SnooperText: A Text Detection System for Automatic Indexing of Urban Scenes

Rodrigo Minetto[1], Nicolas Thome[2], Matthieu Cord[2], Neucimar J. Leite[3] and Jorge Stolfi[3]

[1]*DAINF, Federal University of Technology - Paraná, Curitiba, Brazil*

[2]*Laboratoire d'Informatique Paris 6 (LIP6), Université Pierre et Marie Curie, Paris, France*

[3]*Institute of Computing, University of Campinas, Campinas, Brazil*

Abstract

In this paper, we describe a complete text extraction system for automatic indexing of geo-referenced mosaics of building façades, especially signage in stores and services. The system consists of an original text detector, called SNOOPERTEXT, whose output is fed to the TESSERACT open-source OCR software. SNOOPERTEXT uses a multi-resolution approach to remove irrelevant detail from character shapes and to avoid the use of overly large image processing kernels. At each resolution scale, our system locates candidate characters by using image segmentation and shape descriptor based character/non-character classification. The candidate characters are then grouped to form either candidate words or candidate text lines. These candidate regions are then validated by a text/non-text classifier using a HOG-based descriptor specifically tuned to single-line text regions. We show that SNOOPERTEXT outperforms other published state-of-the-art text detection algorithms on standard image benchmarks. We also describe two metrics to evaluate the end-to-end performance of text extraction systems.

Keywords: Text detection, text classification, histogram of oriented gradients for text, text descriptor, textual indexing in urban scene images.

[3]Corresponding author: Rodrigo Minetto, rodrigo.minetto@gmail.com. This work was partly supported by FAPESP (Phd grant 07/54201-6), CAPES/COFECUB (592/08) and ANR 07-MDCO-007-03.

1. Introduction

The detection of text embedded in images or videos of urban scenes is a challenging problem in computer vision [1] with many potential applications, such as traffic monitoring, geographic information systems, road navigation, and scene understanding. Here we consider one particular application, the iTowns project [2], which aims to build tools for resource location and immersive navigation in urban environments, similar to that of Google's Street View [3]. The main raw data for the iTowns project is a collection of GPS-tagged high-resolution digital photos of the city, including several building façades, taken with a set of car-mounted cameras. The raw images obtained by the car are stitched into geo-referenced mosaics. The mean viewpoint spacing between photo sets is about one meter. See figure 1.

Figure 1: Imaging vehicle and example of an urban scene image captured by the iTowns project.

The frontal images are then processed offline to extract any legible textual information, such as street and traffic signs, store names, and building numbers. The extracted strings are then stored in a geo-referenced database, which is used to answer textual queries by users—for example, to locate the addresses of stores with a specified name or selling a specified product. See figure 2.

The iTowns project could easily generate hundreds of thousands of such mosaics in a

Figure 2: Result of a search for the query string “sushi” through the iTowns user interface. The textual information was extracted with our text detection system.

single city. The manual annotation of all these images with the visible textual information would be very time consuming and probably impractical. Clearly, automated algorithms for this task are highly desirable.

The difficulties in this task mainly come from the diversity of the texts (including extreme text size and font variations, and tilted or curved baselines), the complexity of the backgrounds (including many vaguely text-like objects such as fences, windows, cobblestones, etc) and difficult illumination conditions. OCR algorithms designed for scanned documents perform very poorly on such photos. See figure 3(a). Much better results are obtained by applying an OCR algorithm to the output of a text detector designed specifically for such images, as illustrated in figure 3(b).

In this paper, we describe SNOOPERTEXT, the text detector we developed for the iTowns project. SNOOPERTEXT initially locates candidate characters by using image segmentation and a shape-based character/non-character binary classifier. The candidate characters found

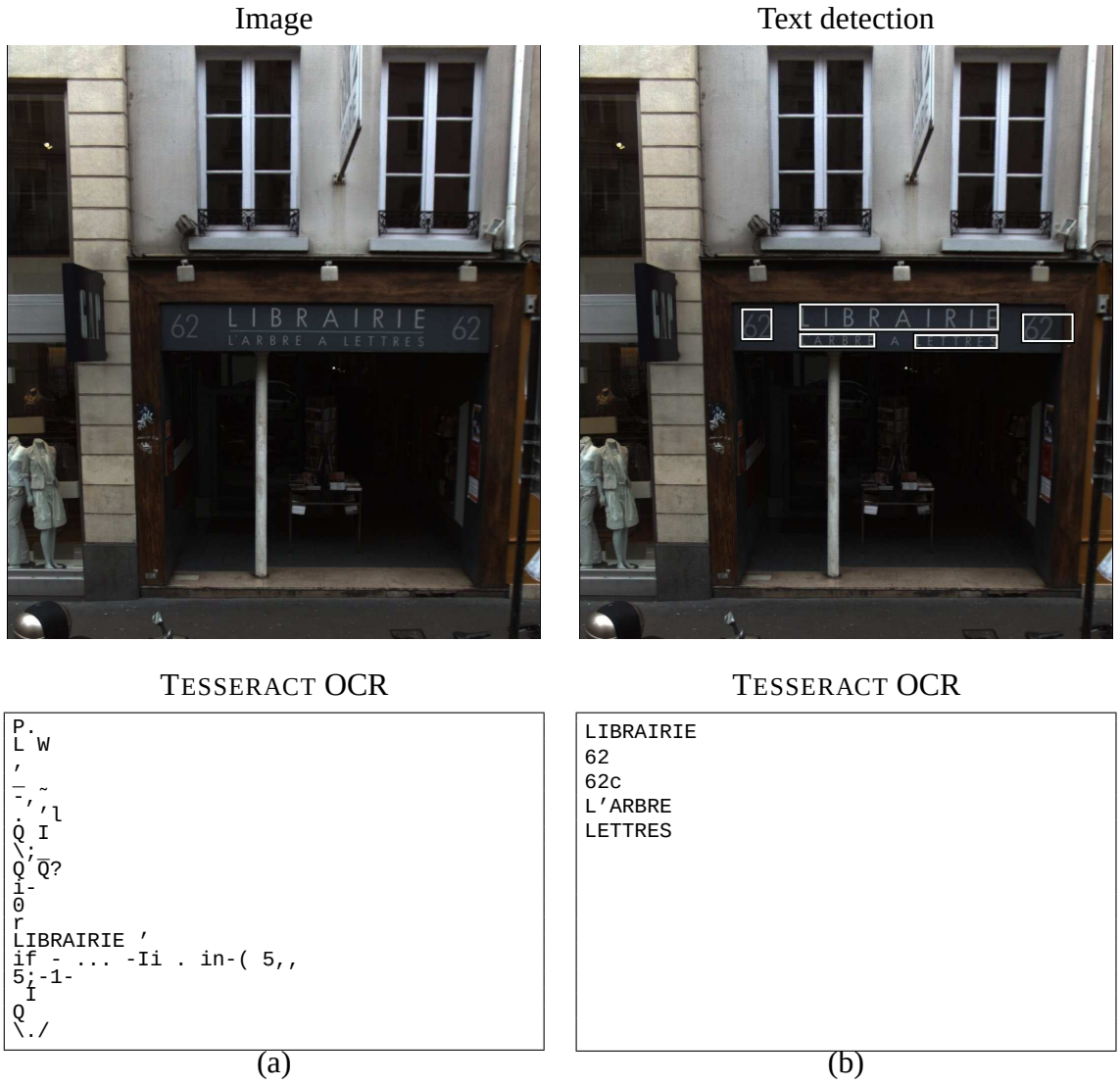

Figure 3: (a) A store-front photo from the iTowns image base and the output of the TESSERACT OCR software applied over the whole image. (b) Text line regions identified by our detector and the output of the back-end of the TESSERACT OCR software applied to those regions.

in this step, represented by their bounding boxes, are then grouped by simple geometric criteria to form either candidate words or candidate text lines. These steps are performed in a multi-scale fashion, in order to efficiently handle widely different character sizes and to

suppress irrelevant texture details inside the characters. Finally, the candidate text regions are validated by a binary text/non-text classifier, that rejects any candidate region that does not seem to contain a single line of text. This classifier uses the T-HOG descriptor [4], which is based on the multi-cell *histogram of oriented gradients* (HOG) of Dalal and Triggs [5]. The regions found by SNOOPERTEXT are then fed to TESSERACT's back-end for OCR processing [6]. See figure 4.

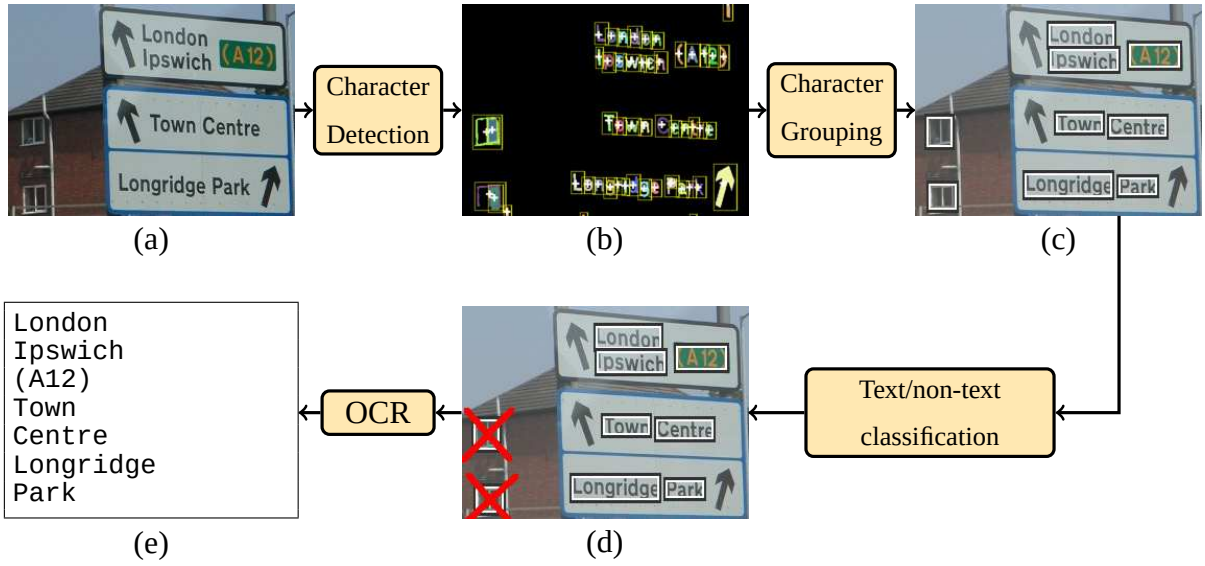

Figure 4: Overall diagram of the SNOOPERTEXT detector (a)–(d) and the OCR step (d)–(e).

Tests show that SNOOPERTEXT is comparable or better than other state-of-the-art text detectors described in the literature [1, 7, 8, 9, 10, 11]. In this paper we also evaluate the end-to-end performance of the complete system (SNOOPERTEXT with TESSERACT's back-end), measured by both approximate and exact string matching of the extracted and ground-truth texts. As a collateral result, we show that we can improve the TESSERACT front-end text detector on street images by using SNOOPERTEXT's T-HOG-based validation module to filter out most of its false positives.

The SNOOPERTEXT algorithm described and tested here is an improvement of a previous work [12] in the use of the T-HOG descriptor for text-region validation and tunning of various internal algorithm parameters. We extended our previous work by describing in detail each part of SNOOPERTEXT (character segmentation, geometric filtering, character grouping, and multiscale processing), reporting its performance in additional databases, and showing its use to improve the word recognition in urban scene images.

This paper is organized as follows. In section 2 we review the literature on text detectors and text/non-text classification, with emphasis on urban photos. The SNOOPERTEXT detector is described in section 3, and its experimental evaluation is reported in sections 4 and 5. Concluding remarks are provided in section 7.

2. Previous work

2.1. Text detection

There is an extensive literature on text detection. The survey of Jung et al. [13] and Liang et al. [14] covers some systems up to 2005. Many approaches for text detection are devoted to specific contexts, such as postal addresses on envelopes [15], cursive handwriting [16], license plates [17], etc. For natural scene processing, more generic systems have been recently considered [1, 7, 8, 18].

Recently, some benchmarks [19, 20, 21] and challenges [22] were organized to give a clear understanding of the current state-of-the-art of natural scene text detection algorithms. Basically, two general approaches for text detection have been proposed: *bottom-up*, consisting of character identification by analyzing the structures that make up text letters, such as edges, textures, colors or connected components, followed by grouping into texts; and *top-down*, which look first for text regions, by exhaustively sampling sub-regions in the original image with a sliding window mechanism, and then splitting those regions into characters.

It is interesting to note that the two leading systems in the 2005 ICDAR challenge [22] rely on different methodologies. The system of Hinnerk Becker [23] (winner of the 2005 ICDAR challenge) is an example of bottom-up solution. It uses an adaptive binarization scheme to extract character regions which are then combined into text lines according to certain geometrical constraints. Alex Chen et al. [23] (second place in the 2005 ICDAR challenge) developed a top-down approach that makes use of a statistical analysis over regions of the frame to identify those that are likely to contain text. Then, they used a cascade of classifiers trained over the chosen features to prune those candidates regions. Finally, those regions are segmented into which are assumed to be text characters.

Most of the text detection systems for natural scenes have been evaluated in the ICDAR dataset since 2005. Seven state-of-the-art systems, either top-down or bottom-up, are briefly discussed in what follows.

In 2007, Mancas-Thillou and Gosselin [18] proposed a top-down color-based approach, by clustering similar color together based on the Euclidean distance and a cosine-based similarity, in the RGB color space, for character segmentation and extraction. The authors did not focus in the text detection part, they assumed that the text regions were previously bounded. They have used intensity and spatial information obtained by Log-Gabor filters to segment characters into individual components. This approach is prone to fail in those texts with similar colors for foreground and background.

In 2010, Epshtein et al. [7] proposed a bottom-up approach known as Stroke Width Transform (SWT) to detect characters in images. They used the pixel gradients orientation over image edges to determine a "local stroke width" and gather pixels with similar stroke widths into regions which are likely to be characters. In addition, the authors provided a new annotated benchmark with urban scene images taken with hand-held cameras [19].

In 2011 and 2012, Chen et al. [8] and Neumann et al. [10], proposed bottom-up methods based on Extremal Regions (ER). Chen et al. used Maximally Stable Extremal Regions

(MSER), which are a subset of ER, for edge-enhancement in order to candidate character detection. The letter candidates were then filtered out using stroke width information computed by the SWT. However, as observed by Neumann et al., MSER detectors have problems with blurry images and characters with low contrast. Therefore, Neumann et al. used all ERs and classified them as being characters or not by developing original features.

Also in 2011, Pan et al. [9] proposed an hybrid multi-scale approach for text detection. They used a sliding window detector, in each scale of the pyramid, composed by a cascade of classifiers trained over HOG features, to initially estimate the text positions. The estimates were then used to enhance the original image in order to help the character segmentation. Then, the authors performed character classification and grouping.

In 2012, Yi et al. [11] and Yao et al. [1] proposed bottom-up methods that uses the stroke width information to character identification. As observed by Yi et al., the letter boundary, used by the SWT, may be broken or connected to a non-text object due to background interference. To avoid this problem, the authors proposed an approach which combines edge pixel clustering and the structural analysis of the stroke boundary with a color assignment procedure. The character grouping was done by considering the geometric properties of nearby characters. Yao et al. proposed an approach to detect texts in arbitrary orientations. The authors used the SWT to character extraction and two layers of filters based on geometric and statistical properties, as well as, a classifier trained with scale and rotation invariant features to reject non-text characters found by the SWT. The character grouping was done by considering the stroke properties, geometric and color features of nearby characters. A greedy hierarchical agglomerative clustering method was also applied to aggregate characters pairs into candidate chains.

2.2. Text classification

Comparatively little has been published about text/non-text *classification* algorithms, although they are often present as post-filters in many text detectors.

Text classification is often cast as a texture classification problem, and several texture descriptors have been considered in the literature. For instance, in 2004, Kim et al [24] described a text recognizer that decomposes the candidate sub-image into a multiscale 16×16 cell grid and compute wavelet moments for each block. Then each block is classified as text or not using an SVM. The ratio of text to non-text outcomes is used to decide if the entire sub-region is text or non-text. In 2005, Ye et al. [25] described a similar text recognizer with multiscale wavelet decomposition but they used more elaborate features including moments, energy, entropy, etc. In 2004, Chen and Yuille [26] proposed a descriptor that combines several features, including 2D histograms of image intensity and gradient, computed separately for the top, middle and bottom of the text region, as well as for more complex slices subdivisions of the image—89 features in total.

Other text detectors, such as the one described by Anthimopoulos et al. in 2010, have used descriptors based on multiscale *local binary patterns* (LBP) introduced by Ojala et al. [27]. Their descriptor has 256 features.

In 2012, Yi et al. [11] proposed a text line descriptor that combines the Gabor filter with gradient and stroke information. They used the block patterns proposed by Chen and Yuille. Their descriptor has 98 features.

The use of gradient orientation histograms (HOGs) as texture descriptors was introduced by Dalal and Triggs in 2005 [5], for human recognition. HOG descriptors are used in some recent text recognizers, such as the one proposed in 2008 by Pan et al. [28]. They partition the candidate sub-image into 14 cells, as proposed by Chen and Yuille, but compute for each cell a 4-bin HOG complemented by a 2×3 array of LBP features. Their complete descriptor has 140 features.

Other HOG-based text recognizers have been proposed in 2009 by Hanif and Prevost [29] for single-line text, and Wang et al. [30] for isolated Chinese and Roman characters as well as single-line text. Hanif and Prevost's descriptor has 151 features (16 cells each with a 8-bin HOG, supplemented by 7 mean difference and 6 standard deviation features over 16 cells). The descriptor of Wang et al. has 80 features (8 cells each with 10 features: a 8-bin HOG; 1 mean difference; and 1 standard deviation).

These HOG-based text recognizers, and several others, use vertical cuts as well as horizontal ones when partitioning the candidate region. The use of vertical cuts was apparently borrowed from the Dalal and Triggs paper [5] on pedestrian recognition. They may be justifiable for isolated characters, but they do not appear to be useful for multi-character texts of variable width. In such texts, the gradient distribution is largely independent of horizontal position; therefore, a cell layout with vertical cuts increases the size of the descriptor without providing any additional relevant information.

3. Text detection and classification

As shown in figure 4, the SNOOPERTEXT detector consists of three main modules: character detection, letter grouping, and text region validation.

3.1. Character detection

The structure of SNOOPERTEXT's character detection module is outlined in figure 5. It consists of three stages: foreground/background image segmentation, geometric filtering and letter/non-letter classification. They are described in sub-sections 3.1.1–3.1.3. These steps are applied in multi-scale fashion, as described in section 3.3.

3.1.1. Image segmentation

The segmentation algorithm used in SNOOPERTEXT was developed by Fabrizio et al. [31]. It is a modified version of Serra's *toggle mapping* [32], a morphological operator for local

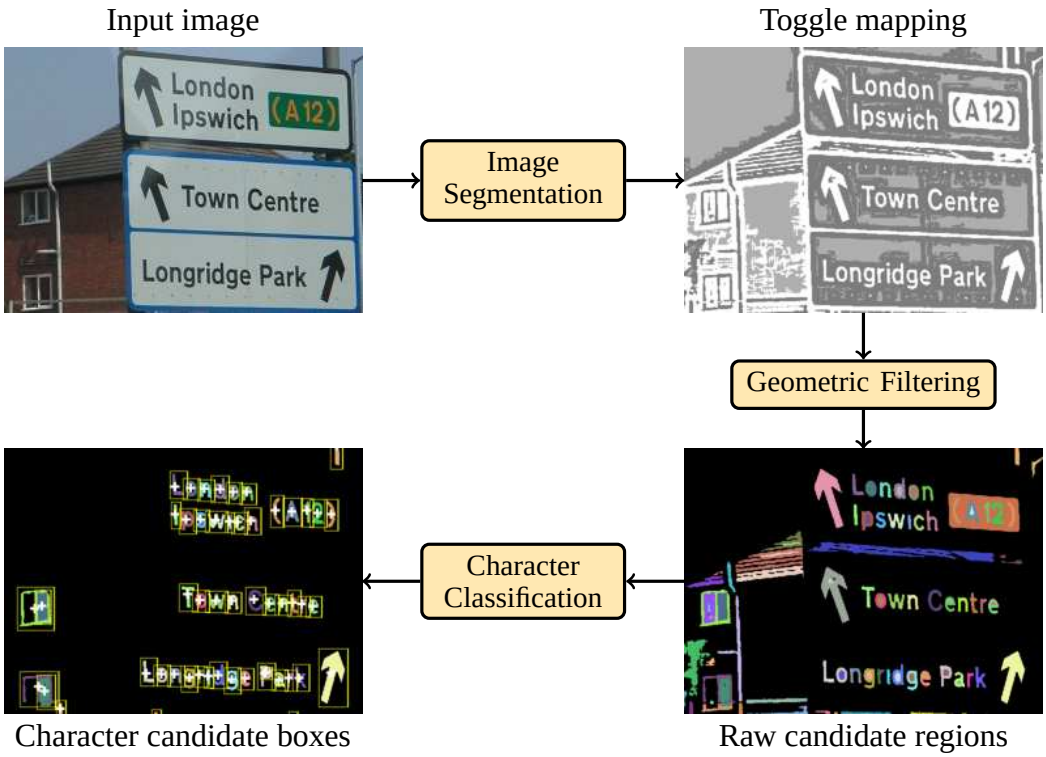

Figure 5: The letter detection stage of SNOOPERTEXT.

contrast enhancement and thresholding, using morphological erosions and dilations [33] to define the local foreground and background levels.

Specifically, in order to segment the original input image $\mathbb{I}$, we first compute a local background image $\mathbb{B}$ by grayscale erosion (neighborhood minimum) and a local foreground image $\mathbb{F}$ by grayscale dilation (neighborhood maximum), using a 9×9 squared structuring element. Note that $\mathbb{B}(x, y) \leq \mathbb{I}(x, y) \leq \mathbb{F}(x, y)$ for every pixel x, y. Then each pixel $\mathbb{I}(x, y)$ is mapped to a ternary class value $\mathbb{D}(x, y) \in \{0, 1, 2\}$ as follows. If $|\mathbb{F}(x, y) - \mathbb{B}(x, y)|$ is less than a fixed threshold $c_{\min}$, then $\mathbb{D}(x, y)$ is set to 1 (indeterminate). Otherwise, $\mathbb{D}(x, y)$ is set to 0 (presumed background) or 2 (presumed foreground) depending on whether the relative brightness $|\mathbb{I}(x, y) - \mathbb{B}(x, y)|/|\mathbb{F}(x, y) - \mathbb{B}(x, y)|$ is less than or greater than another threshold

t.

Since the thresholding is not symmetrical between dark and light regions, and target scenes often have light text on dark background, the segmentation is repeated on the negative (pixel-wise complemented) image. See figure 6.

(a) (b)

Figure 6: (a) Toggle segmentation of the positive image into background (dark gray), foreground (white) and indeterminate (light gray) pixels. (b) Toggle segmentation of the negative image.

3.1.2. Geometric filtering

The foreground regions (letter candidates) from both segmentations (positive and negative) are filtered by simple geometric criteria, based on the area A, width w, and height h of their bounding boxes (minimal axis-aligned enclosing rectangles). Namely, a letter candidate region is accepted iff

$$\begin{aligned} A_{\min} \leq \quad A \quad &\leq A_{\max} \\ w_{\min} \leq \quad w \quad &\leq w_{\max} \\ h_{\min} \leq \quad h \quad &\leq h_{\max} \end{aligned}$$

where $A_{\min}$, $A_{\max}$, $w_{\min}$, $w_{\max}$, $h_{\min}$ and $h_{\max}$, are internal parameters of SNOOPERTEXT and depends on the size range of the letters. See figure 7.

(a) (b)

Figure 7: (a) The regions found by segmentation; (b) The surviving regions after the geometric filtering.

3.1.3. Letter/non-letter classification

For each box that passes the geometric criteria, the algorithm extracts from the corresponding segmented region three scale- and rotation-invariant shape descriptors: Fourier moments, pseudo-Zernike moments, and an original polar encoding [31]. These descriptors are fed to three separate SVM classifiers, whose numeric outputs are packed as a three-element vector and fed to a final SVM classifier [34]. The output of the final SVM is then thresholded to yield a binary letter/non-letter decision. See figure 8.

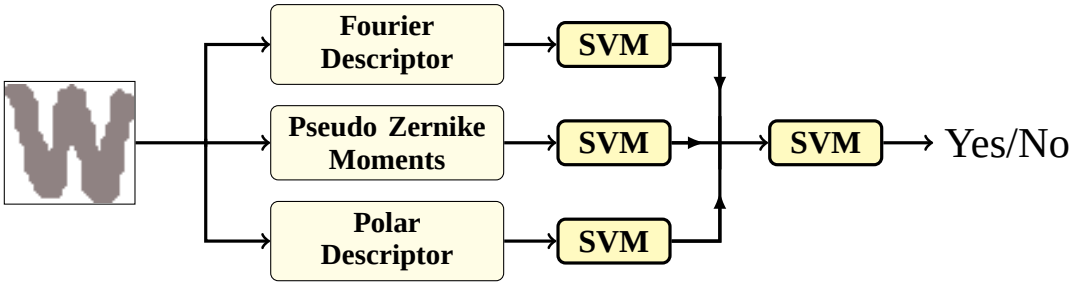

Figure 8: The letter/non-letter classifier of the character detection module.

These SVM were trained on a dataset of bi-level images selected by hand from the output of the segmentation and geometric filtering phases, comprising 16200 instances of uppercase and lowercase letters (positive samples) and 16200 randomly chosen non-letter segments (negative samples). See figure 9.

3.2. Letter grouping

SNOOPERTEXT's character grouping module joins the candidate letters found by the char-

Figure 9: Some of the positive and negative segmented shapes used to train the letter/non-letter classifier.

acter detector into text regions — which may be either words or text lines — according to geometric criteria defined by Retornaz and Marcotegui [35]. These criteria take into account the heights h_1, h_2 and widths w_1, w_2 of the two bounding boxes, as well as the coordinates (x_1, y_1) and (x_2, y_2) of their centers. See figure 10.

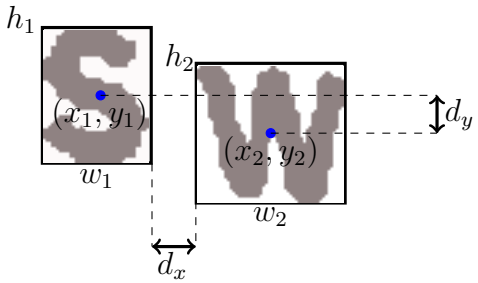

Figure 10: Geometric parameters used for letter grouping.

Specifically, let $h = \min(h_1, h_2)$, $d_x = |x_1 - x_2| - (w_1 + w_2)/2$ and $d_y = |y_1 - y_2|$. Note that d_x is negative iff the two boxes overlap in the x direction. Then the two boxes are said to

be *compatible* — that is, assumed to belong to the same text word or line — iff

$$\begin{aligned} |h_1 - h_2| &< t_1\ h \\ d_x &< t_2\ h \\ d_y &< t_3\ h \end{aligned}$$

where t_1, t_2 and t_3 are parameters of the module. The parameter t_2, in particular, determines whether the groups will be words or text lines, while t_3 controls whether multiple lines are to be merged into text blocks.

These criteria are applied to all pairs of detected characters. The groups are the equivalence classes of the transitive closure of this compatibility relation.

At this stage, letters that were not joined to any group are discarded. This requirement normally eliminates a large fraction of the false positives (non-letter regions classified as letters by the previous steps). Each group is then summarized by a single axis-aligned rectangle, which is the bounding box of its component letters. See figure 11.

Figure 11: Grouping letters into text words.

The grouping module is applied separately to the candidates found in each segmentation (positive and negative). Then the two lists of candidate text regions are merged, and any two regions that have significant overlap (70%) are fused into a single region.

3.3. Multi-scale processing

The segmentation and character/non-character recognition phases perform rather poorly if used at a single scale. Text embedded in photos of urban scene may have characters of

widely different sizes and styles. Characters that are much larger than the structuring element used in the morphological thresholding are often over-segmented. To overcome this problem, those two steps described above are applied in a multi-scale fashion [36]. At each resolution level, the segmentation is applied to a reduced version of the input image, with the goal of detecting characters of a limited size range, and automatically ignoring small irrelevant details of character shape and texture.

More precisely, for each image $\mathbb{I}$, SNOOPERTEXT first builds a multi-scale *image pyramid* $\mathbb{I}^{(0)}, \mathbb{I}^{(1)}, \ldots, \mathbb{I}^{(m)}$. The base $\mathbb{I}^{(0)}$ of the pyramid is the original image $\mathbb{I}$, and each level $\mathbb{I}^{(l)}$ is a copy of the next lower one $\mathbb{I}^{(l-1)}$, reduced to half its width and half its height (so that level $\mathbb{I}^{(l)}$ has $1/4^l$ as many pixels as level $\mathbb{I}^{(0)}$). The maximum level m depends on the size of the original image and the minimum size of the characters to be detected.

The segmentation, character/non-character classification, and grouping steps are applied separately to each level of the pyramid. At each level, the algorithm only looks for letters whose areas lie in the range $[A_{\min} \ \ldots \ A_{\max}]$, defined in section 3.1.2, which corresponds to the size range $[4^l A_{\min} \ \ldots \ 4^l A_{\max}]$ in the original image. The parameters $A_{\min}$ and $A_{\max}$ are chosen so that there is some overlap between two consecutive scales l and $l+1$, namely $A_{\max} > 4A_{\min}$. Similar considerations apply to the linear patameter $h_{\min}$, $h_{\max}$, $w_{\min}$, and $w_{\max}$. Segmented regions whose area fall outside the interval $[A_{\min} \ \ldots \ A_{\max}]$ are ignored, since they are expected to be found at other scales. See figure 12. One advantage of the multi-scale approach is that we can use use a structuring element of fixed (and modest) size in each morphological operation, with significant speed gains. Note that the cost of processing the whole image pyramid, for letters of any size, is only $\sum_{i=0}^{l} 1/4^i \approx 4/3$ times the cost of processing the original image for letters within the fixed ranges of section 3.1.2.

Another advantage of the multi-scale approach is that it makes the segmentation algorithm insensitive to letter texture — high frequency details that are much smaller than the letters themselves. Those details may cause each letter to be split into several separate segments, and

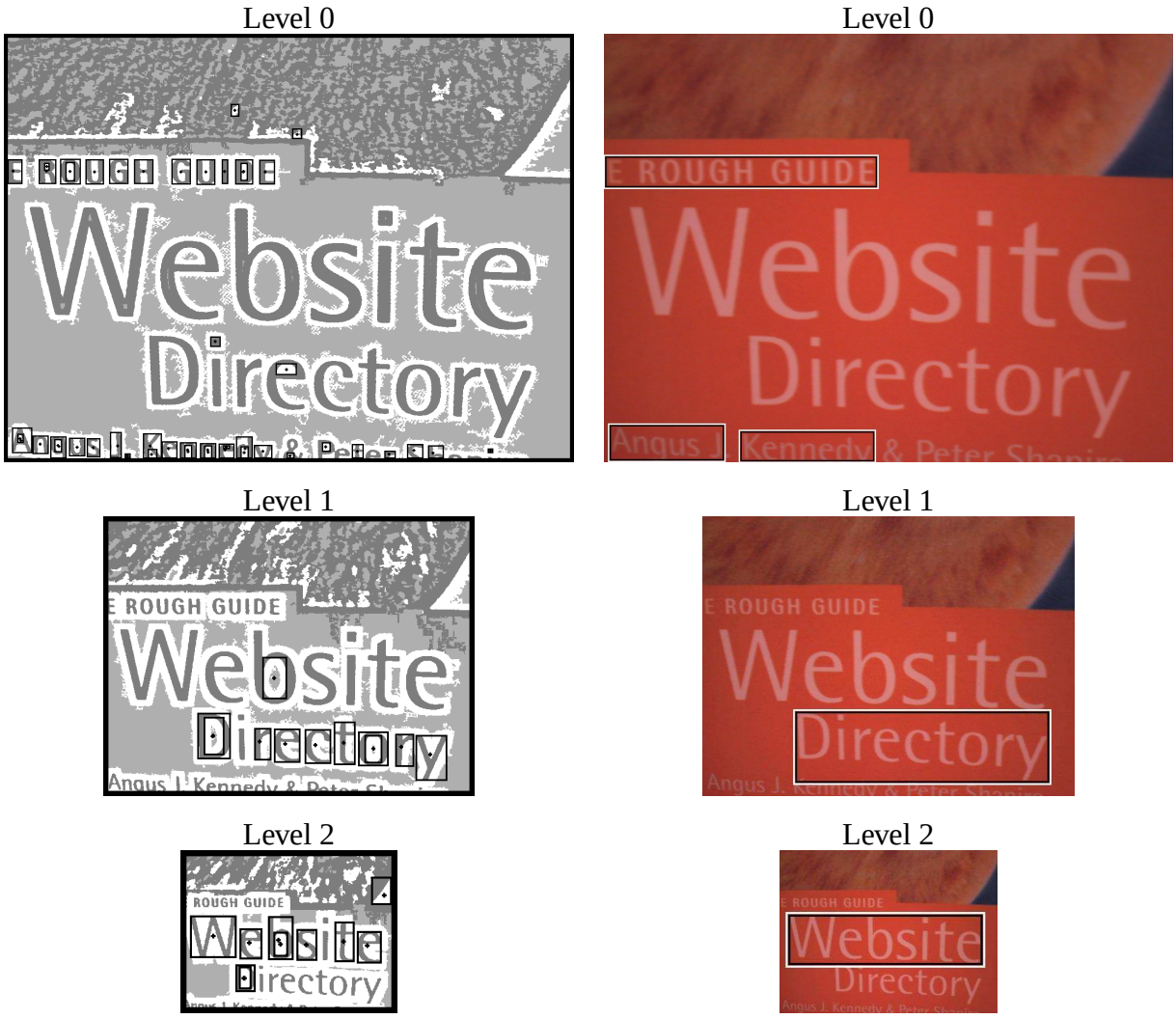

Figure 12: Example of multiscale segmentation and letter detection, showing accepted candidate letters (left) and accepted text lines (right) at each level.

will tend to confuse the the character/non-character classifier. With the multi-scale approach, these problems are largely avoided when the segmentation procedure is applied the scale where the letters are still legible but those finer details have been blurred away. See figure 13.

Figure 13: Example of multiscale segmentation and letter detection, illustrating texture suppression at the proper scale. Note that the letters of 'SIGNO' are oversegmented in levels 0 and 1 but correctly segmented and recognized (black boxes) in level 2.

3.4. Text region validation module

The character detection module analyzes only the segmented character shapes in isolation, and the character grouping module looks only at their bounding boxes. In order to obtain a

good end-to-end recall score, these two modules must be tuned to accept a high rate of false positives—regions that are mis-identified as characters and grouped into spurious text line candidates. See figure 14.

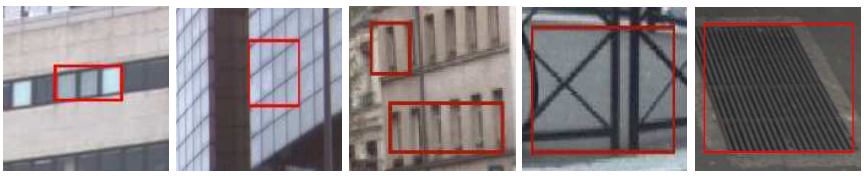

Figure 14: Some examples of false positives found by the character detection and grouping modules.

The task of SNOOPERTEXT's region validation module is to weed out these false positives, by analyzing the image content of each candidate text region as a whole.

This module is basically a texture classifier, based on the T-HOG descriptor [4]. The latter is a variant of Dalal and Triggs's R-HOG descriptor [5], specialized to capture the gradient distribution characteristic of letter strokes in occidental-like scripts. The T-HOG descriptor is fed to an SVM classifier, whose output is thresholded to give a binary text/non-text classification.

The T-HOG descriptor is based on the observation by Chen and Yuille (2004) that different parts of Ocidental-like text have distinctive distributions of edge directions [26]. That is because images that consist of bi-level strokes (such as Roman letters), the orientations of the strongest gradients tell the orientations of those strokes.

The HOG-based text/non-text discriminators reported in the literature generally use a two-dimensional array of $n_x \times n_y$ cells, as Dalal and Triggs used for human body recognition [5]. The use of a two-dimensional cell array may be justifiable for isolated characters, but does not seem to be useful for multi-character texts of variable width. In such texts, the gradient distribution is largely independent of horizontal position; therefore, a cell layout with vertical cuts increases the size of the descriptor without providing any additional relevant information.

The detailed description of the T-HOG descriptor and its experimental analysis have been published separately [4]. To obtain the descriptor, the image $\mathbb{I}$ delimited by the candidate rectangle is first extracted, converted to grayscale, scaled to a fixed height H, preserving its aspect ratio, and normalized for local variations of brightness and contrast with a Gaussian weight window. The sub-image is then divided into a small fixed number n_y of horizontal stripes, and a histogram with n_b of bins is built for each stripe. Specificaly, the image gradient is computed at each pixel within the stripe, its direction is quantized into a small number n_b of equal angular ranges, and the corresponding bins of the histogram are incremented. Opposite directions are identified, so each bin is π/n_b radians wide. The T-HOG descriptor is the concatenation of those n_y histograms.

The contribution of each pixel to the histogram is weighted by the gradient's norm, so that the small gradients that result from camera and quantization noise are largely ignored. Both the stripes and the histogram bins have gradual boundaries in order to minimize the impact of small vertical shifts and rotations of the text inside the bounding box.

Indeed, through extensive experiments [4] we confirmed that, for any descriptor length, the partition into horizontal stripes was generally more effective than a two-dimensional arrangement. Moreover, near-optimal results could be obtained with fairly small descriptors: we are using $n_\mathrm{y} = 7$ and $n_\mathrm{b} = 9$ in SNOOPERTEXT, but if used instead $n_\mathrm{y} = 4$ and $n_\mathrm{b} = 5$ we would reduce the descriptor size to 20 while lowering the scores by 1 to 2% on average.

We also found that pre-scaling the given text region to a small fixed height H (currently 24 pixels) was more efective than computing the HOGs at the original resolution. We believe that this resizing step is a good balance between preservation of useful detail and removal of noise and spurious texture

4. Performance of the text detector

In this section, we compared the performance of SNOOPERTEXT with that of other text detectors published in the literature, in their ability to locate text lines.

4.1. Image collections

For our tests, we used four image collections, described below. For each of these datasets we have a *reference file* containing the bounding boxes of its text regions and the corresponding text contents, in a simple XML format.

1. `ITW`: a subset of the iTowns Project's image collection [2], consisting of 100 frontal high-resolution color photos of Parisian façades, with 1080×1920 pixels, as taken by the iTowns vehicle. The reference file, containing 848 readable text words, is available on-line [37].
2. *`SVT`*: a public benchmark of 249 urban photos selected from the Google Street View images by Wang et al., ranging from 1024×768 to 1918×898 pixels. The reference file contains 647 readable words [20].
3. *`EPS`*: the benchmark used by Epshtein et al. [7], with 307 color images of urban scenes, ranging from 1024×768 to 1024×1360 pixels, taken with hand-held cameras. The reference file, containing 1981 readable text lines, was provided by the authors [19] and converted to XML by us [37].
4. *`ICD`*: the "testing" half of the 2005 ICDAR Challenge collection [22], consisting of 249 color images, ranging from 307×93 to 1280×960 pixels, captured with various digital cameras, of book covers, road signs, posters, etc. The reference file, containing 1107 readable text words, was provided by the Challenge organizers.

The `ICD` collection is not very appropriate for our purposes, since it has images with just one single big character, photos of book covers, etc. We included it because it is a popular

benchmark for text detection, and it is the only one for which we have several performance of other detectors.

4.2. Parameter settings

The SNOOPERTEXT parameters, described in section 3, were set as follows: an image pyramid height $m = 3$ levels; minimum constrast $c_{\min} = 8$ for `ITW`, `ICD` and `SVT` datasets, and $c_{\min} = 12$ for `EPS` dataset; relative threshold $t = 0.79$; maximum height difference $t_1 = 1.1$; the t_2 parameter (maximum relative letter spacing) was set differently for each dataset so that characters would be grouped in the same way as in the corresponding reference file: namely, into words for `ITW`, `ICD` and `SVT` ($t_2 = 0.4$), and into lines for `EPS` ($t_2 = 0.8$); and merging of multiple lines $t_3 = 0.7$.

The T-HOG was trained with the true and false positives regions reported by the character detection and grouping modules of SNOOPERTEXT, when applied to the "training" subset of the `ICD` image dataset and with ground-truth text regions. The T-HOG SVM used a Gaussian χ^2 kernel, whose standard deviation was found by cross-validation.

4.3. Competing detectors

We compared SNOOPERTEXT against several state-of-the-art text detectors described in the literature. Specifically, we compared it with the contestants of the ICDAR Challenge [23], and also with the detectors of Yao et al. [1], Epshtein et al. [7], H. Chen et al. [8], Pan et al. [9], Neumann et al. [10] and Yi et al. [11]. (The system of Mancas-Thillou and Gosselin [18] uses the text detector of Alex Chen, which is included in our set.)

We added to our list of competing text detectors the front-end module of the popular open-source TESSERACT OCR software (TESSFRONT), whose task is to locate the candidate text regions in the input image before calling the back-end (TESSFRONT) to parse them. TESSERACT is considered one of the best OCRs publicly available today [6]; however, its

front-end was designed for scanned documents, and it usually reports a large number of false positives when applied to photos of 3D scenes. See figure 15 (top). Therefore, we also added to our list of competing detectors the combination of TESSFRONT with the T-HOG as an output filter (TESSFRONT+T-HOG). See figure 15 (bottom).

4.4. Rectangle-based performance metrics

The quantitative criteria we used to compare these text detector systems are based on the ICDAR 2005 measure of similarity [23] between two rectangles r, s, defined as

$$m(r,s) = \frac{A(r \cap s)}{A(r \cup s)} \tag{1}$$

where $A(t)$ is the area of the smallest rectangle enclosing the set t. The function $m(r,s)$ ranges between 0 (if the rectangles are disjoint) and 1 (if they are identical). See figure 16.

The function m is extended to a set of rectangles S by the formula

$$m(r,S) = \max_{s \in S} m(r,s) \tag{2}$$

From this indicator one derives the ICDAR *precision* P and *recall* R scores [23],

$$P = \frac{\sum_{r \in E} m(r,T)}{\#E} \qquad R = \frac{\sum_{r \in T} m(r,E)}{\#T} \tag{3}$$

where T is the set of rectangles in the reference file, and E is the set of rectangles reported by the detector.

For ranking purposes, the ICDAR 2005 Committee used the F*-measure* [23], which is the harmonic mean of precision and recall: $F = 2/(1/P + 1/R)$.

4.5. Computing average scores

There are several ways of averaging the P, R, and F scores over a multi-image database. The approach used by the ICDAR 2005 scoring program (method I) is to evaluate P, R and

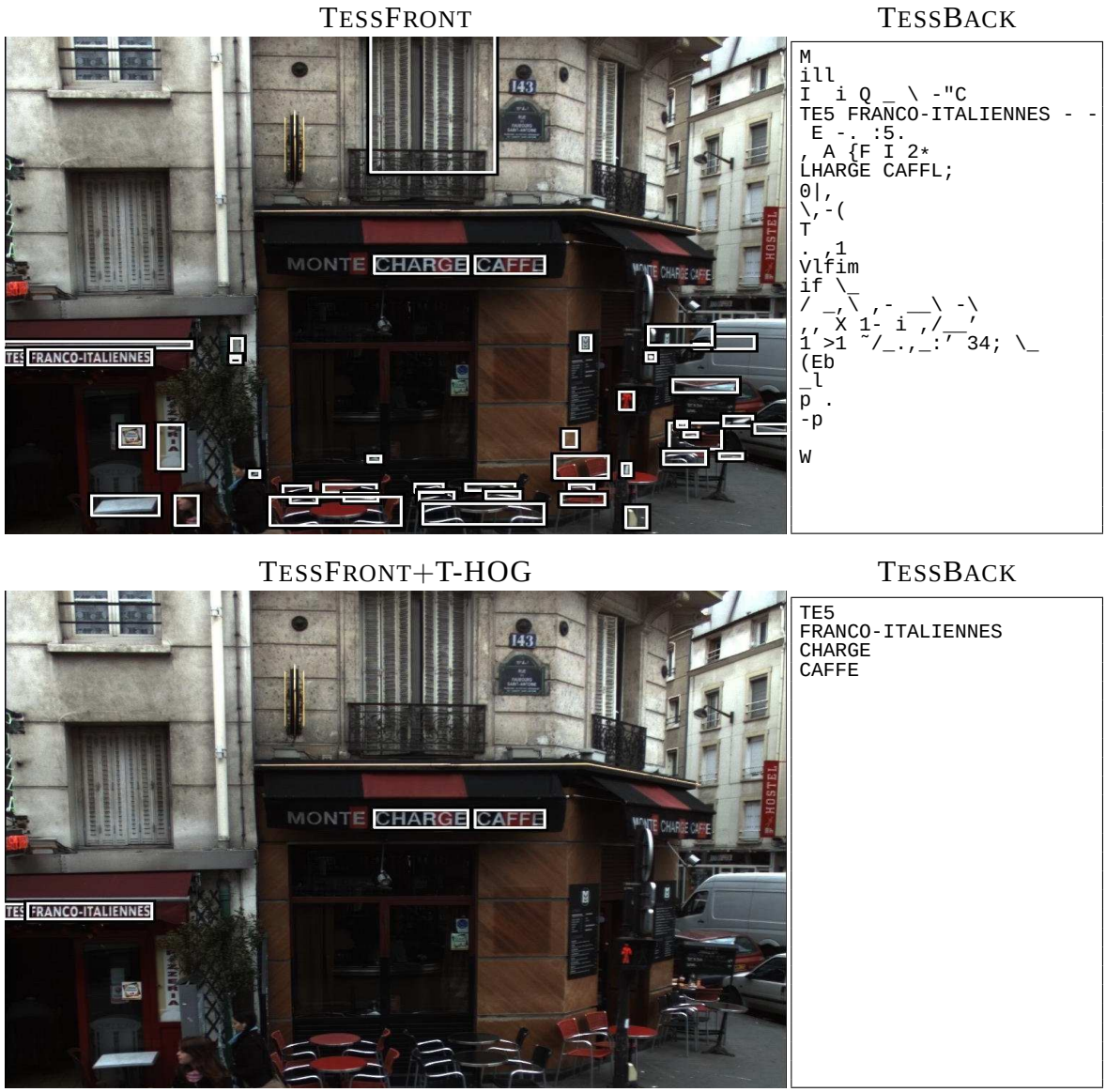

Figure 15: Examples of the TESSFRONT detector output (top) and TESSFRONT with T-HOG post filter (bottom). At left, the input image with the detected regions highlighted. At right the OCR (TESSBACK) output for those text regions.

Figure 16: The rectangle similarity score $m(r, s)$ for various text regions s detected by SNOOPERTEXT (solid outlines) and the best-matching regions r from the reference file (dashed).

F separately for each image, and then compute the arithmetic mean of all three scores over all images. Another approach (method II) is to compute P and R for each image, then take the arithmetic means of all P and R values, and compute F from these means. Yet another approach (method III) is to compute the precision and recall formulas (3) by taking E and T as the union of all text regions in all images. We note that averaging method I suffers from higher sampling noise and a negative bias compared to the other two, because it gives equal weight to each image irrespective of the number of recoverable text objects in it and the F-score is a non-linear function of the P and R rates. In particular, the averaged F score (method I) tends to be lower than the harmonic mean of averaged P and R (method II).

This point must be considered when comparing F values reported by different authors, since it is not always clear how they were averaged.

4.6. Results of text detection evaluation

The text detection scores on the four image collections are shown in tables 1–4. For our comparisons, in order to match the other entries, we had to recomputed the overal F score ourselves (according to the averaging method II), for all detectors, from the global P and R scores reported by the authors. We could not use method III because, for most competitors, the required data (the set of rectangles detected in each image) was not available.

	Detection scores		
System	P	R	F
SNOOPERTEXT	**0.72**	**0.50**	**0.59**
TESSFRONT+T-HOG	0.30	0.13	0.18
TESSFRONT	0.05	0.15	0.07

Table 1: Text detection scores of SNOOPERTEXT and other detectors on the ITW dataset.

	Detection scores		
System	P	R	F
SNOOPERTEXT	**0.36**	**0.51**	**0.42**
Neumann et al. [10]	0.19	0.33	0.26
TESSFRONT + T-HOG	0.15	0.15	0.15
TESSFRONT	0.04	0.18	0.06

Table 2: Text detection scores of SNOOPERTEXT and other detectors on the SVT dataset.

4.7. Discussion

As can be seen in table 1, the performance of SNOOPERTEXT was much better than that of TESSFRONT (which was not designed for photos of outdoor scenes), even using high resolution images of the ITW dataset. When we consider the two challenging urban scene datasets SVT and EPS, tables 2–3, with text in different orientations and images with noise,

	Detection scores		
System	P	R	F
SNOOPERTEXT	**0.59**	**0.47**	**0.52**
Epshtein et al. [7]	0.54	0.42	0.47
TESSFRONT+T-HOG	0.21	0.10	0.13
TESSFRONT	0.02	0.14	0.04

Table 3: Text detection scores of SNOOPERTEXT and other detectors on the EPS dataset.

	Detection scores		
System	P	R	F
Yi et al. [11]	**0.73**	0.67	**0.70**
Pan et al. [9]	0.67	**0.70**	0.69
SNOOPERTEXT	**0.73**	0.61	0.67
Yao et al. [1]	0.69	0.66	0.67
H. Chen et al. [8]	**0.73**	0.60	0.66
Epshtein et al. [7]	**0.73**	0.60	0.66
Hinnerk Becker†	0.62	0.67	0.64
Alex Chen†	0.60	0.60	0.60
Ashida†	0.55	0.46	0.50
HWDavid†	0.44	0.46	0.45
Wolf†	0.30	0.44	0.36
Qiang Zhu†	0.33	0.40	0.36
TESSFRONT+T-HOG	0.35	0.27	0.30
Jisoo Kim†	0.22	0.28	0.25
Nobuo Ezaki†	0.18	0.36	0.24
TESSFRONT	0.18	0.29	0.22
Todoran†	0.19	0.18	0.19

Table 4: Text detection scores of SNOOPERTEXT and other detectors on the ICD dataset. The competitors of the ICDAR 2003 and 2005 challenges are marked with †.

the SNOOPERTEXT improved the F score by 16% and 5%, respectively, over the second best. Note in table 3, that SNOOPERTEXT outperformed the Stroke Width Transform (SWT) [7] of Epshtein et al. on their own dataset.

In the `ICD` dataset, table 4, the performance of SNOOPERTEXT is comparable to that of the best detectors described in the literature.

As can be seen in tables 1–4, filtering the output of TESSFRONT through our text region validation module significantly improved its precision with little loss of recall.

5. End-to-end performance

We also measured the correctness of the strings extracted from the SNOOPERTEXT-detected regions by the OCR algorithm used in the iTowns application. Although the latter is not part of SNOOPERTEXT, we felt that this evaluation was necessary to confirm that the detector was adequate for the task (for example, that the text bounding boxes are neither too small or too large). For these tests, we used TESSERACT's back-end (TESSBACK), the module that attempts to parse the strings supposedly contained in the regions identified by the text detector.

TESSERACT is publicly available at [6] but is designed for scanned documents. Robust OCR algorithms especially designed for images of urban scenes is an active area of research, and some recently advances were described by Wang et al. [38], Mishra et al. [39] and Neumann and Matas [10]. However, an evaluation of OCR algorithms is beyond the scope of this paper.

Specifically, the goal of this section is to evaluate the following points: (1) text regions missed by the SNOOPERTEXT really impacts on the text recognition performance; (2) OCR designed for scanned documents are feasible in urban scenes. Therefore, we considered three alternative algorithms, namely: the "perfect" detector (IDEAL), that returns the manually annotated text regions from the reference file; TESSERACT's front-end module (TESSFRONT);

and that same module with its output filtered by SNOOPERTEXT's region validation module (TESSFRONT+T-HOG). See figures 15 and 17.

Figure 17: Examples of the reference text regions (top) and the SNOOPERTEXT detection (bottom). At left, a crop of the input image with the detected regions highlighted. At right the OCR (TESSBACK) output for those text regions. The correct readings of the reference regions are: "143", "MONTE", "CHARGE" (2x), "CAFFE" (2x), "TES", "FRANCO" and "ITALIENNES".

5.0.1. OCR-based performance metrics

For these comparisons, we used two scoring functions that take into account the correctness of the OCR-extracted text. Both functions assume that the strings are converted to lower case, because it is often impossible to tell whether a text (for example, formed by the letters C, O, S, U, V, X, W and Z) in urban signage is in upper or lower case.

We assume that the OCR algorithm attaches the extracted text, denoted by $r.ocr$, to the given rectangle r. We define the *rigorous OCR similarity score* m' for two rectangles r and s as

$$m'(r,s) = \begin{cases} 1 & \text{if } m(r,s) \geq \lambda \text{ and } r.ocr = s.ocr \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where m is the rectangle similarity function defined in equation (1), and λ is a fixed threshold (0.2 in our tests).

The scoring function m' may be considered too rigorous, because at the current state of the art one cannot expect that an OCR algorithm will correctly read store and product names which are missing from its spell-checking dictionary. Therefore, we also defined a *tolerant OCR similarity score* m'' that gives credit for partially correct OCR readings; namely,

$$m''(r,s) = \begin{cases} 1 - \dfrac{\mathrm{dist}(r.ocr, s.ocr)}{\max(|r.ocr|, |s.ocr|)} & \text{if } m(r,s) \geq \lambda \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Here $|u|$ denotes the length of string u, and dist denotes the Levenshtein distance between strings [40]. The latter is defined as the minimum number of edit operations needed to transform one string into the other, where each operation is the insertion, deletion, or substitution of a single character. Since the Levenshtein distance does not exceed the length of the longest string, the metric $m''(r,s)$ ranges between 0 (when the strings have no letters in common) and 1 (when the strings are equal).

As in section 4.4, we extend the scoring function m' to a set of OCR-scanned rectangles

S by the formula

$$m'(r, S) = \max_{s \in S} m'(r, s) \tag{6}$$

We then define the *rigorous OCR performance scores* P' (precision) and R' (recall) by the formulas

$$P' = \frac{\sum_{r \in E} m'(r, T)}{\#E} \qquad R' = \frac{\sum_{r \in T} m'(r, E)}{\#T} \tag{7}$$

where T is the set of manually identified text regions in all input images, with the ocr fields set to the visually extracted text values, as recorded in the reference file; and E is the set of text regions reported by the detector, with the TESSBACK-computed ocr fields. As before, we combine the OCR precision and recall into a single OCR score $F' = 2/(1/P' + 1/R')$. The *tolerant OCR performance scores* P'' R'', and F'' are defined in the same way, using m'' instead of m' in formulas (6–7).

Figure 18 illustrates the metrics m' (equation 4) and m'' (equation 5) on some text regions reported by SNOOPERTEXT.

	OCR scores					
	Rigorous			Tolerant		
System	P'	R'	F'	P''	R''	F''
IDEAL	0.29	0.29	0.29	0.50	0.50	0.50
SNOOPERTEXT	0.22	0.18	0.20	0.43	0.37	0.40
TESSFRONT + T-HOG	0.28	0.03	0.06	0.45	0.07	0.12
TESSFRONT	0.01	0.05	0.01	0.01	0.10	0.03

Table 5: OCR performance scores of the TESSERACT back-end with the three text detectors on the `ITW` dataset.

5.1. Discussion

Table 5 shows that the rigorous OCR score F' of SNOOPERTEXT on the `ITW` dataset (20%), while low in absolute terms is 68% of the score obtained with IDEAL text detector,

Figure 18: The OCR similarity scores $m'(r, s)$ and $m''(r, s)$ for various text regions r extracted by SNOOPERTEXT+TESSBACK (solid outlines), and the best-matching regions s from the human-produced reference file (dashed).

	OCR scores					
	Rigorous			Tolerant		
System	P'	R'	F'	P''	R''	F''
IDEAL	0.22	0.22	0.22	0.40	0.40	0.40
SNOOPERTEXT	0.13	0.20	0.16	0.21	0.34	0.26
TESSFRONT + T-HOG	0.46	0.06	0.11	0.57	0.11	0.18
TESSFRONT	0.01	0.07	0.02	0.01	0.12	0.02

Table 6: OCR performance scores of the TESSERACT back-end with the three text detectors on the SVT dataset.

(29%). The tolerant OCR score F'' of SNOOPERTEXT (40%) is 80% of the IDEAL score (50%). In both aspects, SNOOPERTEXT is significantly better that TESSERACT's front end,

	OCR scores					
	Rigorous			Tolerant		
System	P'	R'	F'	P''	R''	F''
IDEAL	0.10	0.10	0.10	0.25	0.25	0.25
SNOOPERTEXT	0.06	0.05	0.05	0.24	0.18	0.21
TESSFRONT + T-HOG	0.24	0.01	0.01	0.38	0.03	0.06
TESSFRONT	0.00	0.01	0.00	0.01	0.04	0.01

Table 7: OCR performance scores of the TESSERACT back-end with the three text detectors on the EPS dataset.

	OCR scores					
	Rigorous			Tolerant		
System	P'	R'	F'	P''	R''	F''
IDEAL	0.44	0.44	0.44	0.55	0.55	0.55
SNOOPERTEXT	0.41	0.29	0.34	0.57	0.42	0.49
TESSFRONT + T-HOG	0.58	0.12	0.19	0.75	0.17	0.28
TESSFRONT	0.04	0.16	0.06	0.05	0.22	0.08

Table 8: OCR performance scores of the TESSERACT back-end with the three text detectors on the ICD dataset.

even when the latter is combined with the T-HOG validation module. Therefore, we can say that the OCR algorithm, not the text detector, is the main bottleneck of the iTowns system at present.

Note that the OCR precision scores P' and P'' of SNOOPERTEXT are close to (or even better than) those of the IDEAL text detector, because regions in the refererence file that are difficult for TESSBACK tend to be missed by SNOOPERTEXT. See figure 19.

The low OCR scores on the EPS dataset (table 7), even with the IDEAL text detector, are partly explained by the lower image quality and the smaller size of the texts included in its reference file, which even humans find hard to read. See figure 20. Indeed, 252 text regions

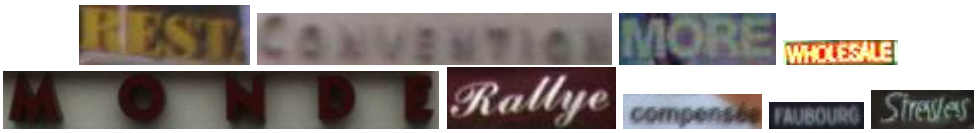

Figure 19: Examples of tougher text regions for text detectors and OCRs. Samples from the ITW and EPSs datasets.

(about 13%) in the reference file do not even have a string annotation, and therefore will be counted as errors even with an ideal OCR algorithm.

Figure 20: Examples of text regions in the EPS reference file without textual annotation.

6. Limitations

The SNOOPERTEXT errors seem to be due to texts that are near the low limit of legibility (small in size, blurred, partly obscured by noise), to groups of two or more characters that cannot be separated by the segmentation phase, and to isolated letters that are discarded by the grouping module. SNOOPERTEXT does not detect vertical aligned text regions or extremely tilted. See figure 21.

Figure 21: Text regions missed by the SNOOPERTEXT.

7. Conclusions

The combination of our SNOOPERTEXT detector with a standard OCR algorithm (TESSBACK) was used in the iTowns project to extract store signage and other textual information from photos building façades. These strings make it possible for iTowns users to locate stores by textual queries on store names and products. On a sample of the iTowns images, SNOOPERTEXT was able to accurately locate about 50% of the legible text regions, with less than 30% of false positives.

We attribute SNOOPERTEXT's success mainly to the use of multi-scale processing for segmentation an character detection, and to its effective text region validation module based on the T-HOG descriptor.

At present, the weakest spot in the iTowns text extraction system is the OCR algorithm, that yields the correct string only 30% of the time even when provided with an accurate bounding box. For this reason, the end-to-end recall score of the system (SNOOPERTEXT plus TESSBACK) is only 18%, with 22% precision.

SNOOPERTEXT can also accurately locate 60% of the text regions present the ICDAR Challenge benchmark, with less than 30% false positives. It is therefore competitive with state-of-the-art text detectors.

For photos of urban scenes SNOOPERTEXT is also significantly better than TESSERACT's built-in text detector, that achieves only 15% recall with and 5% precision (or 30% precision if combined with SNOOPERTEXT's T-HOG-based validator.)

References

[1] C. Yao, X. Bai, W. Liu, Z. Tu, Y. Ma, Detecting texts of arbitrary orientations in natural images, IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2012) 1–8.

[2] A. N. de Recherche, The iTowns Project, `http://www.itowns.fr`.

[3] Google, Google Street View, maps.google.com/help/maps/streetview.

[4] R. Minetto, N. Thome, M. Cord, N. J. Leite, J. Stolfi, T-hog: An effective gradient-based descriptor for single line text regions, Pattern Recognition 46 (3) (2013) 1078–1090.

[5] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2005) 886–893.

[6] Google, Tesseract-OCR, http://code.google.com/p/tesseract-ocr/.

[7] B. Epshtein, E. Ofek, Y. Wexler, Detecting text in natural scenes with stroke width transform, IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2010) 886–893.

[8] H. Chen, S. Tsai, G. Schroth, D. Chen, R. Grzeszczuk, B. Girod, Robust text detection in natural images with edge-enhanced maximally stable extremal regions, IEEE International Conference on Image Processing (ICIP) (2011) 1–4.

[9] Y.-F. Pan, X. Hou, C.-L. Liu, A hybrid approach to detect and localize texts in natural scene images, IEEE Transactions on Image Processing 20 (3) (2011) 800–813.

[10] L. Neumann, J. Matas, Real-time scene text localization and recognition, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, Washington, DC, USA, 2012, pp. 3538–3545.

[11] C. Yi, Y. Tian, Localizing text in scene images by boundary clustering, stroke segmentation, and string fragment classification, IEEE Transactions on Image Processing 21 (9) (2012) 4256–4268.

[12] R. Minetto, N. Thome, M. Cord, J. Fabrizio, B. Marcotegui, Snoopertext: A multiresolution system for text detection in complex visual scenes, IEEE International Conference on Image Processing (ICIP) (2010) 3861–3864.

[13] K. Jung, K. Kim, A. Jain, Text information extraction in images and video: a survey, Pattern Recognition (PR) - Elsevier 37 (5) (2004) 977–997.

[14] J. Liang, D. Doermann, H. Li, Camera-based analysis of text and documents: a survey, International Journal on Document Analysis and Recognition 7 (2) (2005) 84–104–104.

[15] P. W. Palumbo, S. N. Srihari, J. Soh, R. Sridhar, V. Demjanenko, Postal address block location in real time, Computer 25 (7) (1992) 34–42.

[16] I. S. I. Abuhaiba, M. J. J. Holt, S. Datta, Recognition of off-line cursive handwriting, Computer Vision and Image Understanding (CVIU) - Elsevier 71 (1) (1998) 19–38.

[17] N. Thome, A. Vacavant, L. Robinault, S. Miguet, A cognitive and video-based approach for multinational license plate recognition, Machine Vision and Applications.

[18] C. Mancas-Thillou, B. Gosselin, Color text extraction with selective metric-based clustering, Computer Vision and Image Understading (CVIU) - Elsevier 107 (1-2) (2007) 97–107.

[19] Epshtein et al. Text Detection Database, http://research.microsoft.com/enus/um/people/eyalofek/text_detection_database.zip.

[20] K. Wang, The street view text dataset, http://vision.ucsd.edu/~kai/svt/.

[21] A. N. de Recherche, The iTowns Dataset (annotated), www.itowns.fr/benchmarking.html (2009).

[22] S. M. Lucas, Text Locating Competition - International Conference on Document Analysis and Recognition (ICDAR) (2003-2005), http://algoval.essex.ac.uk/icdar/Datasets.html.

[23] S. M. Lucas, Icdar 2005 text locating competition results, in: International Conference on Document Analysis and Recognition (ICDAR), 2005, pp. 80–84 Vol. 1.

[24] K. C. Kim, H. R. Byun, Y. J. Song, Y. W. Choi, S. Y. Chi, K. K. Kim, Y. K. Chung, Scene text extraction in natural scene images using hierarchical feature combining and verification, in: International Conference on Pattern Recognition (ICPR), Vol. 2, 2004, pp. 679–682.

[25] Q. Ye, Q. Huang, W. Gao, D. Zhao, Fast and robust text detection in images and video frames, Image and Vision Computing (IVC) 23 (2005) 565–576.

[26] X. Chen, A. L. Yuille, Detecting and reading text in natural scenes, IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2 (2004) 366–373.

[27] T. Ojala, M. Pietikäinen, T. Mäenpää, Multiresolution gray-scale and rotation invariant texture classification with local binary patterns, IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 24 (2002) 971–987.

[28] Y.-F. Pan, X. Hou, C.-L. Liu, A robust system to detect and localize texts in natural scene images, IAPR International Workshop on Document Analysis Systems (2008) 35–42.

[29] S. M. Hanif, L. Prevost, Text detection and localization in complex scene images using constrained adaboost algorithm, International Conference on Document Analysis and Recognition (ICDAR) (2009) 1–5.

[30] X. Wang, L. Huang, C. Liu, A new block partitioned text feature for text verification, International Conference on Document Analysis and Recognition (ICDAR) 0 (2009) 366–370.

[31] J. Fabrizio, M. Cord, B. Marcotegui, Text extraction from street level images, City Models, Roads and Traffic (CMRT).

[32] J. Serra, Toggle mappings, From pixels to features (1989) 61–72J.C. Simon (ed.), Elsevier.

[33] J. Serra, Image Analysis and Mathematical Morphology, Academic Press, Inc., Orlando, FL, USA, 1983.

[34] C. Cortes, V. Vapnik, Support-vector networks, Machine Learning 20 (1995) 273–297.

[35] T. Retornaz, B. Marcotegui, Scene text localization based on the ultimate opening, ISMM 1 (2007) 177–188.

[36] J.-M. Jolion, A. Rosenfeld, A Pyramid Framework for Early Vision: Multiresolutional Computer Vision, 1994.

[37] R. Minetto, Reference files for the `ITW`, `EPS`, and `ICD` datasets, `http://www.dainf.ct.utfpr.edu.br/~rminetto/Text_Detection`.

[38] K. Wang, B. Babenko, S. Belongie, End-to-end scene text recognition, in: International Conference on Computer Vision (ICCV), IEEE Computer Society, Washington, DC, USA, 2011, pp. 1457–1464. doi:10.1109/ICCV.2011.6126402.
URL `http://dx.doi.org/10.1109/ICCV.2011.6126402`

[39] A. Mishra, K. Alahari, C. V. Jawahar, Top-down and bottom-up cues for scene text recognition, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, Washington, DC, USA, 2012, pp. 2687–2694.
URL `http://dl.acm.org/citation.cfm?id=2354409.2354999`

[40] V. I. Levenshtein, Binary codes capable of correcting deletions, insertions and reversals., Soviet Physics Doklady. 10 (8) (1966) 707–710.