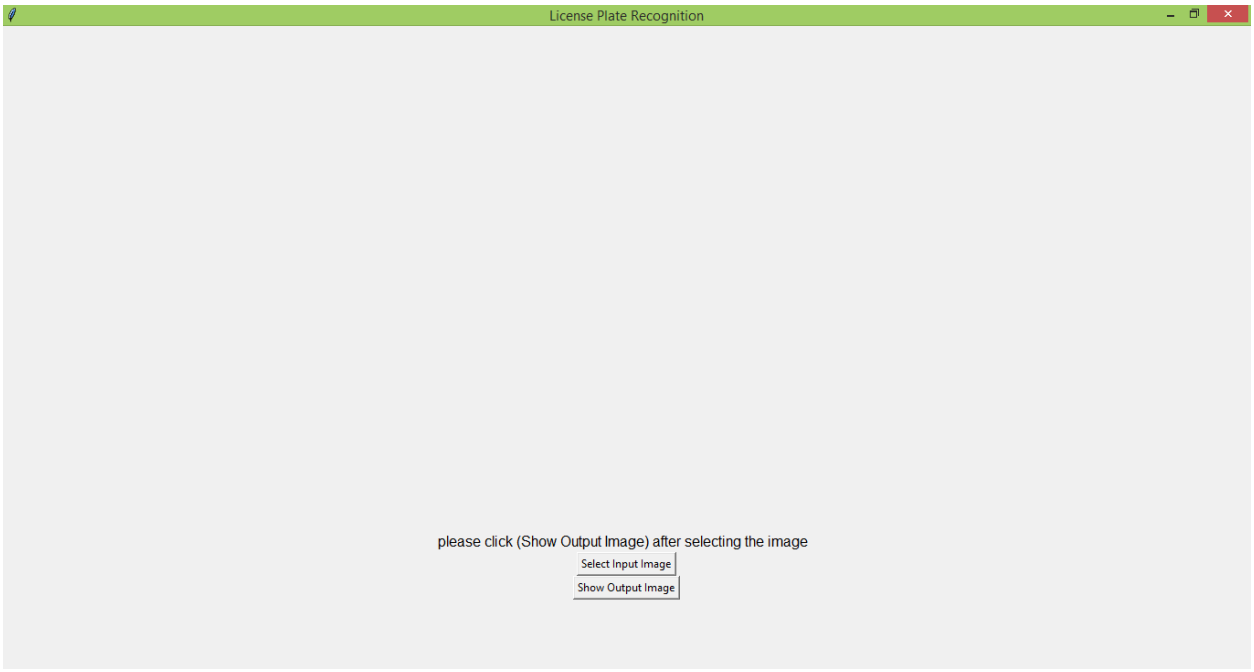
```
btn2=Button(window,text='Select Input Image',command=PhotoCallBack)
btn2.pack()
```

-Finally the last part of the process is to sow the text extracted from the image as the license number , so we define a label to display that text as follow .

```
label1 = Label(window, text="")
label1.pack()
label1.place(x = 10, y = 60)
```

Hint : Wait a little time between two run operation for load code

GUI



Test cases

Input image

Output

Input

Output


License Plate Recognition
NITESKY
" mi
NITESKY]
Select Input Image