Letter Localization

Markus Köhler
University of Konstanz
Konstanz, Germany
markus.koehler@uni-konstanz.de

***Abstract*—The abstract of this paper.**
***Index Terms*—letter, localization, SVM, HOG, IoU**

I. Introduction

Computer-printed letters can be found everywhere in our cities, e.g. on traffic signs, stores and advertisement posters. The variety of those letters is huge. Although the same letter can appear in different sizes, colors, fonts, text styles and so on, humans are usually very good at locating and classifying them.

A. Goal

This paper is dedicated to the question how to locate these letters on any input image using Machine Learning. We do not care about classifying these letters afterwards. To keep it more simple, we only want to detect the 62 characters 0-9, A-Z, a-z. Although it is true that our detector might also be able to detect some hand-written letters, we restrict our search space to the computer-printed versions of the letters. Another restriction is that we only use rectangular bounding boxes and we do not take into account that a letter may be rotated.

B. Challenges

In spite of those restrictions in I-A, there are still some challenges to handle:

- As mentioned in I, there are many properties of the letters themselves which make them differently.
- The aspect ratio of a letter may also vary, especially, if the perspective of view changes. Then also the letter may also be distorted.
- The 62 letters have very different shapes and aspect ratios. So it might not be possible only to train one model for all letters in order to get good results.
- The background of the bounding boxes containing the letter may also vary or even contain parts of other letters.

So one way to handle those challenges is to find similarities of letters in all varieties which we can use to extract features on it. In the following chapter, we will do this by using HOG features and image segmentation.

II. Data

A. Image selection

The selection of the data consists of 3 parts. The GitHub page "image-text-localization-recognition" [1] was very helpful for image selection.

Firstly, we need positive training examples which contain letters and we also know their positions within the images. It is an advantage if those letters already fulfill most of the properties described in I-B. Otherwise, we need to construct them out of the available examples or we have a feature extraction technique which removes those properties.

We used a part of the data set "Chars74k" [2] of 62992 computer-generated images. Each image has the size 128x128 and contains exactly one character of 0-9, A-Z, a-z with a specific font and may be italic, bold, both or normal. Each character occurs in 1016 images.

Secondly, we also need negative training examples which should not contain any kind of letters. One single letter usually contains a small space in a whole image where we want to locate the letters. So we can use a part of the image as one negative training example which means that we can reuse the same image for different parts of it and we need less initial images if they are large enough.

We used two subsets of the data set "Google Open Images V4" [3] which contains more than 9 million images with a high diversity of shown scenes. One subset contains 100 images which do not contain any letters and the other one contains the first 2627 images.

Thirdly, we need test data which are images which we could also have added to the training examples. There should be easier and more difficult examples such that we may easier see where the evaluated model has its problems.

We used some images of "Google Open Images V4" [3] again and some of "ctw1500" [4]. Additionally, we created a very simple test example just consisting of some randomly distributed letters with white background.

B. HOG features

One technique to get a more abstract representation of the images, which still contains the information about the shape of the letters, is HOG feature extraction. Since the contrast between the border of the images and their background is usually pretty high, this piece of information is also contained in the HOG features.

An advantage of this technique is that we do not have to care about the colors of the images. We also have reduced the dimensions of the data set by a lot.

An issue of this technique is that differences in the fonts of the same letter are still visible in the HOG features. So it is necessary that the data set [2] contains a lot of different fonts.

Also the differences between the letters are still visible. Since we later want to use a SVM with Gaussian kernel, we have to train a separate model for each letter since the differences between the letters are too huge to fit all in one model. Another issue is that we have to use a fixed aspect ratio of a model, but it is no problem for the data set [2] since all images have the same size.

C. Image manipulation

We still are able to do some manipulation of the images before we extract HOG features from them. Since it is very easy to find the most narrow bounding boxes for the letters in data set [2], we can easily create new training examples out of them. We implemented 4 options to do so:

1) plain:
2) crop:
3) resize:
4) locate:

There would have even be the possibility to use a negative example as a background for

III. Models

A. Support Vector Machine (SVM)

B. Cascade Object Detector

IV. Evaluation

A. Non-maximum suppression

B. Intersection over Union (IoU)

V. Framework

References

[1] https://github.com/whitelok/image-text-localization-recognition

[2] T. E. de Campos, B. R. Babu and M. Varma, "Character recognition in natural images", In Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP), Lisbon, Portugal, February 2009. http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/.

[3] https://storage.googleapis.com/openimages/web/download.html

[4] https://github.com/Yuliang-Liu/Curve-Text-Detector