Misr University for Science & Technology

College of Information Technology

جامعة مصر للعلوم والتكنولوجيا كلية تكنولوجيا المعلومات

s

CS - IS 409 Selected Topics in CS - IS

Project Title

Spring 2022 - 2023

Team members:

Name
Omnia ashraf

Submitted to:

Prof. Dr. Mostafa Gadal-haqq

Misr University for Science & Technology

College of Information Technology

جامعة مصر للعلوم والتكنولوجيا كلية تكنولوجيا المعلومات

EN: Mariam Essam

List of Content

Table of Contents

1.1 Overview

1.2 Automatic vehicle license plate detection and recognition is a key technique in most traffic related applications and is an active research topic in the image processing domain. Different methods, techniques and algorithms have been developed for license plate detection and recognition. Approach: Due to the varying characteristics of the license plate from country to country like numbering system, colors, language of characters, style (font) and sizes of license plate, further research is still needed in this area. Results: In most of the middle East countries, they use the combination of Arabic and English letters, plus their countries logo. Thus, it makes the localization of plate number, the differentiation between Arabic and English letters and logo's object and finally the recognition of those characters become more challenging research task.

1.3 Problem Statement (Definition)

suppose an ANPR system is mounted on a toll road. It needs to be able to detect the license plate of each car passing by OCR the characters on the plate, and then store this information in a database so the owner of the vehicle can be billed for the toll.

Several compounding factors make ANPR incredibly challenging, including finding a dataset you can use to train a custom ANPR model! Large, robust ANPR datasets that are used to train state of-the-art models are closely guarded and rarely (if ever) released publicly:

- These datasets contain sensitive identifying information related to the vehicle, driver, and location.
- ANPR datasets are tedious to curate, requiring an incredible investment of time and staff hours to annotate.

- ANPR contracts with local and federal governments tend to be highly competitive. Because of that, it's often not the trained model that is valuable, but instead the dataset that a given company has curated.

For that reason, you'll see ANPR companies acquired not for their ANPR system but for the data itself!

1.4 The Proposed solution

1. Import packages Here, we are importing OpenCV and inutile.
2. Read image file from specific folder. It is good to have it in the source file directory.
3. Resizing. input image because every image comes with a different shape. So, resizing makes it all in one standard size.
4. Color conversion, here, we are converting the input color (BGR) image into a grayscale image. Because the canny edge detector input **image** should be a single-channel 8-bit input image.
5. Image smoothing, Filtering is perhaps the most fundamental operation of image processing. Gaussian and Median filters tend to blur edges. So, I am going to apply a bilateral filter it can reduce unwanted noise very well while keeping edges fairly sharp. However, it is very slow compared to most filters.
6. Edge Detection, Canny Edge Detection is a popular edge detection algorithm. The first argument is our input image. Second

and third arguments are our lower and upper threshold respectively.

Canny use two thresholds (upper and lower)

- If a pixel gradient is greater than the upper threshold, then pixel is accepted as edge.
- If a pixel gradient is lower than lower value, then it’s rejected.
- If the pixel gradient is in between the two thresholds, then it will be accepted only if it is connected to a pixel that is above the *upper* threshold.

- Contouring:
- **Contours** are defined as the line joining all the points along the boundary of an image that are having the same intensity. Contours are used in shape analysis and object detection. Contour works well with binary images.
- Number plate detection
- Finally, we have contour with large area and its coordinates.
- Loop over our contours to find the best possible contour of license plate.

1.5 System Tools

a. Software Requirements

Kaggle & Colab software

b. Hardware Requirements

Pc – windows operating system

1.6 Results and Discussions

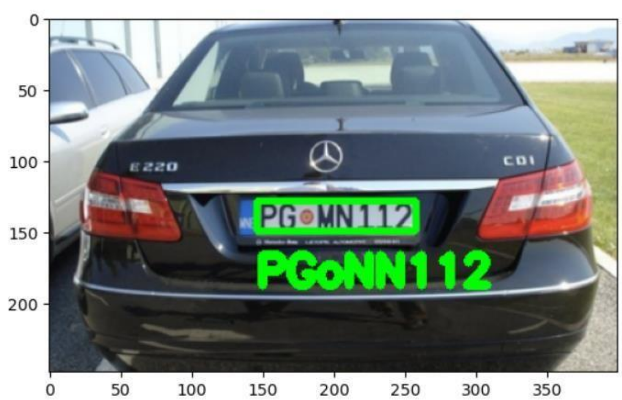
selected-dataset Draft saved

File Edit View Run Add-ons Help

Run All Code Draft Session (12m)

```
text = result[0][1]
font = cv2.FONT_HERSHEY_SIMPLEX
res = cv2.putText(img, text = text, org = (approx[0][0][0], approx[1][0][1]+60), fontFace = font, fontScale = 1, color
res = cv2.rectangle(img, tuple(approx[0][0]), tuple(approx[2][0]), (0,255, 0), 3)
plt.imshow(cv2.cvtColor(res, cv2.COLOR_BGR2RGB))
```

[23]: <matplotlib.image.AxesImage at 0x7de2c9459ed0>



1.7 Conclusions

In this tutorial, you learned how to build a basic Automatic License/Number Plate Recognition system using OpenCV and Python. Our ANPR method relied on basic computer vision and image processing techniques to localize a license plate in an image, including morphological operations, image gradients, thresholding, bitwise operations, and contours. This method will work well in controlled, predictable environments — like when lighting conditions are uniform across input images and license plates are standardized (such as dark characters on a light license plate background).However, if you are developing an ANPR system that *does not* have a controlled environment, you'll need to start inserting machine learning and/or deep learning to replace parts of our plate localization pipeline. HOG + Linear SVM is a good starting point for plate localization if your input license plates have a viewing angle that doesn't change more than a few degrees. If you're working in an unconstrained environment where viewing angles can vary dramatically, then deep learning-based models such as Faster R-CNN, SSDs, and YOLO will likely obtain better accuracy.

1.8 Appendix

```
In [1]:  pip install easyocr

         Collecting easyocr
           Using cached easyocr-1.7.0-py3-none-any.whl (2.9 MB)
         Requirement already satisfied: numpy in c:\users\hp\anaconda3\lib\site-packages (from easyocr) (1.23.5)
         Requirement already satisfied: Pillow in c:\users\hp\anaconda3\lib\site-packages (from easyocr) (9.4.0)
         Collecting python-bidi
           Using cached python_bidi-0.4.2-py2.py3-none-any.whl (30 kB)
         Collecting ninja
           Using cached ninja-1.11.1-py2.py3-none-win_amd64.whl (313 kB)
         Requirement already satisfied: PyYAML in c:\users\hp\anaconda3\lib\site-packages (from easyocr) (6.0)
         Requirement already satisfied: torch in c:\users\hp\anaconda3\lib\site-packages (from easyocr) (1.12.1)
         Collecting opencv-python-headless
           Using cached opencv_python_headless-4.7.0.72-cp37-abi3-win_amd64.whl (38.1 MB)
         Collecting Shapely
           Downloading shapely-2.0.1-cp310-cp310-win_amd64.whl (1.4 MB)
              ---------------------------------------- 1.4/1.4 MB 923.7 kB/s eta 0:00:00
         Collecting pyclipper
           Downloading pyclipper-1.3.0.post4-cp310-cp310-win_amd64.whl (94 kB)
              ---------------------------------------- 94.5/94.5 kB 81.6 kB/s eta 0:00:00
         Requirement already satisfied: scipy in c:\users\hp\anaconda3\lib\site-packages (from easyocr) (1.10.0)
         Collecting torchvision>=0.5
           Downloading torchvision-0.15.2-cp310-cp310-win_amd64.whl (1.2 MB)
              ---------------------------------------- 1.2/1.2 MB 219.3 kB/s eta 0:00:00
         Requirement already satisfied: scikit-image in c:\users\hp\anaconda3\lib\site-packages (from easyocr) (0.19.3)
         Requirement already satisfied: requests in c:\users\hp\anaconda3\lib\site-packages (from torchvision>=0.5->easyocr) (2.28.1)
         Collecting torch

In [7]:  import cv2

In [8]:  import numpy as np
         import pandas as pd
         import matplotlib.pyplot as plt

In [10]: import easyocr

In [15]: img = cv2.imread('test.jpg')

In [21]: img = cv2.imread(r"C:\Users\amb\Downloads\test.jpg")

In [22]: gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

selected-dataset Draft saved

File Edit View Run Add-ons Help

Run All Code Draft Session (15m)

[8]:
```
plt.imshow(cv2.cvtColor(gray, cv2.COLOR_BGR2RGB))
```

[8]: <matplotlib.image.AxesImage at 0x7de2c9bb10f0>

+ Code + Markdown

[9]:
```
bfilter = cv2.bilateralFilter(gray, 11, 11, 17)
```

[9]:
```
bfilter = cv2.bilateralFilter(gray, 11, 11, 17)
```


[10]:
```
edged = cv2.Canny(bfilter, 30, 200)
```

[11]:
```
pip install imutils
```

```
Collecting imutils
  Downloading imutils-0.5.4.tar.gz (17 kB)
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: imutils
  Building wheel for imutils (setup.py) ... done
  Created wheel for imutils: filename=imutils-0.5.4-py3-none-any.whl size=25859 sha256=0cd86d9fc45a6de018a43e425613bf9caa485daaf68e5a1ae14845f
ab6b9bf9b
  Stored in directory: /root/.cache/pip/wheels/85/cf/3a/e265e975a1e7c7e54eb3692d6aa4e2e7d6a3945d29da46f2d7
Successfully built imutils
Installing collected packages: imutils
Successfully installed imutils-0.5.4
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is reco
mmended to use a virtual environment instead: https://pip.pypa.io/warnings/venv class="ansi-yellow-fg">
Note: you may need to restart the kernel to use updated packages.
```

selected-dataset Draft saved

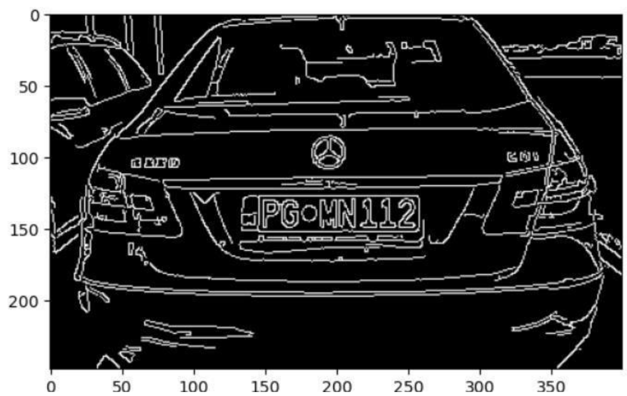
File Edit View Run Add-ons Help

Run All Code Draft Session (18m) HDD CPU RAM

[12]:
```
import imutils
```

[13]:
```
plt.imshow(cv2.cvtColor(edged, cv2.COLOR_BGR2RGB))
```

[13]: <matplotlib.image.AxesImage at 0x7de2c98ed870>



selected-dataset Draft saved

File Edit View Run Add-ons Help

Run All Code Draft Session (19m) HDD CPU RAM

+ Code + Markdown

[14]:
```
keypoints = cv2.findContours(edged.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
```

[15]:
```
contours = imutils.grab_contours(keypoints)
```

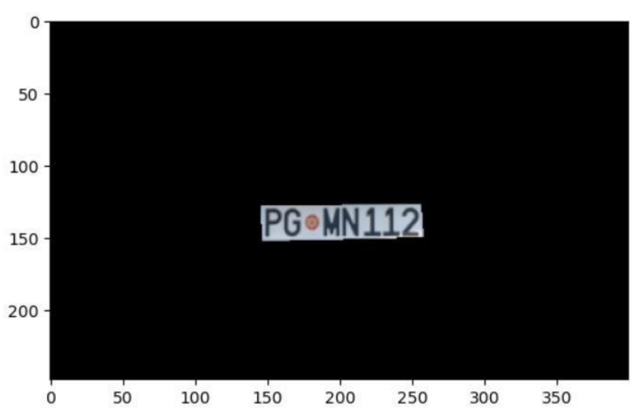
[16]:
```
contours = sorted(contours, key = cv2.contourArea, reverse = True)[:10]
```

[17]:
```
location = None
for contour in contours:
  # cv2.approxPolyDP returns a resampled contour, so this will still return a set of (x, y) points
  approx = cv2.approxPolyDP(contour, 10, True)
  if len(approx) == 4:
    location = approx
    break
```

[18]:

[20]:
```
new_image = cv2.bitwise_and(img, img, mask = mask)
plt.imshow(cv2.cvtColor(new_image, cv2.COLOR_BGR2RGB))
```

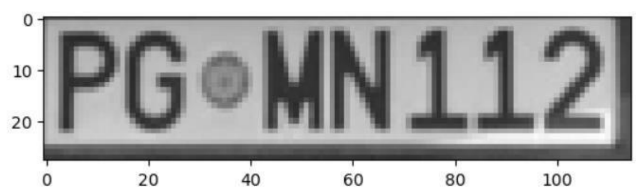
[20]:
```
<matplotlib.image.AxesImage at 0x7de2cbcc5d20>
```



+ Code + Markdown

[21]:
```
(x, y) = np.where(mask == 255)
(x1, y1) = (np.min(x), np.min(y))
(x2, y2) = (np.max(x), np.max(y))
# Adding Buffer
cropped_image = gray[x1:x2+3, y1:y2+3]
plt.imshow(cv2.cvtColor(cropped_image, cv2.COLOR_BGR2RGB))
```

[21]: <matplotlib.image.AxesImage at 0x7de2c93b91e0>



+ Code + Markdown

[17]:
```
location = None
for contour in contours:
  # cv2.approxPolyDP returns a resampled contour, so this will still return a set of (x, y) points
  approx = cv2.approxPolyDP(contour, 10, True)
  if len(approx) == 4:
    location = approx
    break
```

[18]:
```
mask = np.zeros(gray.shape, np.uint8)
```

+ Code + Markdown

[19]:
```
new_image = cv2.drawContours(mask, [location], 0, 255, -1)
```

[20]:
```
new_image = cv2.bitwise_and(img, img, mask = mask)
plt.imshow(cv2.cvtColor(new_image, cv2.COLOR_BGR2RGB))
```

In [31]:
```
contours = sorted(contours, key = cv2.contourArea, reverse = True)[:10]
```

In [32]:
```
location = None
for contour in contours:
  # cv2.approxPolyDP returns a resampled contour, so this will still return a set of (x, y) points
  approx = cv2.approxPolyDP(contour, 10, True)
  if len(approx) == 4:
    location = approx
    break
```

In [33]:
```
mask = np.zeros(gray.shape, np.uint8)
```

In [34]:
```
new_image = cv2.drawContours(mask, [location], 0, 255, -1)
```

In [37]:
```
reader = easyocr.Reader(['en'])
result = reader.readtext(cropped_image)
print(result)
```

```
Neither CUDA nor MPS are available - defaulting to CPU. Note: This module is much faster with a GPU.
Downloading detection model, please wait. This may take several minutes depending upon your network connection.
```

```
Progress: |██████████████████████████████████████████████████| 100.0% Complete
```

```
Downloading recognition model, please wait. This may take several minutes depending upon your network connection.
```

```
Progress: |██████████████████████████████████████████████████| 100.0% Complete[([[36, 0], [244, 0], [244, 54], [36, 54]], 'G
YI2 RZB', 0.9281695706200098)]
```

In [38]:
```
text = result[0][1]
font = cv2.FONT_HERSHEY_SIMPLEX
res = cv2.putText(img, text = text, org = (approx[0][0][0], approx[1][0][1]+60), fontFace = font, fontScale = 1, color = (0,
res = cv2.rectangle(img, tuple(approx[0][0]), tuple(approx[2][0]), (0,255, 0), 3)
plt.imshow(cv2.cvtColor(res, cv2.COLOR_BGR2RGB))
```

```
text = result[0][1]
font = cv2.FONT_HERSHEY_SIMPLEX
res = cv2.putText(img, text = text, org = (approx[0][0][0], approx[1][0][1]+60), fontFace = font, fontScale = 1, color
res = cv2.rectangle(img, tuple(approx[0][0]), tuple(approx[2][0]), (0,255, 0), 3)
plt.imshow(cv2.cvtColor(res, cv2.COLOR_BGR2RGB))
```

[23]: <matplotlib.image.AxesImage at 0x7de2c9459ed0>