# Scene Text Recognition

Vincent BODIN & Thomas MOREAU

**Abstract**

*Scene text recognition refers to finding automatic ways of extracting text in pictures of every day life. As computer vision is more and more successful, scene text recognition has naturally become a significant issue nowadays. Unlike OCR which is more or less well understood and implemented, scene text recognition still needs many improvement to be considered as a powerful tool. We implemented paper [?] that explains how to extract words efficiently. A graphical model is introduced inside, with the creation of graph represented possible detected words, and algorithm TRW-S [?] is used to extract the optimal one.*

## 1 Introduction

Our task is to extract from a picture the underlying text, and determine what word is being written. It mainly requires two steps:

**Step 1:** We have to detect all the possible characters in the picture. It is very important not to miss one so that leads us to a very flexible criterion of selection.

**Step 2:** Among all those possible characters, try to find those which really are ones, and construct a word from them.

In this project we will essentially be interested in the second step of the project, the first one being implemented in *Object Recognition and Computer Vision* course.

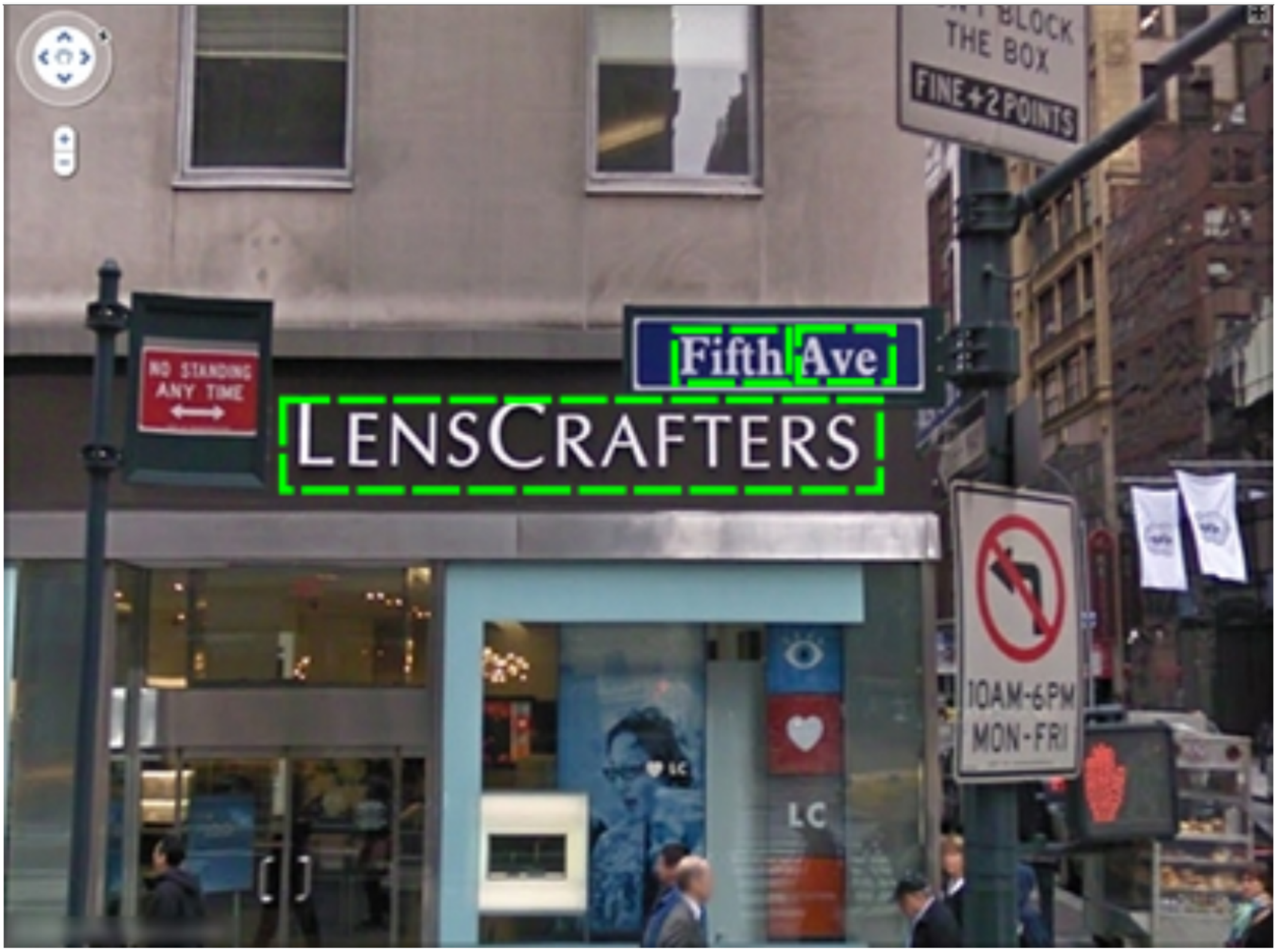

Figure 1: Task

# 2 The graphical model construction

## 2.1 Learning characters and words

**Learning characters.** We need to build classifiers for each character to recognize them in natural pictures.

- Use a database to identify characters: ICDAR 2003 [?], Chars74K [?];
- extract features: Histogram Of Gradient (HOG) [?]. They extract the information of a patch by computing the gradient on bins of sizes $4 \times 4$, saved in an histogram of 12 orientations;
- build $K$ SVMs ($K$ is the number of classes, 62) with RBF Kernel, Fig.(??):

  $$\exp(-\gamma|x - x'|^2), \gamma > 0 \tag{1}$$

  method one-versus-all, we used a Python librairy scikit-learn [?]: two parameters $\gamma$ and regularization $C$ optimized by cross-validation -> test error without optimization 25%, with optimization 16%.

**Learning words.** We have to build a prior lexicon that contains how characters interact to create words, for this:

- Load a database of words [?];
- count for each pair of characters $(c_i, c_j)$ their frequency of occurrence $p(c_i, c_j)$ in the database (bi-gram model);

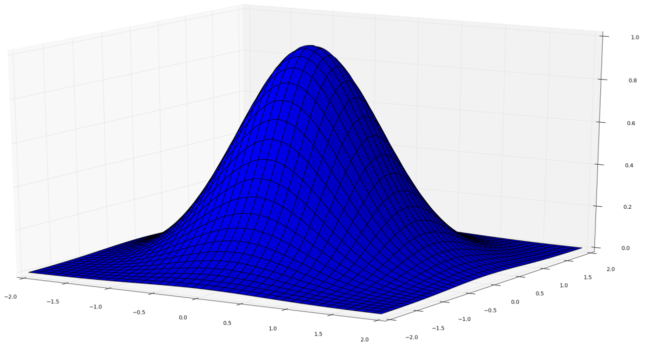

Figure 2: RBF Kernel

This frequency of occurrences will be used to penalize through an energy term the pair of words that are rare in the dictionary (*e.g.* 'xz').

## 2.2 Character detection: sliding windows

The first task is to detect all possible characters in the image, this is done through the sliding windows algorithm:

**Algorithm 1** Sliding Window scanning

```
Input: Image
Output: l list of characters
  l = [ ];
  for all windows in Image do
      $p \leftarrow$ svm.predict_proba
      $\mathrm{GS} \leftarrow \max_c p(c|\phi_i) \exp\left(-\frac{(a_i-\mu_{a_j})^2}{2\sigma_{a_j}^2}\right)$
      if GS $>$ 0.1 then
          l.append(window)
      end if
  end for
```

We did not specify in here the grid we took for the width and the height of the windows: for the moment, to avoid too many detection, we rather tune it by ourselves, giving a range close to the actual size of the letters in the image.

**Algorithm 2** Non-Maximum Suppresion (NMS)

```
Input: $l$ list of windows, $c$ character with maximum probability for each window, threshold
Output: $l_p$ pruned list
  while $l$ is not empty do
      $w_1, c_1$ pair of characters in l with highest probability
      $m = [w]$
      for $w_2, c_2$ in $l, c$ do
          if criterion $>$ threshold and $c_1 == c_2$ then m.append($w_2$)
          end if
      end for
      $l_p$.append(mean($m$),$p$)
  end while
```

# References

[1] C. V. Jawahar A. Mishra, K. Alahari. Top-down and bottom-up cues for scene text recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[2] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Learning (PAMI)*, 2006.

[3] Robust ocr. `http://algoval.essex.ac.uk/icdar/Datasets.html`. Accessed: 2013-12-16.

[4] The chars74k dataset: Character recognition in natural images. `http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/`. Accessed: 2013-12-16.

[5] B. Triggs N. Dalal. Histogram of oriented gradients for human detectection. *CVPR*, 2005.

[6] Scikit-learn. `http://scikit-learn.org/stable/modules/svm.html#parameters-of-the-rbf-kernel`. Accessed: 2013-12-16.

[7] Robust word recognition. `http://algoval.essex.ac.uk/icdar/Datasets.html`. Accessed: 2013-12-16.