UNIVERSITY OF BURGUNDY

SCENE SEGMENTATION AND INTERPRETATION

TEXT RECOGNITION

REPORT

Submitted by

Poorna Naga Avinash Narayana, Yamid Espinel

Supervisor:

Prof. **Desire Sidibe.**

Introduction

The detection and extraction of text regions in an image is a well-known problem in the computer vision research area. The goal of this project is to Detect Text from images, We analysed two commonly used Methodologies in complete text recognition systems: stepwise and integrated. Basic approaches to text localization in natural images: MSER (Maximal Stable Extremal Regions). The algorithm is implemented and evaluated using a set of images of natural scenes that vary along the dimensions of lighting, scale and orientation. We used two different character classifiers to check the performance.

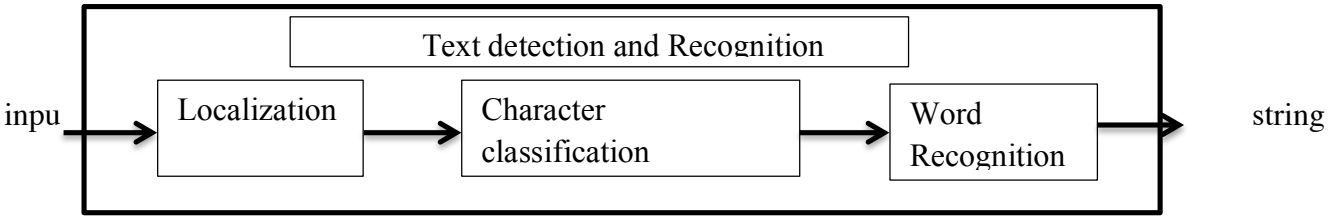


Figure 1 Frame work for commonly used text detection and recognition Methodologies

METHODOLOGY

The methodology typically employ a good strategy, which first localizes the text candidates and then verifies and recognize them.

Localization

The objective of text localization is to localize Text components precisely.

One attractive feature is that most of the background is filtered in the course of localization step, which greatly reduces the computational cost and computational efficiency. The other feature is that processes oriented text as the text orientations are estimated in the Localization step.

For localization step there are several approaches. Our approach is by MSER based text localization [1], done in the following steps:

1. Detect Candidate Text Regions Using MSER

   The MSER algorithm extracts number of co-variant regions from image. MSER is based on the idea of taking regions which stay nearly the same through a wide range of thresholds. All the pixels above or equal to a given threshold are black and all the pixels below a given threshold are white.

2. Remove Non-Text Regions Based On Basic Geometric Properties

MSER detects almost all text regions from image but alongside it also detect some non text regions. To remove those non text regions first we apply Geometric Properties on image. Geometric Properties detects the non text regions from image and remove those regions. The non text regions which are not removed in Geometric Properties for those regions we use the Stroke Width Variation Properties.

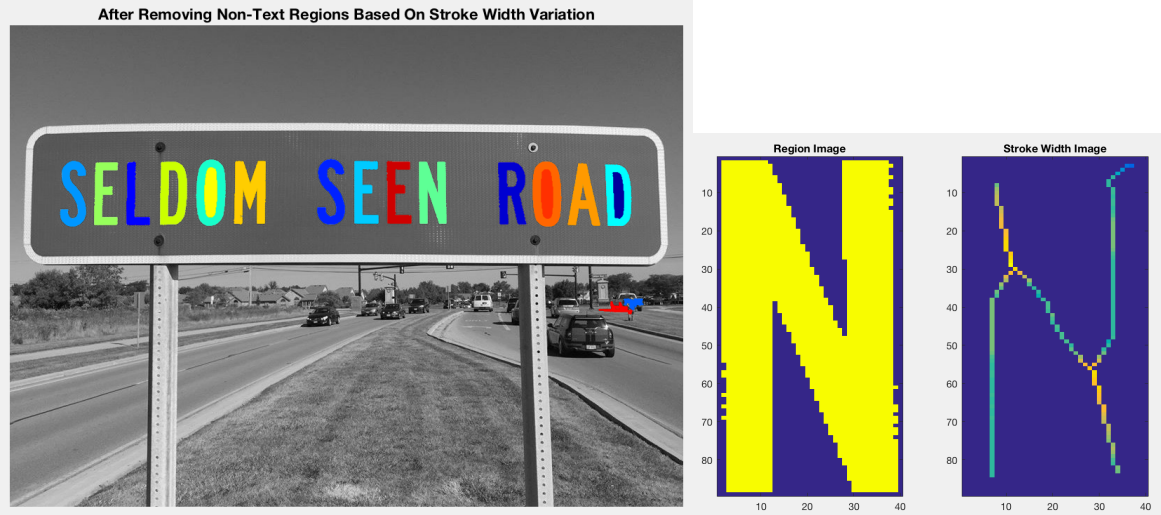
There are several geometric properties that are good for discriminating between text and non-text regions including:

- Aspect ratio
- Eccentricity
- Euler number
- Extent
- Solidity

Use regionprops to measure a few of these properties and then remove regions based on their property values.

3. Remove Non-Text Regions Based On Stroke Width Variation

The Stroke Width Variation Properties are also used to remove the non text regions. Stroke Width is a measure of the curves and lines that make up character. Text regions have little stroke width variation, where as non text regions have larger variations. To remove the non text regions using stroke width we require thresholds. All the pixels above or equal to a given threshold are black and all the pixels below a given threshold are white.



4. Merge Text Regions For Final Detection Result

At this point, the individual detected letters are grouped into words or text lines. The grouping of letters carries more meaningful information than just the individual letter. For example, recognizing the string 'HELP' vs. the set of individual characters {'L','H','P','E'}, where the meaning of the word is lost without the correct ordering.

Character Classification

The character dataset used for training our classifiers is the “Char74K” dataset, proposed by T. de Campos [2]. This dataset has a total of 74,000 characters, divided in digits (0-9), capital letters (A-Z) and small letters (a-z), giving a total of 62 classes to train:

In order to improve the character classification, each picture is preprocessed in the following way:

- *RGB-grayscale conversion*
- *Histogram equalization*
- *Resizing (28x28)*
- *Image binarization*

Deep Neural Network:

A typical convolutional network is composed of stacked feature stages or layers. Multiple convolutional layers are interspersed by pooling or subsampling layers followed by a multiple fully connected layers and finally a single output layer that performs the classification.

As we have discussed in the different layers in CNN, the convolution and subsampling layers perform feature extraction from the input image while the fully connected layer classifies the image. Training of CNN involves the following steps :

a. Initialization: All the filters and weights are initialized to random values.

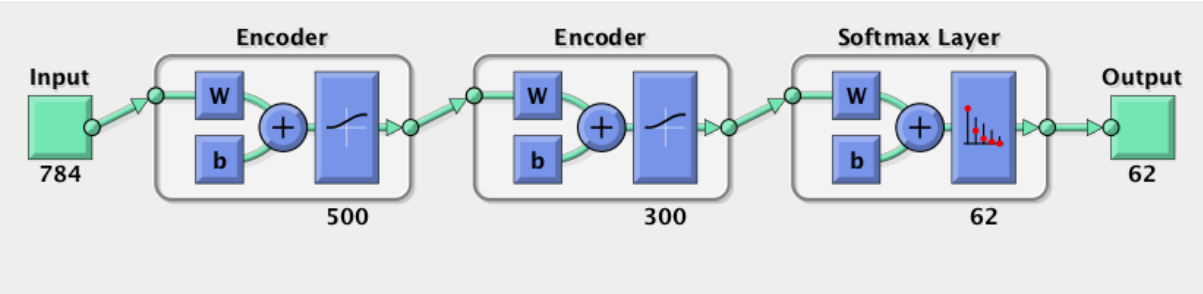
b. Forward Propagation: The convolutional, ReLu, subsampling/pooling layers and the fully connected layers carry on the forward propagation step on the input training image and computes the output probabilities for the different classes.

c. Error Calculation: Summation of the different classes in the output layer can be used to compute the total error.

d. Backpropagation: It is used to calculate the error gradients of the weights and gradient descent is used to update the weights for output error minimization.

e. Iteration: The forward propagation, error calculation, and back propagation steps are repeated for the remaining images in the training set.

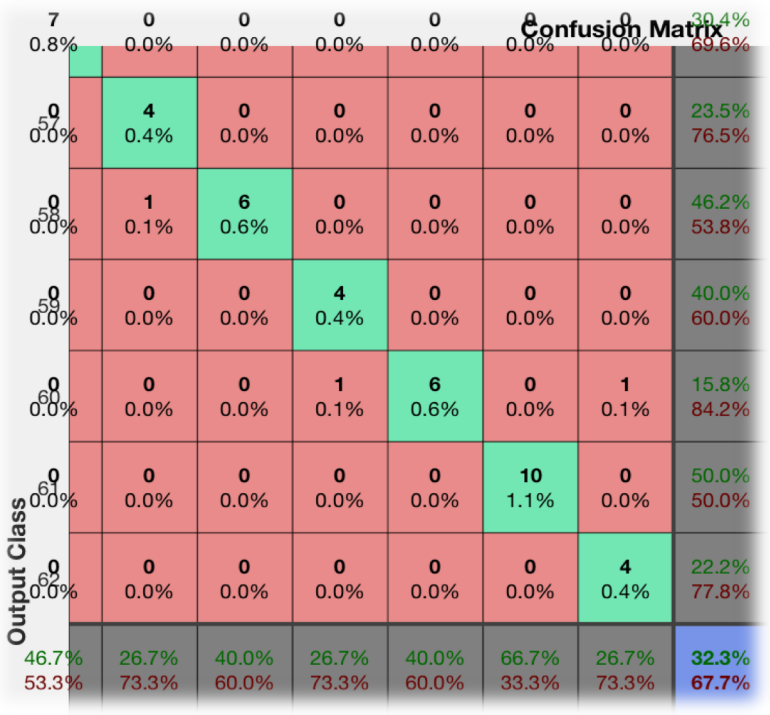
In our case, we use a network with the following structure:



As we can see, we have two hidden layers and one output layer. Each hidden layer will take the most significative features so in the end we are reducing the amount until we end with the 62 character classes.

The training of each layer is done independently, taking the input features for each as the outputs from the training of the previous layer. Once all the layers are trained, we use the outputs of the final classification to re-train the whole network, improving the accuracy in the classification.

In the end, we obtain an accuracy of 33% for our validation dataset; so we move on by trying an SVM classifier.



Support Vector Machine:

Support vector machine (SVM) classifier to segment text from non-text in an image. Initially text is detected in multi scale images. These detected text regions are then verified using HOG (Histogram of oriented gradient) features and SVM.

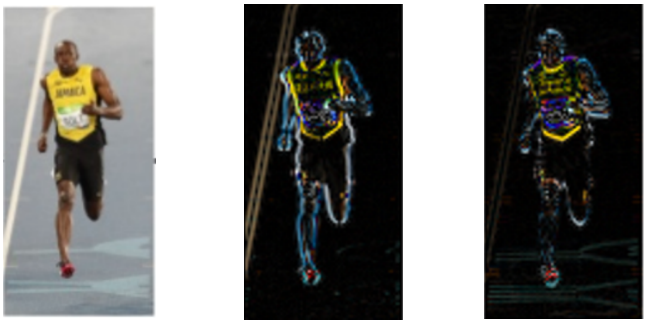
In the HOG feature descriptor, the distribution (histograms) of directions of gradients (oriented gradients) are used as features. Gradients (x and y derivatives) of an image are useful because the magnitude of gradients is large around edges and corners (regions of abrupt intensity changes) and we know that edges and corners pack in a lot more information about object shape than flat regions.

To calculate a HOG descriptor, we need to first calculate the horizontal and vertical gradients; after all, we want to calculate the histogram of gradients. This is easily achieved by filtering the image with the following kernels:

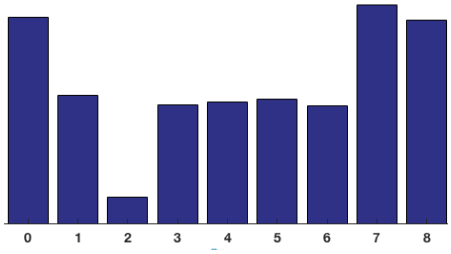
-1	0	1

-1
0
1

The gradient image removed a lot of non-essential information (e.g. constant colored background), but highlighted outlines. In other words, you can look at the gradient image and still easily say there is a person in the picture:



Then, the image is divided into 8×8 cells and a histogram of gradients is calculated for each 8×8 cells:



This set of values obtained from the histogram is what we use as the features for the SVM classifier, giving in the end the possibility of having a total of 62 classes.

Results:

After training and trying the SVM classifier with our test dataset, we obtain an accuracy of 52%, which is better than the one got with our NN approach, but still is not a very good result.

Anyways, we use this classifier to recognize the characters present in the image shown at the beginning. According to the number of the class returned by the classifier for a given character picture, we convert it to its respective string character; an example of the SVM's performance can be seen in the following illustrations:


```
finalstring =

  Columns 1 through 8

    'S'    'E'    'u'    'D'    'O'    '1'    'M'    's'

  Columns 9 through 16

    'E'    'E'    'N'    'R'    'O'    'U'    'A'    'J'

  Columns 17 through 19

    'D'    'B'    'y'
```

As seen in the results, most of the characters (14 out of 15) are correctly classified. However, there are some extra characters in between which come from the bounding boxes that are don't contain any letter. We can say then that there exist the need of adding a text vs. non-text classifier in order to filter out there noisy bounding boxes.

CONCLUSIONS:

MSER combined with other filter methods (such as stroke width variation) let us to segment text easily thanks to its inherent properties (such as color stability and constant stroke width). Even though, there is a need of creating a method that estimates the correct parameters that can let us detect characters in a more precise way, taking into account that they won't work in the same way for any kind of natural image with text inside it.

Even with a not-so-high accuracy (52%), the SVM classifier along with HOG features is able to detect correctly most of the letters enclosed in the bounding boxes. It's then a matter of improving the preprocessing methods and giving a wider training data, increasing even more the classification performance.

The classification stage can take care of the noise that comes from the MSER stage, if it's properly trained with non-text images too. So, there is the need of adding an extra class to the SVM (a non-text class) so that the classifier only return the characters that belong to any of the previous letter classes.