Automatic License Plate Recognition using Python and OpenCV

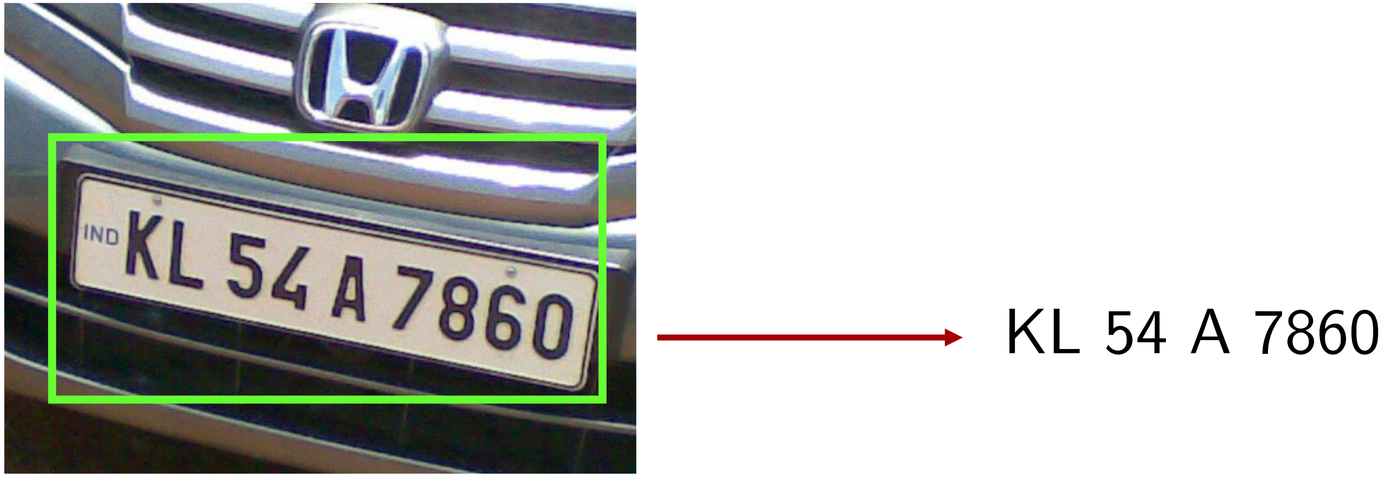
K.M. Sajjad
Department of Computer Science and Engineering
M.E.S. College of Engineering, Kuttippuram, Kerala

Presented by Pongsapas Watana

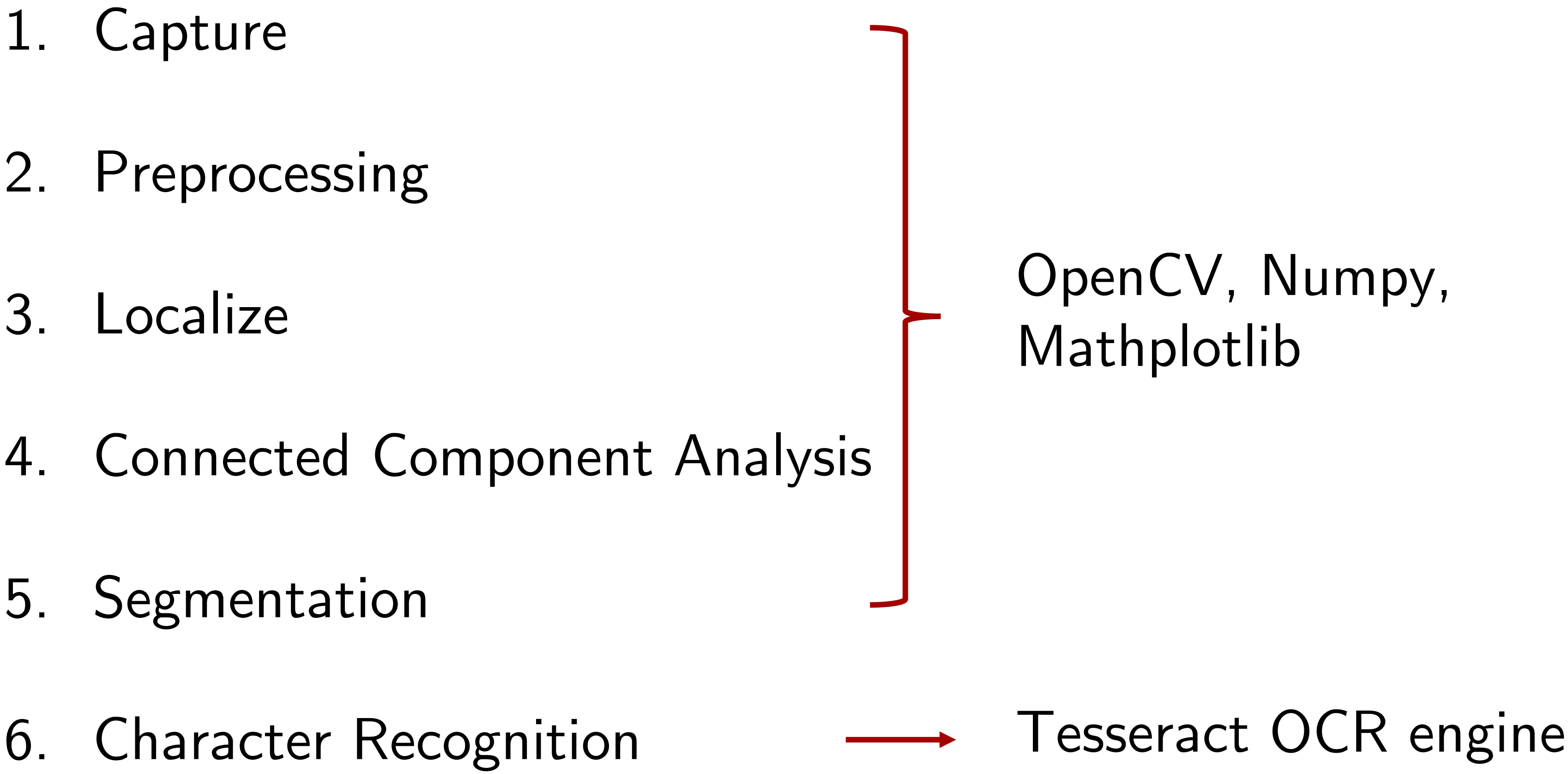
03-Nov-2020

ALPR (Automatic License Plate Recognition)

- ALPR extracts characters on vehicle's license plate
- Main tools for implementation were Python, OpenCV library, and Tesseract library (Optical Character Recognition engine)
- The system in this paper was designed for static images of license plate of vehicles in India



Proposed System



1. Capture
2. Preprocessing
3. Localize
4. Connected Component Analysis
5. Segmentation
6. Character Recognition

OpenCV, Numpy, Mathplotlib

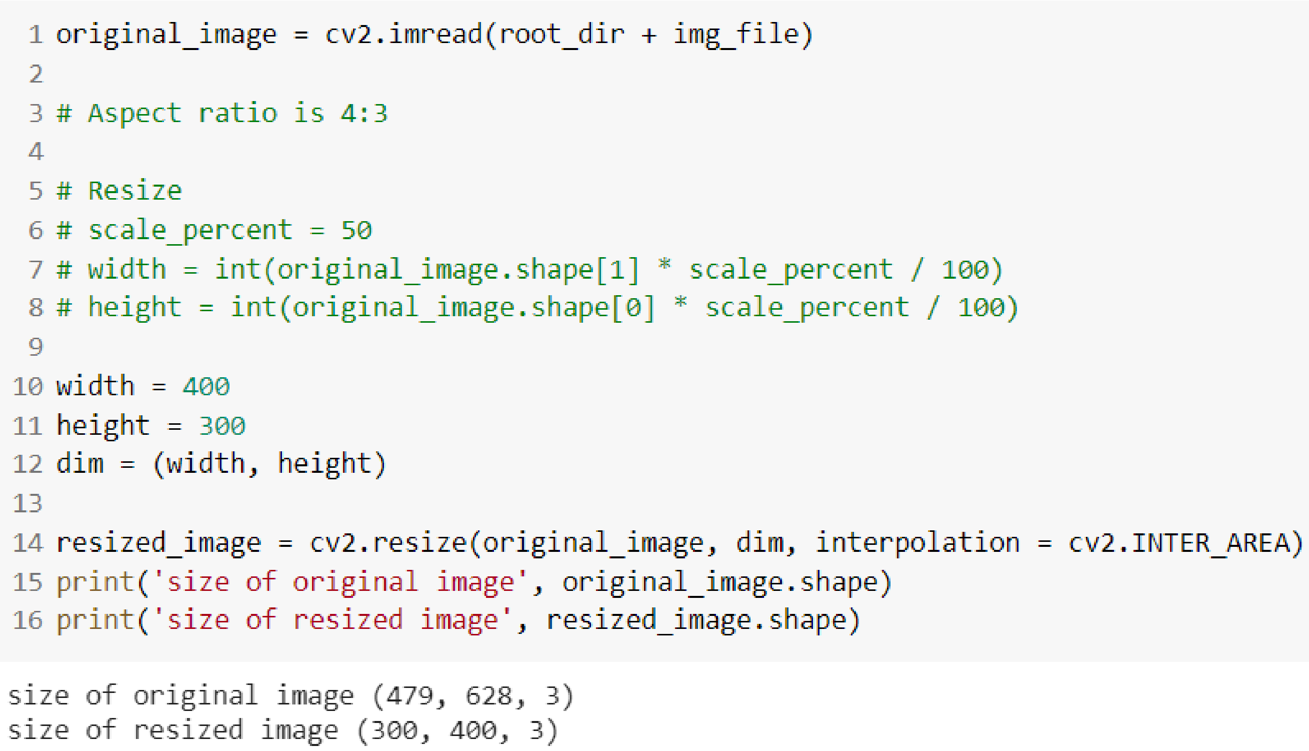
Tesseract OCR engine

Capture

- Images were captured with a high resolution camera
- Proper equipment setup captured image with sharpness and low distortion

Preprocess

- **Resizing** and converting **color space** from BGR to gray enhanced processing speed

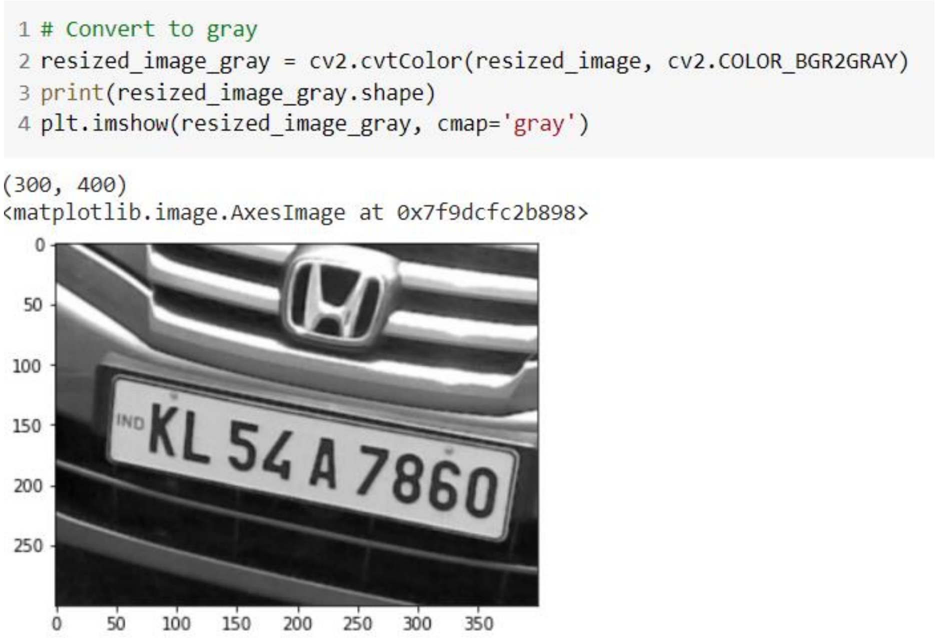


```
original_image = cv2.imread(root_dir + img_file)

# Aspect ratio is 4:3

# Resize
# scale_percent = 50
# width = int(original_image.shape[1] * scale_percent / 100)
# height = int(original_image.shape[0] * scale_percent / 100)

width = 400
height = 300
dim = (width, height)

resized_image = cv2.resize(original_image, dim, interpolation = cv2.INTER_AREA)
print('size of original image', original_image.shape)
print('size of resized image', resized_image.shape)
```

```
size of original image (479, 628, 3)
size of resized image (300, 400, 3)
```

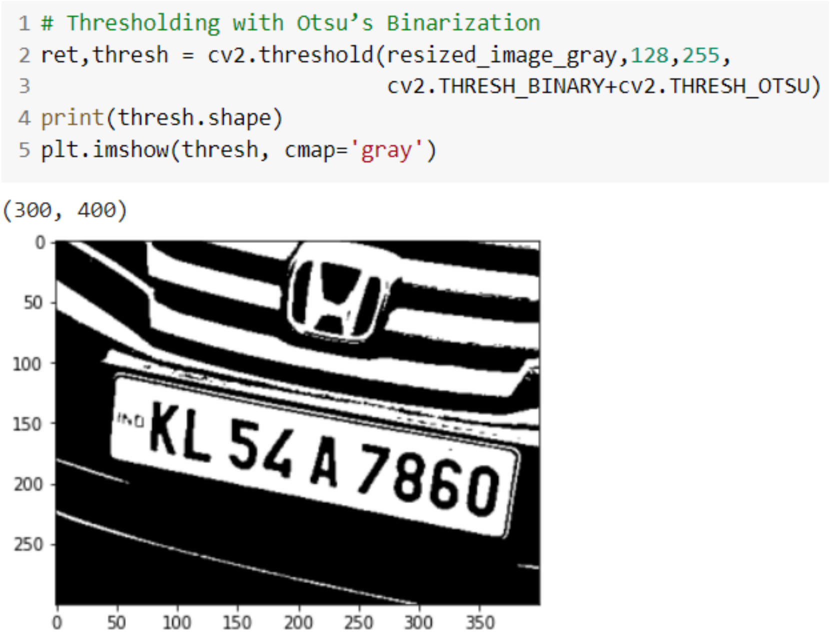
Localize

- Image was converted to gray scale
- **Thresholding** was used to highlight the characters and suppress a background



```python
# Convert to gray
resized_image_gray = cv2.cvtColor(resized_image, cv2.COLOR_BGR2GRAY)
print(resized_image_gray.shape)
plt.imshow(resized_image_gray, cmap='gray')
```

```
(300, 400)
<matplotlib.image.AxesImage at 0x7f9dcfc2b898>
```



```python
# Thresholding with Otsu's Binarization
ret,thresh = cv2.threshold(resized_image_gray,128,255,
                                  cv2.THRESH_BINARY+cv2.THRESH_OTSU)
print(thresh.shape)
plt.imshow(thresh, cmap='gray')
```

```
(300, 400)
```

Connected Component Analysis

- The author used Blob-Detector (cvBlobsLib) for extracting objects in the image

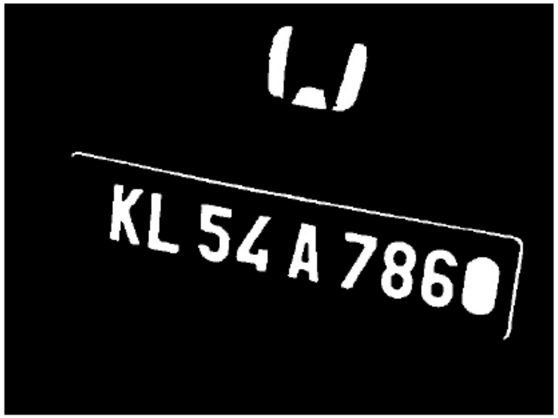


Fig. 4. Connected Components (Blobs)

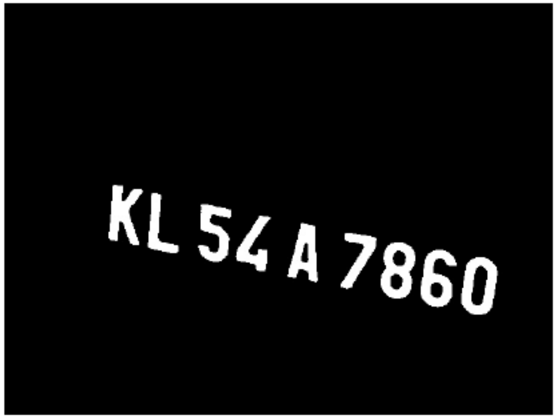
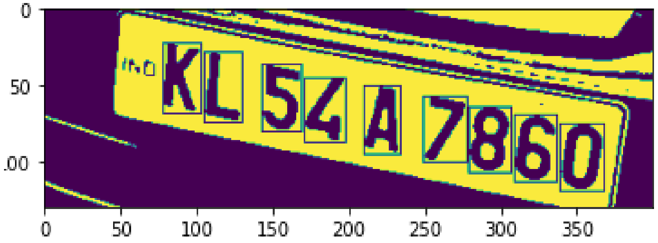


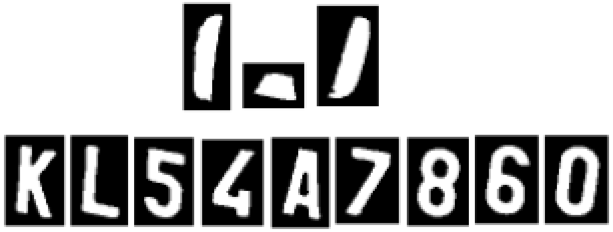
Fig. 6. Classified Blobs

- In the paper reproduction, contouring by cv2.findCountours() was applied instead



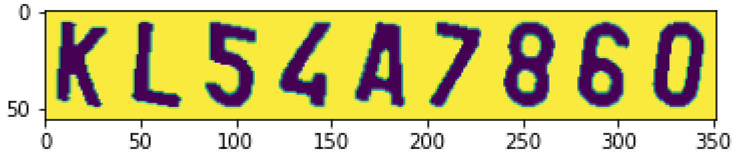
Segmentation

- The author extracted only focus objects from the image with a special algorithm called **Image Scissoring**
- The algorithm scans an image vertically and trims at the column which white pixels don't exist
- Blobs were filtered by two methods
  1) Aspect ratio-based elimination
  2) Pixel coordinate-based elimination

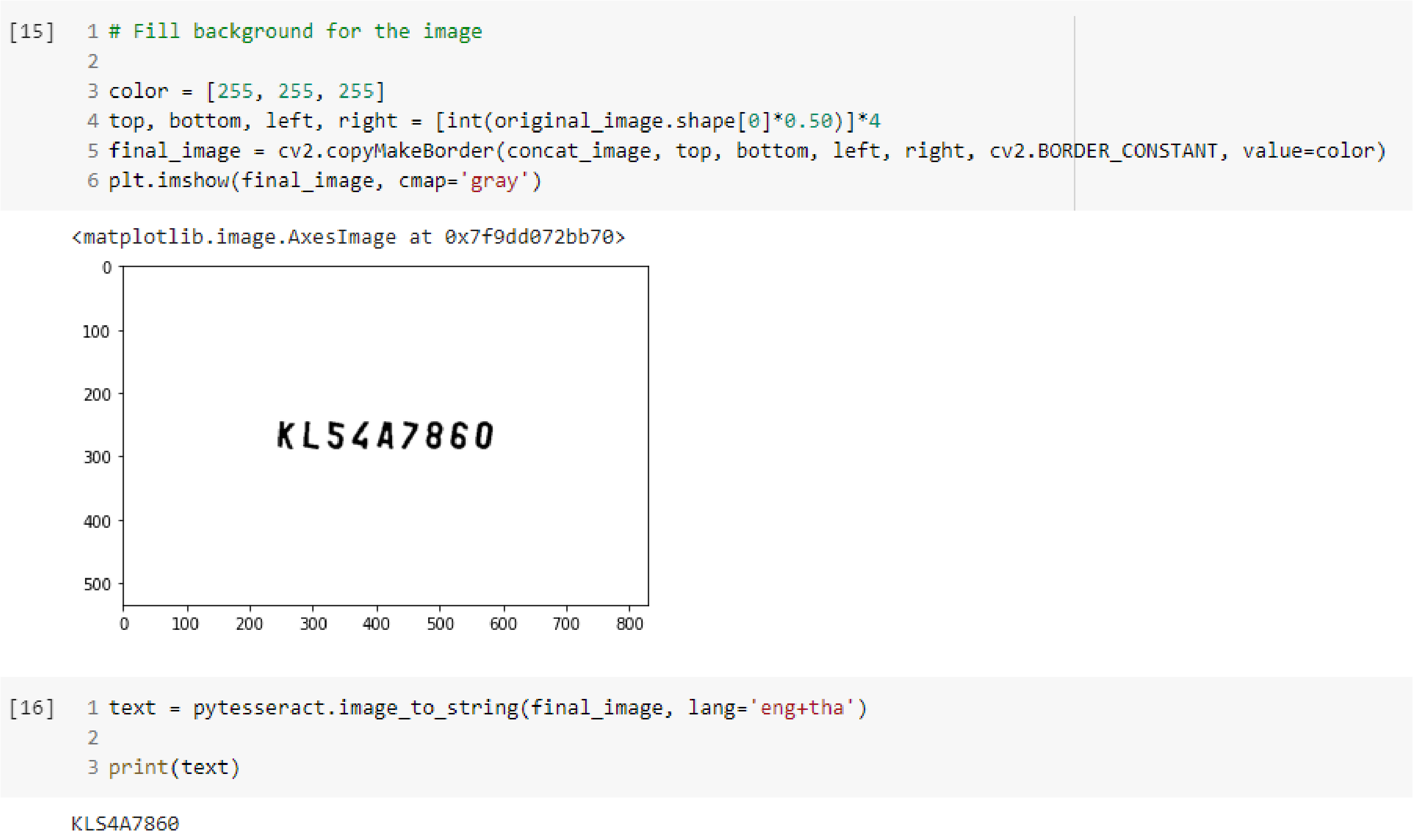


- In the paper reproduction, objects in image were filtered by two methods
  1) Dimension: height > width
  2) Area of sub-images
- Each selected image were resized to the same dimension and patched together

```
<matplotlib.image.AxesImage at 0x7f9dd07d34e0>
```



Character Recognition



```
[15]  # Fill background for the image

      color = [255, 255, 255]
      top, bottom, left, right = [int(original_image.shape[0]*0.50)]*4
      final_image = cv2.copyMakeBorder(concat_image, top, bottom, left, right, cv2.BORDER_CONSTANT, value=color)
      plt.imshow(final_image, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7f9dd072bb70>
```

```
[16]  text = pytesseract.image_to_string(final_image, lang='eng+tha')

      print(text)
```

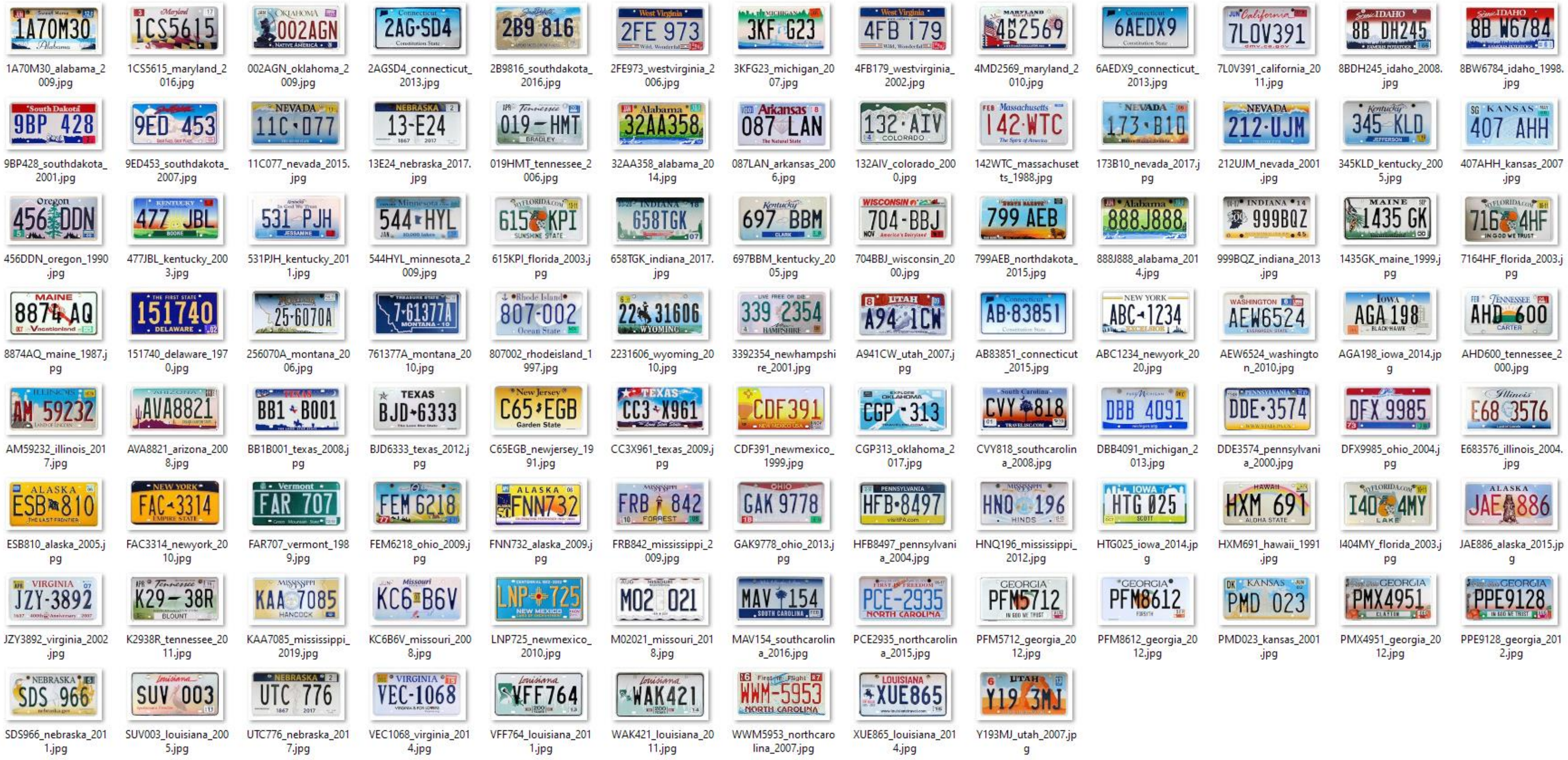
```
KLS4A7860
```

- Big white background was added to enhance readability of Tesseract
- Apply Tesseract OCR engine for character recognition

Original text: KL54A7860, Predicted text: KLS4A7860

Evaluating the model

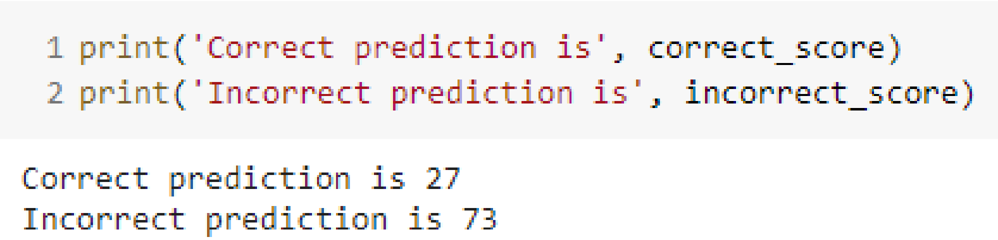
- Acquiring 100 images of vehicle's license plate in India was infeasible
- 100 license plates of vehicle in US was gathered and used for evaluating the model [http://www.worldlicenseplates.com/]



1A70M30_alabama_2009.jpg
1CS5615_maryland_2016.jpg
002AGN_oklahoma_2009.jpg
2AGSD4_connecticut_2013.jpg
2B9816_southdakota_2016.jpg
2FE973_westvirginia_2006.jpg
3KFG23_michigan_2007.jpg
4FB179_westvirginia_2002.jpg
4MD2569_maryland_2010.jpg
6AEDX9_connecticut_2013.jpg
7L0V391_california_2011.jpg
8BDH245_idaho_2008.jpg
8BW6784_idaho_1998.jpg
9BP428_southdakota_2001.jpg
9ED453_southdakota_2007.jpg
11C077_nevada_2015.jpg
13E24_nebraska_2017.jpg
019HMT_tennessee_2006.jpg
32AA358_alabama_2014.jpg
087LAN_arkansas_2006.jpg
132AIV_colorado_2000.jpg
142WTC_massachusetts_1988.jpg
173B10_nevada_2017.jpg
212UJM_nevada_2001.jpg
345KLD_kentucky_2005.jpg
407AHH_kansas_2007.jpg
456DDN_oregon_1990.jpg
477JBL_kentucky_2003.jpg
531PJH_kentucky_2011.jpg
544HYL_minnesota_2009.jpg
615KPI_florida_2003.jpg
658TGK_indiana_2017.jpg
697BBM_kentucky_2005.jpg
704BBJ_wisconsin_2000.jpg
799AEB_northdakota_2015.jpg
888J888_alabama_2014.jpg
999BQZ_indiana_2013.jpg
1435GK_maine_1999.jpg
7164HF_florida_2003.jpg
8874AQ_maine_1987.jpg
151740_delaware_1970.jpg
256070A_montana_2006.jpg
761377A_montana_2010.jpg
807002_rhodeisland_1997.jpg
2231606_wyoming_2010.jpg
3392354_newhampshire_2001.jpg
A941CW_utah_2007.jpg
AB83851_connecticut_2015.jpg
ABC1234_newyork_2020.jpg
AEW6524_washington_2010.jpg
AGA198_iowa_2014.jpg
AHD600_tennessee_2000.jpg
AM59232_illinois_2017.jpg
AVA8821_arizona_2008.jpg
BB1B001_texas_2008.jpg
BJD6333_texas_2012.jpg
C65EGB_newjersey_1991.jpg
CC3X961_texas_2009.jpg
CDF391_newmexico_1999.jpg
CGP313_oklahoma_2017.jpg
CVY818_southcarolina_2008.jpg
DBB4091_michigan_2013.jpg
DDE3574_pennsylvania_2000.jpg
DFX9985_ohio_2004.jpg
E683576_illinois_2004.jpg
ESB810_alaska_2005.jpg
FAC3314_newyork_2010.jpg
FAR707_vermont_1989.jpg
FEM6218_ohio_2009.jpg
FNN732_alaska_2009.jpg
FRB842_mississippi_2009.jpg
GAK9778_ohio_2013.jpg
HFB8497_pennsylvania_2004.jpg
HNQ196_mississippi_2012.jpg
HTG025_iowa_2014.jpg
HXM691_hawaii_1991.jpg
I404MY_florida_2003.jpg
JAE886_alaska_2015.jpg
JZY3892_virginia_2002.jpg
K2938R_tennessee_2011.jpg
KAA7085_mississippi_2019.jpg
KC6B6V_missouri_2008.jpg
LNP725_newmexico_2010.jpg
M02021_missouri_2018.jpg
MAV154_southcarolina_2016.jpg
PCE2935_northcarolina_2015.jpg
PFM5712_georgia_2012.jpg
PFM8612_georgia_2012.jpg
PMD023_kansas_2001.jpg
PMX4951_georgia_2012.jpg
PPE9128_georgia_2012.jpg
SDS966_nebraska_2011.jpg
SUV003_louisiana_2005.jpg
UTC776_nebraska_2017.jpg
VEC1068_virginia_2014.jpg
VFF764_louisiana_2011.jpg
WAK421_louisiana_2011.jpg
WWM5953_northcarolina_2007.jpg
XUE865_louisiana_2014.jpg
Y193MJ_utah_2007.jpg

Evaluating the model

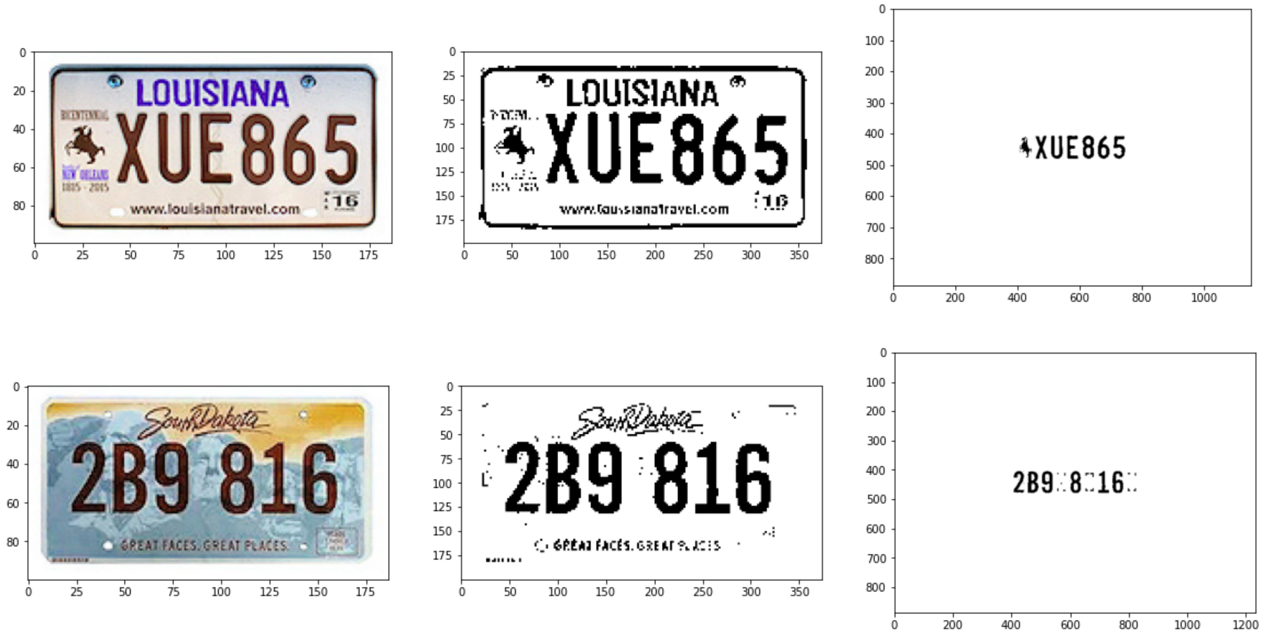
- **Metrics**: prediction is accounted correct only if the predicted text is fully match with the text on license plate
- **27** license plates were correctly predicted
- **73** license plates were incorrectly predicted



```python
print('Correct prediction is', correct_score)
print('Incorrect prediction is', incorrect_score)
```

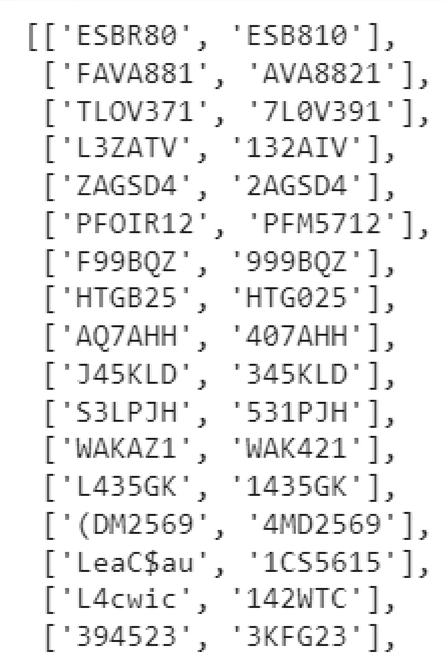
```
Correct prediction is 27
Incorrect prediction is 73
```

Correct prediction

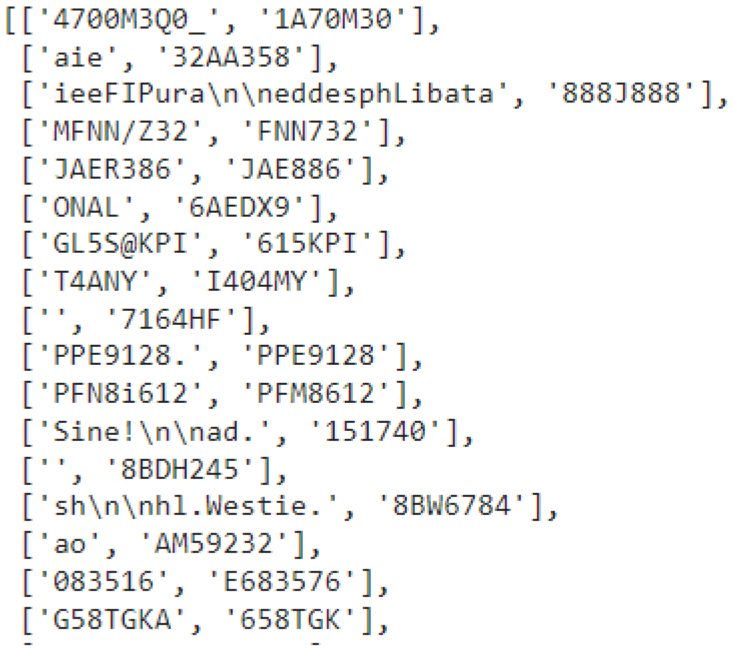


Incorrect prediction

- Factors that influence incorrect predictions were both object localization algorithm and Tesseract engine
- The incorrect results were filtered into two categories
  - Predicted result had a same length as original image
  - Predicted result had a different length as original image

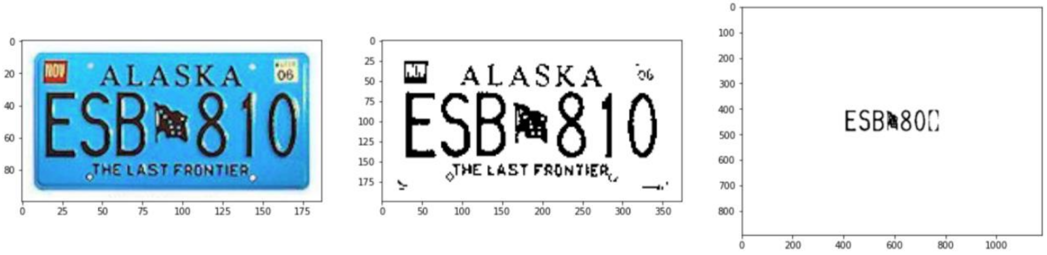


```
[['ESBR80', 'ESB810'],
 ['FAVA881', 'AVA8821'],
 ['TLOV371', '7L0V391'],
 ['L3ZATV', '132AIV'],
 ['ZAGSD4', '2AGSD4'],
 ['PFOIR12', 'PFM5712'],
 ['F99BQZ', '999BQZ'],
 ['HTGB25', 'HTG025'],
 ['AQ7AHH', '407AHH'],
 ['J45KLD', '345KLD'],
 ['S3LPJH', '531PJH'],
 ['WAKAZ1', 'WAK421'],
 ['L435GK', '1435GK'],
 ['(DM2569', '4MD2569'],
 ['LeaC$au', '1CS5615'],
 ['L4cwic', '142WTC'],
 ['394523', '3KFG23'],
```

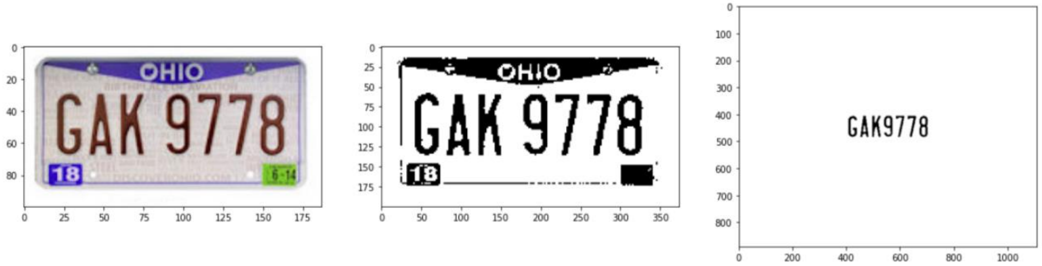


```
[['4700M3Q0_', '1A70M30'],
 ['aie', '32AA358'],
 ['ieeFIPura\n\neddesphLibata', '888J888'],
 ['MFNN/Z32', 'FNN732'],
 ['JAER386', 'JAE886'],
 ['ONAL', '6AEDX9'],
 ['GL5S@KPI', '615KPI'],
 ['T4ANY', 'I404MY'],
 ['', '7164HF'],
 ['PPE9128.', 'PPE9128'],
 ['PFN8i612', 'PFM8612'],
 ['Sine!\n\nad.', '151740'],
 ['', '8BDH245'],
 ['sh\n\nhl.Westie.', '8BW6784'],
 ['ao', 'AM59232'],
 ['083516', 'E683576'],
 ['G58TGKA', '658TGK'],
```

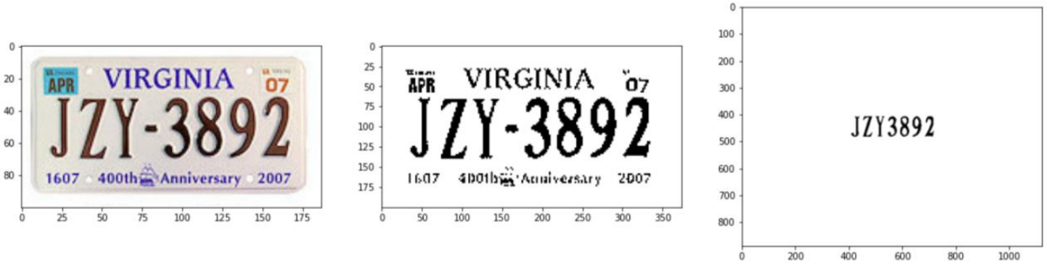
Incorrect prediction, indifferent length



Actual ESB810, **Predicted** ESBR80

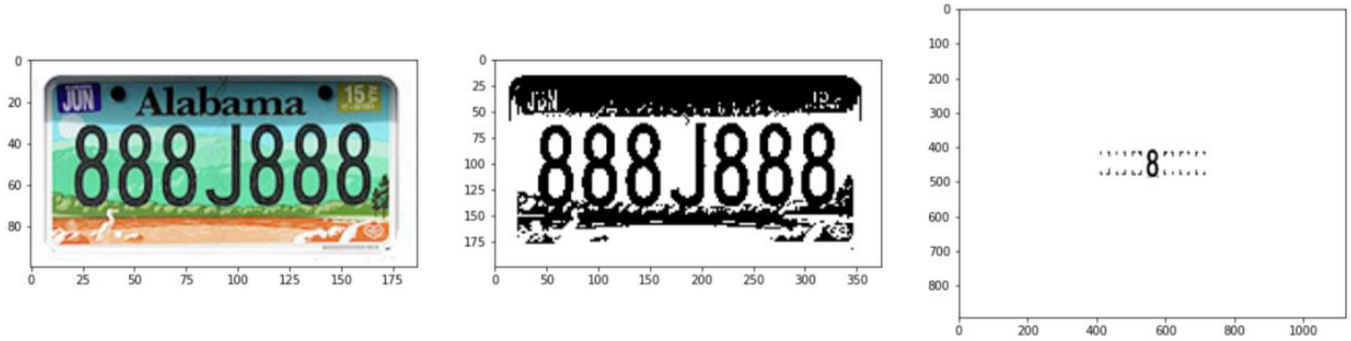


Actual GAK9778, **Predicted** GAKS778

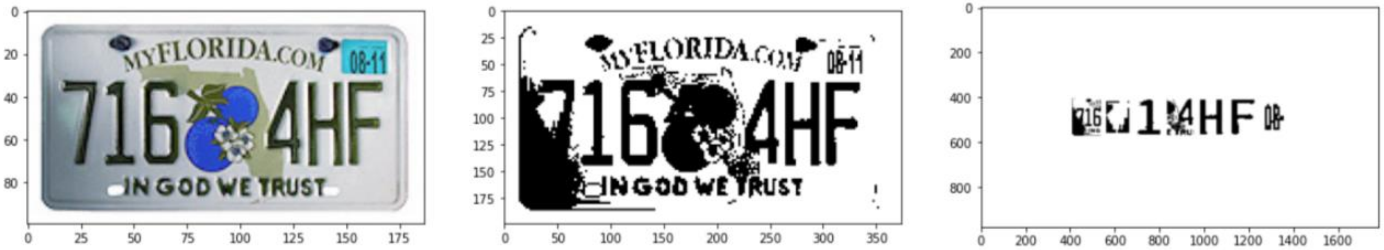


Actual JZY3892, **Predicted** JZ¥3892

Incorrect prediction, different length

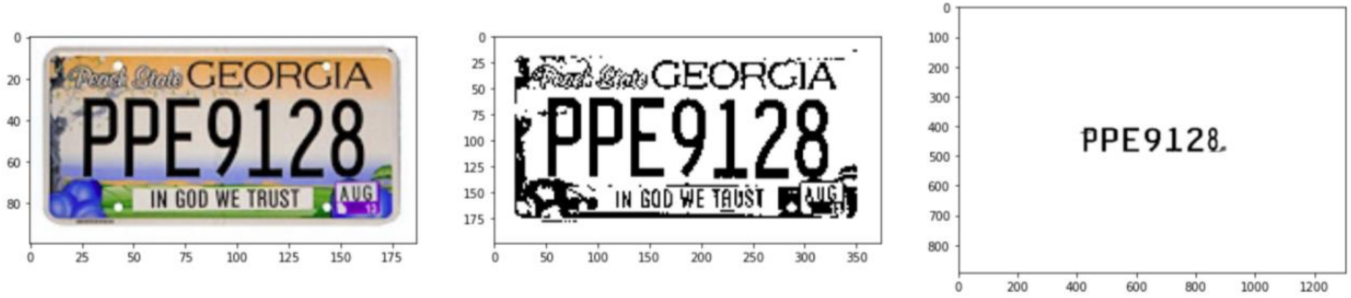


Actual 888J888, **Predicted** ieeFIPura\n\neddesphLibata



Actual 7164HF, **Predicted** *blank*

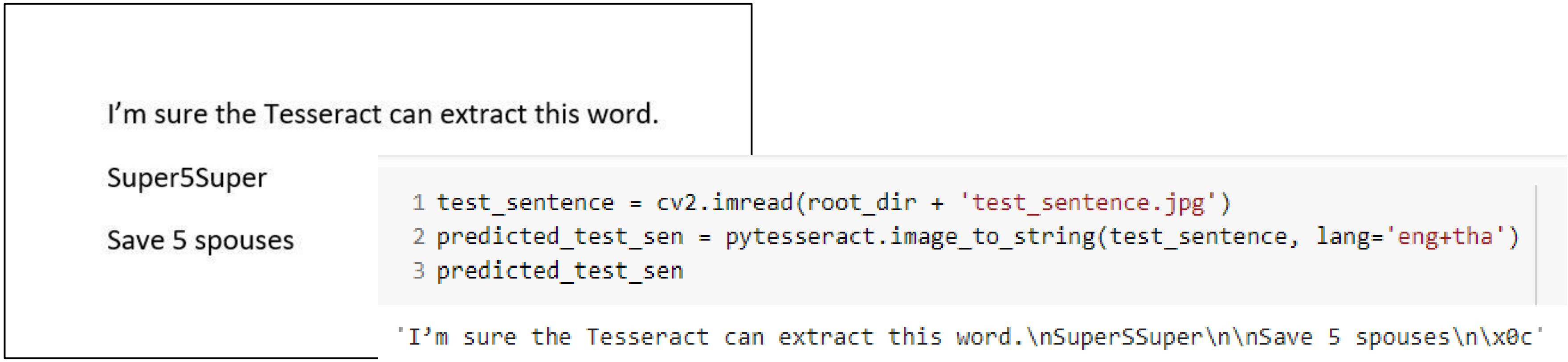
Incorrect prediction, different length



Actual PPE9128, **Predicted** PPE9128.

Conclusions

- The implemented algorithm included contouring, thresholding, segmentation, and Tesseract engine could not universally extract characters from any license plate
- Thresholding is not a proper approach for highlighting the characters on noisy background of image
- Tesseract engine is good in extracting texts from a document. However, it shows a limitation in translating alike characters such as “5” and “S”, especially when the character is not presented as a sentence-like format



I’m sure the Tesseract can extract this word.

Super5Super

Save 5 spouses

```
test_sentence = cv2.imread(root_dir + 'test_sentence.jpg')
predicted_test_sen = pytesseract.image_to_string(test_sentence, lang='eng+tha')
predicted_test_sen
```

```
'I’m sure the Tesseract can extract this word.\nSuperSSuper\n\nSave 5 spouses\n\x0c'
```