T-HOG: an Effective Gradient-Based Descriptor for Single Line Text Regions

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

[1]*DAINF, Federal University of Technology of 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

We discuss the use of histogram of oriented gradients (HOG) descriptors as an effective tool for text description and recognition. Specifically, we propose a HOG-based texture descriptor (T-HOG) that uses a partition of the image into overlapping horizontal cells with gradual boundaries, to characterize single-line texts in outdoor scenes. The input of our algorithm is a rectangular image presumed to contain a single line of text in Roman-like characters. The output is a relatively short descriptor, that provides an effective input to an SVM classifier. Extensive experiments show that the T-HOG is more accurate than Dalal and Triggs's original HOG-based classifier, for any descriptor size. In addition, we show that the T-HOG is an effective tool for text/non-text discrimination and can be used in various text detection applications. In particular, combining T-HOG with a permissive bottom-up text detector is shown to outperform state-of-the-art text detection systems in two major publicly available databases.

Keywords: Text detection, text classification, histogram of oriented gradients for text, text descriptor.

[3]Corresponding author: Dr. Rodrigo Minetto, Curitiba, PR, Brazil, rodrigo.minetto@gmail.com

1. Introduction

In this paper we address the *text/non-text classification problem*. The input data for this problem is a rectangular sub-image of a digital photo or video frame. The output is a binary decision that should be 'TRUE' if the sub-image contains a single line of text in Roman-like characters and 'FALSE' otherwise. This classification is an important step in many applications, such as optical character recognition (OCR), indexing, classification of images and videos, and urban navigation aids.

Towards this goal, we describe here the *T-HOG*, publicly available at [1], a novel gradient-based descriptor that efficiently and accurately characterizes images of single-line texts. We show that a support vector machine (SVM) classifier [2] using T-HOG descriptors can effectively solve the text/non-text classification problem. In particular, we show that the combination of a "permissive" text detector [3] with a T-HOG based post-filter outperforms state-of-the-art text detectors described in the literature [4]. We also show how the T-HOG could be used by itself in a top-down sliding-window text detector, and as a component of an OCR system.

The T-HOG descriptor is based on the general *histogram of oriented gradients* (HOG) [5] method for shape recognition, introduced by Dalal and Triggs for the detection of pedestrians in photographs [5] and later used for other solid objects [6]. In order to capture the spatial distribution of gradient orientations, Dalal and Triggs divided the target sub-image into a rectangular grid of cells, computed a HOG for each cell, and concatenated those HOGs to obtain a composite descriptor, which they called *R-HOG*.

In 2004, Chen and Yuille [7] observed that different parts of the text regions have distinctive distributions of edge directions. This property was exploited by other researchers who used the R-HOG descriptors to characterize text regions [8, 9, 10].

The T-HOG descriptor is an improvement of the R-HOG, optimized for the specific task of single-line text recognition. The differences include a contrast normalization step, a different gra-

dient formula, and a specific cell layout with blurred boundaries. In this paper we determine experimentally the optimal cell tiling for text line recognition, which turns out to be a division of the candidate sub-image into horizontal stripes.

The T-HOG and R-HOG descriptors have several parameters that can be tuned in order to trade classifier accuracy for descriptor length. Smaller descriptors are interesting, even if less accurate, because they are more computationally efficient and may help us identify the aspects of the image that are most relevant for text/non-text discrimination. In this paper we also compare the performance of both classifiers experimentally for a wide range of parameters settings. The tests indicate that T-HOG is more accurate than R-HOG for any descriptor size.

1.1. Statement of the problem

We consider here images obtained from a physical scene. A *text object* is any part of the scene carrying a string of two or more letters that are readable in the captured image. We are primarily concerned with texts written in the Roman alphabet or any of its variants. See figure 1.

Figure 1: Image of an urban scene with text objects.

Our text classifier assumes that the candidate text object has been identified and its projection on the image has been bounded by a rectangle. Furthermore, it assumes that the text consists of a single multi-character line. Isolated characters and multiline text should be joined or split into separate lines or words.

1.2. Descriptor outline

Dalal and Triggs observed that a particular texture can often be characterized by the distribution of the directions of the image gradient. If the texture consists of simple bi-level shapes (such as Roman letters) then the orientations of the strongest gradients tell the orientations of the edges of those shapes.

In order to capture the spatial variation of edge orientations, Dalal and Triggs divided the input sub-image into a rectangular grid of (possibly overlapping) cells with n_x columns and n_y rows, which they grouped into 2×2 blocks. Within each cell of each block they computed a histogram of the gradient directions (HOG) with n_b bins. In these histograms the gradient direction of each pixel is weighted by the gradient's magnitude and by a Gaussian *block weight mask*. Their complete descriptor (R-HOG) is a vector with $n_x n_y n_b$ features, that is the concatenation of these $n_x n_y$ HOGs. Note that up to four overlapping or coincident cells may cover the same set of pixels, and each will generate a separate HOG, with different block weight functions. To reduce the effects of local contrast and brightness variations, the HOGs in each block are normalized in a specific way.

Our T-HOG descriptor differs from the original R-HOG in some key details. Firstly, we use different methods to extract the candidate text region, to normalize it for contrast, and to compute its gradient image. Secondly, the cell grid is simplified to a partition into horizontal stripes (i. e. we fix $n_x = 1$). Instead of overlapping blocks and block weight functions, in the T-HOG the cells are defined by overlapping *cell weight functions*. As a result, all internal cell boundaries are blurred, unlike those of the R-HOG. See figure 2. As detailed in section 4, these changes significantly improved the discriminating power for our target objects—single-line text regions of arbitrary length.

1.3. Structure of the paper

This paper is organized as follows. In section 2 we discuss some related work. In sections 3 and 4 we precisely define the R-HOG and T-HOG descriptors, and compare them experimentally. In section 5 we describe some applications. Finally, in section 6 we state the conclusions.

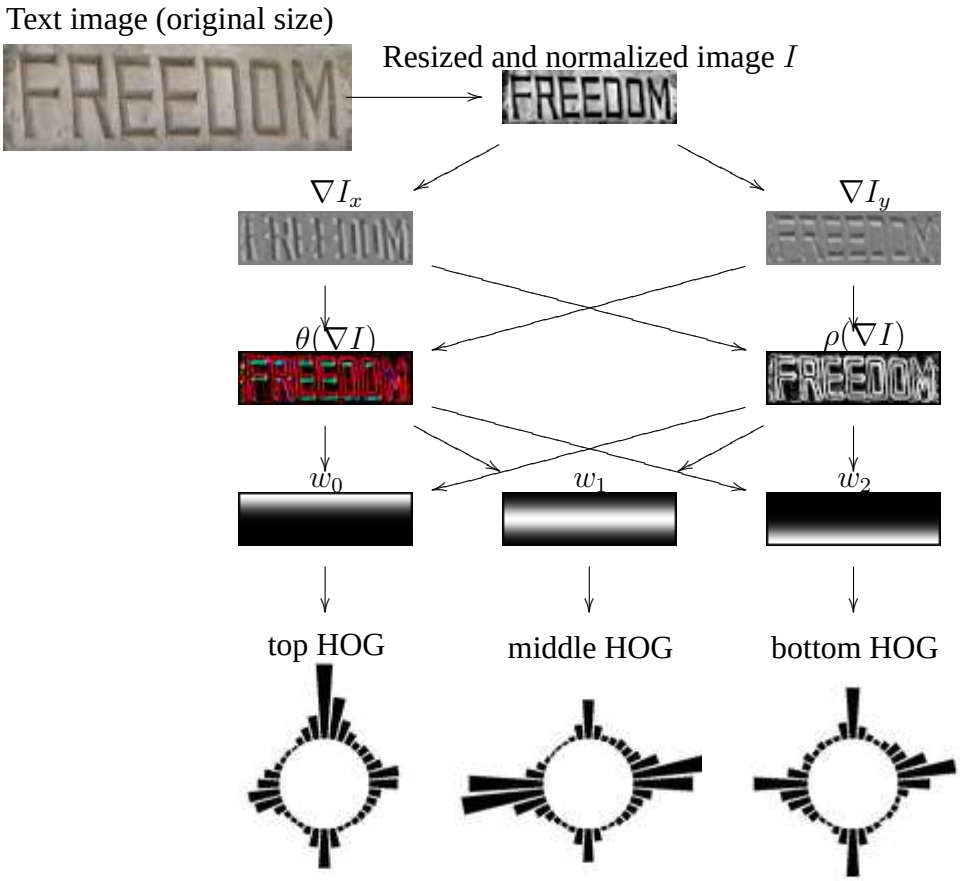


Figure 2: Computing the T-HOG descriptor for a sample image with an $n_x \times n_y = 1 \times 3$ cell grid and $n_b = 24$ histogram bins per cell. The images ∇I_x and ∇I_y are the derivatives of the extracted and normalized sub-image I. The images $\theta(\nabla I)$ and $\rho(\nabla I)$ are the direction and magnitude of the gradient. The images w_0, w_1 and w_2 are the cell weights.

2. Previous work

There is an extensive literature on text *detection*, but most of it are dedicated to specific contexts such as text detection in handwritten documents [11], text recognition in medieval manuscript images [12], and license plate recognition [13, 14]. An exhaustive review of this work is far outside the scope of the paper, and the reader is referred to the survey of Sharma et al. [15], that covers some advances in this area. Comparatively little has been published about text/non-text *classification*

algorithms (our primary interest in this paper), although they are often present as components of text detectors.

Text classification, or text verification [16], 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. [17] described a text recognizer that decomposes the candidate sub-image into a multiscale 16×16 cell grid and computes wavelet moments for each block. Each block is then classified as text or not using an SVM. The ratio of text to non-text outcomes is used to decide whether the entire sub-region is text or non-text. In 2005, Ye et al. [18] described a similar text recognizer with multiscale wavelet decomposition but they used more elaborate features including moments, energy, entropy, etc.

In 2010, Zhao et al. [19] used an edge detector based on the wavelet transform, and sparse representation with discriminative dictionaries to distinguish between text-like and background-like edge patterns. The authors then merged and trimmed the candidate text-like edges into compact regions by using an adaptive run-length smoothing algorithm, morphological operations, and projection profile analysis. Also in 2010, Shivakumara et al. [20] used 6 different gradient edge features (mean, standard deviation, energy, entropy, inertia and local homogeneity) over image blocks, to capture the texture property of the candidate text region.

In 2004, Chen and Yuille [7] 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 subdivisions of the image—89 features in total. Recently some text detectors, such as the one described by Anthimopoulos et al. [21] in 2010, have used descriptors based on multiscale *local binary patterns* (LBP) introduced by Ojala et al. [22]. Their descriptor has 256 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. [9]. 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 [8] for single-line text, and by Wang et al. [10] for isolated Chinese and Roman characters as well as single-line text. Hanif and Prevost's descriptor has 151 features (16 cells each with an 8-bin HOG, supplemented by 7 mean difference and 16 standard deviation features). The descriptor of Wang et al. has 80 features (8 cells with an 8-bin HOG, supplemented by 1 mean difference feature and 1 standard deviation over each cell).

All the HOG-based text recognizers above use vertical cuts as well as horizontal ones when partitioning the candidate region, apparently inspired by the Dalal and Triggs paper [5] on pedestrian recognition. Vertical cuts may be justifiable for isolated characters, but we determined experimentally (in section 4.5) that they are not useful for multi-character texts of variable width. In such texts, the gradient distribution is largely independent of horizontal position. Therefore, we have determined that a cell layout with vertical cuts increases the size of the descriptor without providing any additional relevant information.

3. The T-HOG descriptor

In this section we provide a detailed description of the T-HOG descriptor.

3.1. Size and contrast normalization

The first step of the T-HOG algorithm is to extract the sub-image and scale it to a fixed height H, maintaining its original aspect ratio. The height H should be large enough for the characters to remain readable, but small enough to eliminate most of the noise and other spurious detail. For print-style Roman characters (upper and lower case) we obtained the best results with H between 20 and 25 pixels.

In this step we also convert the image from color to gray scale, since the human visual system uses only the brightness channel to recognize character shapes [23]. We observed that objects in urban contexts are often obscured by non-uniform illumination and localized shadows or reflections. To remove these artifacts, we apply to each sample V of the extracted sub-image a contrast normalization procedure $V \leftarrow 0.5 + (V - \mu)/(3\sigma)$, where μ and σ are the local mean and standard deviation computed with a doubly binomial weight window of width $2H + 1$. The raw deviation σ is adjusted by $\sigma \leftarrow \sqrt{\sigma^2 + \varepsilon^2}$, where ε is the assumed standard deviation of the image sampling noise.

3.2. The basic HOG descriptor

By definition, the HOG descriptor of an arbitrary image I is a histogram of the gradient direction $\theta(\nabla I)$, computed at each pixel, quantized into a small number n_b of bins. Each pixel contributes to the histogram with "mass" proportional to its gradient magnitude $\rho(\nabla I)$, so as to de-emphasize the random noise-related gradient directions that occur in flat parts of the image. As observed by Dalal and Triggs, if $\theta(\nabla I)$ does not fall at the exact center of a bin, the mass should be distributed between the two nearest bins by a linear splitting criterion. To compute the gradient ∇I, we use the simple difference schema recommended by Dalal and Triggs, namely

$$\nabla I(x, y) = \frac{1}{2}(I(x + 1, y) - I(x - 1, y),\ I(x, y + 1) - I(x, y - 1))$$

For this formula, any non-existing pixel (outside the input sub-image) is assumed to be equal to the nearest existing pixel. Note that we compute the gradient after grayscale conversion and contrast normalization, whereas Dalal and Triggs compute the gradient in each color channel and then pick the vector that has the largest norm. We then estimated the magnitude of the gradient by the formula $\rho(\nabla I)(x, y) = \sqrt{\max\{0, |\nabla I(x, y)| - \varepsilon^2\}}$. Note that this formula is zero if the raw gradient norm $|\nabla I|$ is smaller than the assumed sampling noise deviation ε.

The gradient direction $\theta(\nabla I)$ is expressed as an angle in the range $[0, 2\pi]$ radians. Dalal and Triggs found that the recognition of some classes of objects (such as humans) was improved when

opposite directions were considered equivalent [5], in which case the range of $\theta(\nabla I)$ is $[0, \pi]$ radians. We found that this is not the case for text, where the directions had little effect.

Figure 3 shows the gradient magnitude and direction of four isolated letters and their corresponding HOG descriptors. The HOGs have 16 bins, each bin $2\pi/16$ radians wide, centered at orientations $2k\pi/16$ for $k = 0, 1, \ldots, 15$. One can see that the HOG gives the predominant orientation of the letter strokes. For example, the histogram of a rounded letter like 'O' is almost uniform over the whole range $[0, \pi]$, while that of 'I' has significant spikes in the directions perpendicular to the letter's stem.

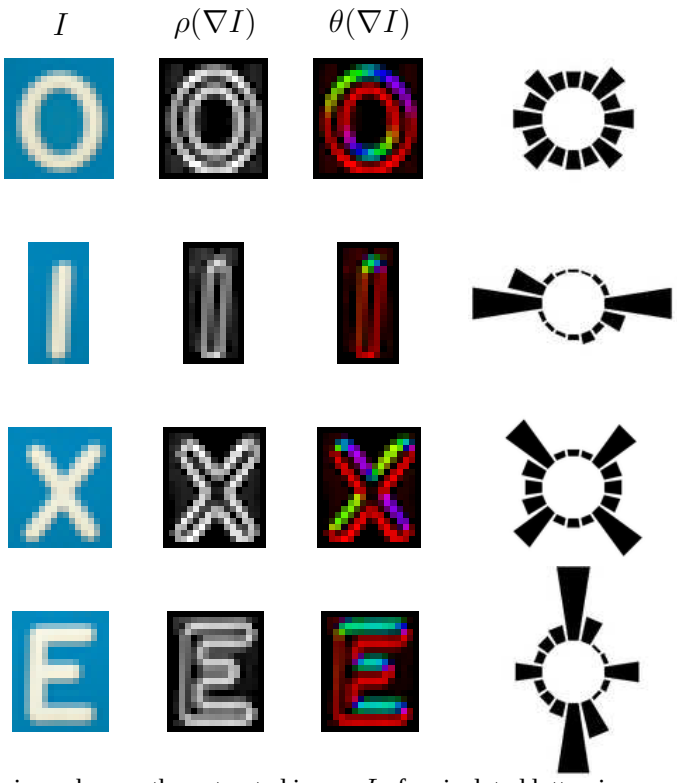


Figure 3: From left to right, in each row: the extracted image I of an isolated letter, images with its gradient magnitude $\rho(\nabla I)$ and gradient direction $\theta(\nabla I)$, and the corresponding HOG.

3.3. Multi-cell HOGs

Images of complex objects typically have different HOGs in different parts. Images of humans, for example, have different gradient orientation distributions in the head, torso, and leg regions. It was this observation that motivated Dalal and Triggs to use a multi-cell HOG (R-HOG) for that application.

This observation is also true for text images. Figure 4 shows the distributions of edge directions in the top, middle, and bottom parts of an image containing a single-line of text. Note that the gradient orientations are predominantly 0 (or 180) and 90 (or 270) degrees, reflecting the predominance of vertical and horizontal strokes. Also note that the top and bottom parts of the image contain a larger proportion of horizontal strokes, so that the gradients in these parts are mostly vertical. The middle part of the image, on the other hand, contains a larger proportion of vertical strokes, and hence of horizontal gradients. In all three regions there is a small amount of diagonal strokes due to letters such as ‘R’ and ‘M’; and to the rounded parts of letters such as ‘R’, ‘D’, and ‘O’. Finally, note that opposite directions tend to be equally represented due to the fact that the two edges of a letter stroke have opposite gradients.

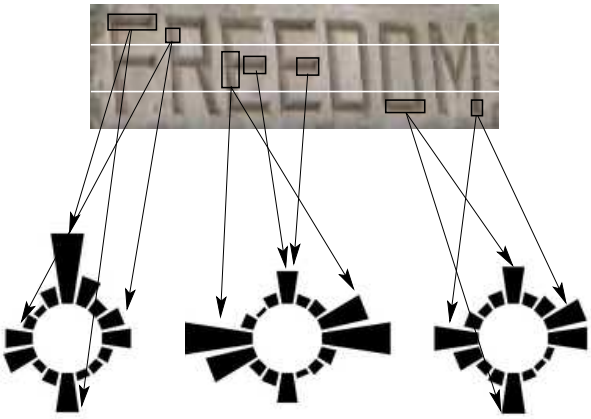


Figure 4: From left to right, the 16-bin HOG descriptors of the top, middle and bottom parts of a text sub-image. The arrows indicate the contribution of specific letter strokes to the histogram.

For comparison, figure 5 shows the HOG descriptors of top, middle and bottom regions of some non-text images. Note that several of these HOGs are quite distinct from those of figure 4, and some are significantly unbalanced. On the other hand, for an image containing an arbitrary single-line,

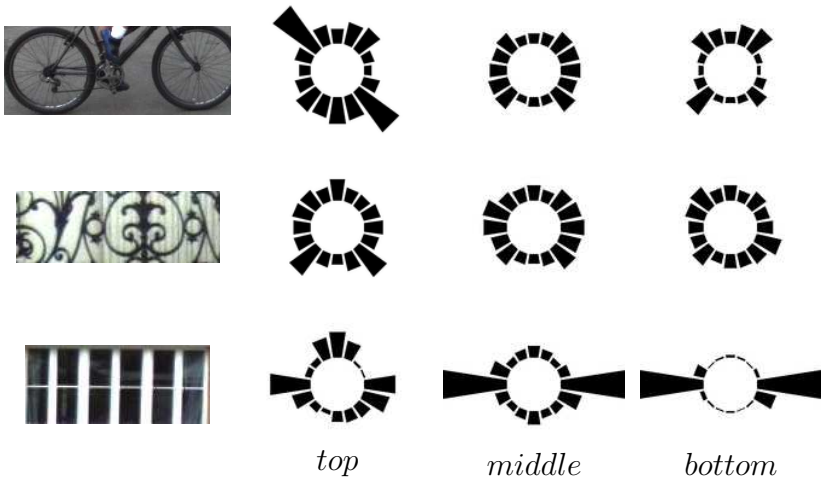


Figure 5: Top, middle, and bottom 16-bin HOG descriptors of some non-text images.

multi-character text, the expected distribution of gradient orientations is largely independent of the horizontal position along the line, as long as the segment analyzed is wide enough to include one whole character. This intuition was confirmed by extensive experimental tests; see section 4.5.

3.4. Cell weights

If the cells were defined by sharp boundaries, their HOGs would change drastically with small displacements of the text inside the candidate sub-image, as letter strokes would shift from one cell to the next. See figure 6 (a,b). To reduce this problem, the T-HOG cells are defined by smooth *cell weight functions*. This choice made the T-HOG more robust to such problems. See figure 6 (c,d).

Namely, let $x_{\min}$, $x_{\max}$, $y_{\min}$, and $y_{\max}$ be the minimum and maximum pixel coordinates in the sub-image. For each pixel with center coordinates (x, y), we define the *relative pixel coordinates*

$$X(x) = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \qquad Y(y) = \frac{y - y_{\min}}{y_{\max} - y_{\min}} \tag{1}$$

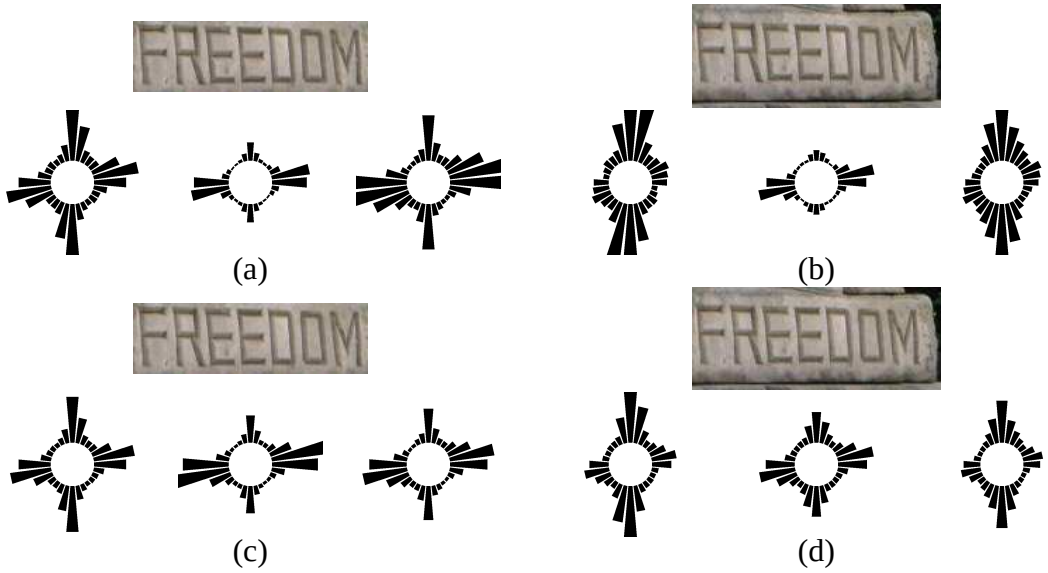


Figure 6: Effect of sharp cell boundaries with two different cropped sub-images of the same text object. (a,b) The HOG descriptors of the top, middle, and bottom parts of each sub-image using sharp cell boundaries. (c,d) The HOG descriptors of the top, middle, and bottom parts of each sub-image using smooth cell boundaries.

The weight of that pixel relative to a cell C_{ij} in column i and row j of the cell grid is then defined as $w_{ij}(x, y) = u_i(X(x))v_j(Y(y))$, where each function u_i or v_j is 1 at the nominal axis of the respective column or row, and falls smoothly to 0 as one moves away from it. The gradient of that pixel contributes to the histogram of cell C_{ij} with mass $\rho(\nabla I)(x, y)w_{ij}(x, y)$, rather than just $\rho(\nabla I)(x, y)$.

3.4.1. Gaussian cell weights

For the one-dimensional weights u_i and v_j, we tested different families of functions (Gaussian bells, Hann windows, Bernstein polynomials, etc). In these experiments, the best results were obtained with Gaussian bell functions. Specifically, for $n_y > 2$ rows of cells, the vertical weight function of cells in row j is

$$v_j(Y) = \gamma \left(-\mu_0 + \frac{1 + 2\mu_0}{(n_y - 1)j} \, , \, \frac{\sigma_0}{n_y} \, , \, Y \right) \tag{2}$$

where $\mu_0 = 0.01$, $\sigma_0 = 0.5$, and

$$\gamma(\mu, \sigma, z) = \exp\left(-\frac{(z-\mu)^2}{2\sigma^2}\right) \tag{3}$$

Figure 7 shows these weights for $n_y = 3$. As a special case, if $n_y = 1$, the single vertical weight

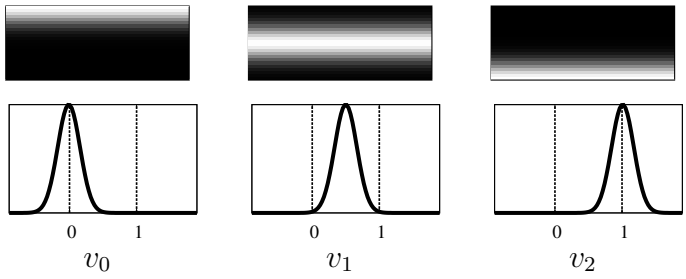


Figure 7: The T-HOG vertical cell weight functions v_0, v_1, and v_2 for $n_y = 3$.

function v_0 is equal to 1 everywhere. Note that the top edges of the topmost cells and the bottom edges of the bottommost cells are still sharp. The horizontal weight functions u_i are defined in the same way.

3.4.2. Emulating cells with hard edges

Hard-edged cells can be emulated in the T-HOG by defining each function u_i or v_j to be the appropriate step function. See figure 8.

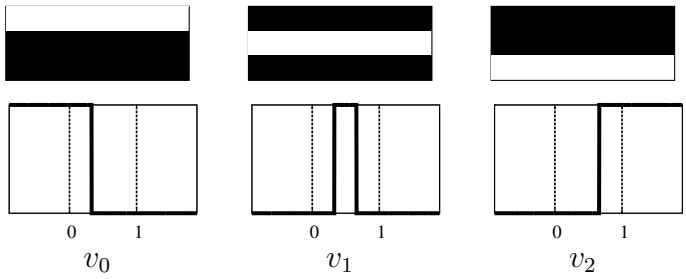


Figure 8: Step weight functions v_0, v_1, and v_2 used to emulate hard-edged cells in the T-HOG model.

3.4.3. Relation to R-HOG weight functions

Dalal and Triggs also used Gaussian weight functions, but in a different and more limited way. Their weight functions were associated to cell blocks (usually containing 2×2 cells) rather than individual cells. With the parameters they used for human recognition, the internal cell boundaries in each block are sharp, while the edges of the sub-image itself fade gradually to zero.

Figure 9 (top) shows the effective R-HOG cell weight functions for the best parameter configuration we found using $n_y = 3$ cells: namely, a single block divided into $1 \times n_y$ cells with a fairly broad block weight function ($\sigma_x = W/2$, $\sigma_y = H/2$, corresponding to setting the `wtscale` parameter to 1 in their implementation). With these parameters, the effective cell weight functions have quite sharp boundaries, as shown in figure 9 (top).

Figure 9 (bottom) shows the R-HOG cell weights for the same parameters, but with narrower block weight functions recommended by Dalal and Triggs for human recognition ($\sigma_x = W/4$, $\sigma_y = H/4$, corresponding to the default `wtscale` $= 2$).

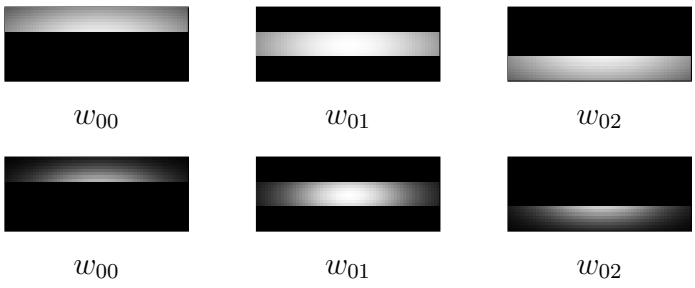


Figure 9: The Dalal and Triggs's cell weight functions for a single block of 1×3 cells. Top: optimal block weight deviations $\sigma_x = W/2$, $\sigma_y = H/2$. Bottom: default block weight deviations $\sigma_x = W/4$, $\sigma_y = H/4$.

One can obtain R-HOG weights somewhat similar to the T-HOG weights of figure 7 by using $1 \times n_y$ overlapping blocks with one cell per block, as shown in figure 10. Comparing the cell weights of figures 7 and 10, we observe that the latter assigns much lower mass to pixels along the edges of the sub-image (among other differences). Presumably for that reason, the R-HOG

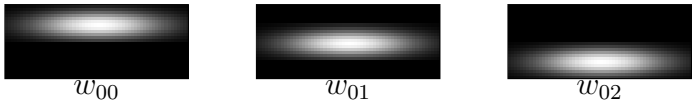


Figure 10: The Dalal and Triggs's cell weight functions for 1×3 single-cell blocks, each with height $H/2$ and overlapped with stride $H/4$, and with the default block weight deviations $\sigma_x = W/4$ and $\sigma_y = H/8$.

classifiers with the weights of figures 9 (bottom) and 10 were less accurate than the R-HOG with the weights of figure 9 (top), for the same descriptor size; and all three were worse than the T-HOG.

3.5. Normalization

Both algorithms, R-HOG and T-HOG, normalize the resulting descriptor. Dalal and Triggs use a per-block normalization scheme, which is intended to compensate for spatial variations of lighting and contrast over the input image. Since the T-HOG algorithm removes those effects beforehand, we simply divide the final descriptor by the sum of all features plus a constant ϵ (L_1 norm).

3.6. Vector classification and thresholding

Like Dalal and Triggs, we use an SVM classifier [2] to turn the descriptor $z \in \mathbb{R}^N$ into a real-valued score $f(z)$, such that positive scores indicate 'probably text' and negative scores indicate 'probably non-text'. The SVM is defined as $f(z) = \sum_{i=1}^{M} \alpha_i K(z_i, z) - b$ where K is the *kernel* [24], a function from $\mathbb{R}^N \times \mathbb{R}^N$ to $\mathbb{R}$; the z_i are the M fixed *support vectors*; the α_i are real weights; and b is the *bias* or *decision threshold*. The support vectors and weights are determined by a *training* step from representative samples of text and non-text descriptors.

3.7. Computation costs

The T-HOG and R-HOG algorithms have linear complexity, that is, proportional to the number of pixels in the extracted sub-image. Since the candidate text image is scaled to a fixed height H, the cost is roughly proportional to the number of characters in the text line.

4. Experiments

In this section, we describe an extensive set of experiments performed in order to determine optimum values for the various parameters of the R-HOG and T-HOG descriptors, and to compare their performance in the basic text/non-text discrimination task. These experiments strongly confirm the advantage of the two main T-HOG innovations, namely the splitting of the image into overlapping horizontal cells (section 3.3) with blurred boundaries (section 3.4).

4.1. Image collections

In our tests we used single-line text samples derived from three image collections:

1. The 2005 ICDAR challenge collection [25], consisting of 499 color images of book covers, road signs, posters, etc., captured with different cameras and resolutions.
2. A subset of the iTowns Project collection [26], consisting of 100 color images of Parisian façades taken by a camera-equipped vehicle (similar to Google's Street View).
3. The Epshtein et al. benchmark [4], with 307 color images of urban scenes, ranging from 1024×1360 to 1024×768 pixels, taken with hand-held cameras.

These image collections are suitable benchmarks for text *detectors*, but not for text *classifiers*. Therefore, we extracted from these image collections six sets of candidate sub-images as follows: We processed each image collection with SnooperText [3], a state-of-the-art text detector algorithm, tuned for high recall and moderate precision. Through visual inspection, we separated the candidate regions returned by SnooperText into a set of text regions X_i, and a set of non-text ('background') regions B_i, for $i = 1, 2, 3$. See figure 11. Table 1 gives the number of sub-images in each set. (For succinctness, we will often omit the index i in the remainder of the paper.)

4.2. Error rate metrics

To quantify the performance of a binary classifier (R-HOG or T-HOG) with a specific set of parameters, we adopted a 'ranking-based' approach. That is, we evaluated the ability of the classi-

i	Image Set	Detected regions	Text ($\|X_i\|$)	Non-Text ($\|B_i\|$)
1	ICDAR	4961	1727	3234
2	iTowns	2242	714	1528
3	Epshtein	7518	1502	6016

Table 1: Sizes of the text and non-text samples X_i, B_i used in our tests.

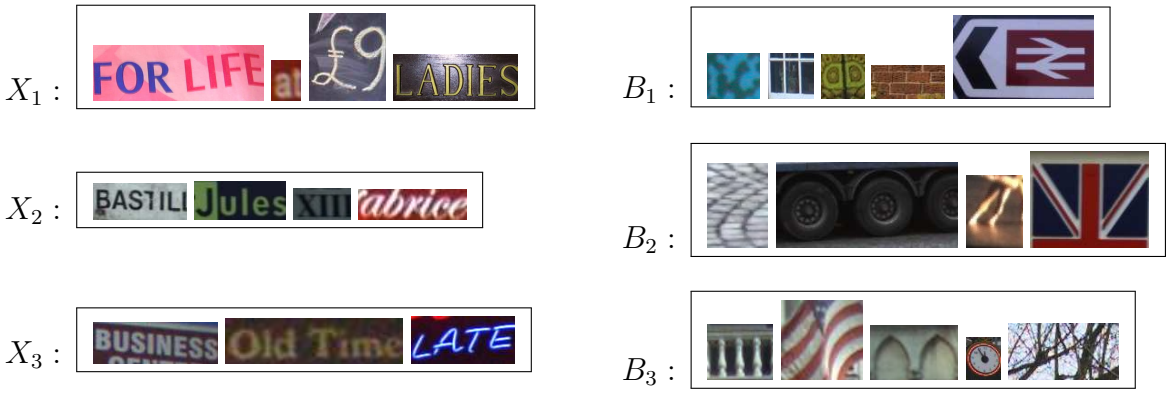


Figure 11: Samples of text regions (set X_i) and non-text regions (set B_i) extracted by SnooperText from the ICDAR, iTowns and Epshtein image collections.

fier to score text regions higher than non-text regions, regardless of the absolute value of the SVM score $f(z)$.

Specifically, in our tests we randomly divided the set X (respectively B) into two disjoint sets, each one with 50% of the elements: a 'training' half X' (respectively B') and a 'testing' half X'' (respectively B''). The sets (X', B') were used to train the SVM. We then applied the classifier to the complementary sets (X'', B''). For several values of the SVM threshold b (see section 3.6), we computed the counts TP_b, TN_b (correct decisions, positive and negative) and FP_b, FN_b (incorrect decisions). From these counts we computed the classification success rates for the text and non-text regions on each evaluation dataset, namely

$$\tau_b = \frac{\text{FN}_b}{|X''|} = \frac{\text{FN}_b}{\text{TP}_b + \text{FN}_b} \quad \beta_b = \frac{\text{FP}_b}{|B''|} = \frac{\text{FP}_b}{\text{TN}_b + \text{FP}_b} \tag{4}$$

The τ_b metric (*false negative rate*) is the complement of the well-known *recall* metric r; it is the probability of our algorithm incorrectly rejecting a text-containing region. The β_b metric (*false positive rate*) is the probability of our algorithm incorrectly accepting a non-text region. We choose to use β_b instead of the common *precision* metric because the latter depends strongly on the ratio $|B''|/|X''|$, which is essentially arbitrary.

By adjusting the threshold b, the user can trade one class of errors for the other. In particular, when b is sufficiently small, the classifier accepts all samples, so that $\tau_b = 1$ and $\beta_b = 0$. Conversely, when b is sufficiently large, all samples are accepted as text, therefore $\tau_b = 0$ and $\beta_b = 1$.

In order to reduce the sampling error, we repeated the whole procedure $L = 10$ times for each pair of datasets (X, B) resulting in L different random partitions (X', X'') and (B', B'') for each set. The raw statistics $\text{TP}_b, \text{TN}_b, \text{FP}_b, \text{FN}_b$ were averaged over these L runs, and for each b.

4.3. DET curve and area metric

We compare classifiers by plotting the *decision error trade-off* (DET) *curve* [5, 27], which is the set of pairs (τ_b, β_b) for $b \in [-\infty, \ldots, +\infty]$. See figure 12. For an ideal classifier, the DET curve lies along the bottom and left edges of the unit square $[0, 1] \times [0, 1]$. The better the classifier, the closer its DET curve should be to this ideal.

In our tests we observed that whenever a classifier C_i was significantly better than another classifier C_j for some threshold b, the same usually happened for most other values of b. In other words, the entire curve of C_i was closer to the ideal than that of C_j (below and to the left of it). Therefore, we can use the *decision error area* (DEA), which is the area A between the DET curve and the ideal curve (the shaded region in figure 12), as a single scalar measure of the performance of a given classifier, independent of the threshold b. The value of A is a monotonically decreasing function of the classifier's accuracy, and is zero if the classifier is perfect (i.e., if one can set the threshold b so that the classifier makes no mistakes). Therefore, we can compare two classifiers C_i and C_j by comparing the respective decision error areas A_i and A_j.

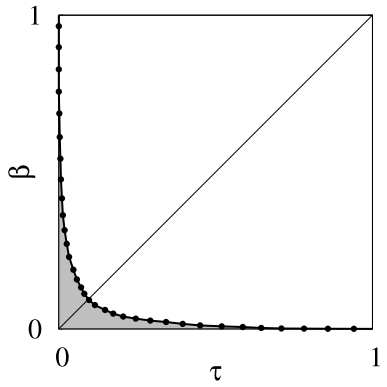


Figure 12: Area under the curve A in grey.

In order to determine whether the difference $A_i - A_j$ is statistically significant, we computed mean values $\boldsymbol{\mu}(A_i)$ and $\boldsymbol{\mu}(A_j)$ and the standard deviations $\boldsymbol{\sigma}(A_i)$ and $\boldsymbol{\sigma}(A_j)$ over the L runs. We then computed Student's test parameter $t(C_i, C_j)$

$$t(C_i, C_j) = \frac{\boldsymbol{\mu}(A_i) - \boldsymbol{\mu}(A_j)}{\sqrt{\frac{S^2}{L} + \frac{S^2}{L}}} \tag{5}$$

where

$$S^2 = \frac{(L-1) \cdot \boldsymbol{\sigma}(A_i)^2 + (L-1) \cdot \boldsymbol{\sigma}(A_j)^2}{2L - 2} \tag{6}$$

The performance variation between C_i and C_j is considered statistically significant at risk level α if $|t(C_i, C_j)|$ is above the corresponding threshold t_α from Student's table.

4.4. General parameter settings

In both the R-HOG and T-HOG algorithms, the sub-images were rescaled during extraction with the Lanczos interpolation filter [28] to the chosen height H. Since the extracted height must be a multiple of the effective number of cell rows, we used $H = 25$ pixels for 5 rows, $H = 21$ pixels for 7 rows, and $H = 24$ pixels for all other tests (with 1, 2, 3, 4, 6, 8 and 12 rows). The rescaled width W was chosen so as to maintain the aspect ratio of the original sub-image, but rounded to the nearest integer multiple of cell columns (which was 1 for most tests). For the mean-variance normalization and for gradient magnitude computation, we assumed a sampling noise with

deviation $\varepsilon = 0.02$. In all tests we used a Gaussian χ^2 SVM kernel K, whose standard deviation parameter σ was optimized by cross-validation on the training sets (X', B').

In an extensive series of preliminary tests, we concluded that the best performance of the R-HOG as text classifier, for all three datasets, is achieved with L_1 block histogram normalization, RGB colorspace with gamma correction 0.5 (RGB_SQRT), oriented gradient directions ranging over $[0, 2\pi]$ or $[0, \pi]$, and block mask parameter `wtscale` set to 1 ($\sigma_x = W/2$, $\sigma_y = H/2$). In another series of tests, the best T-HOG performance was obtained with L_1 whole-descriptor normalization, and oriented gradient directions ranging over $[0, 2\pi]$ or $[0, \pi]$. These optimal settings where then used for all subsequent tests.

4.5. Optimal cell arrangements

We next performed a series of tests to determine the optimum cell arrangement for text/non-text classification with the R-HOG algorithm, as a function of the total cell count $n_x n_y$. R-HOG allows the cells to be grouped into blocks, which may partially overlap. The possible arrangements with six cells (counting overlaps) are shown in figure 13. Arrangements (a)–(d) have disjoint, non-

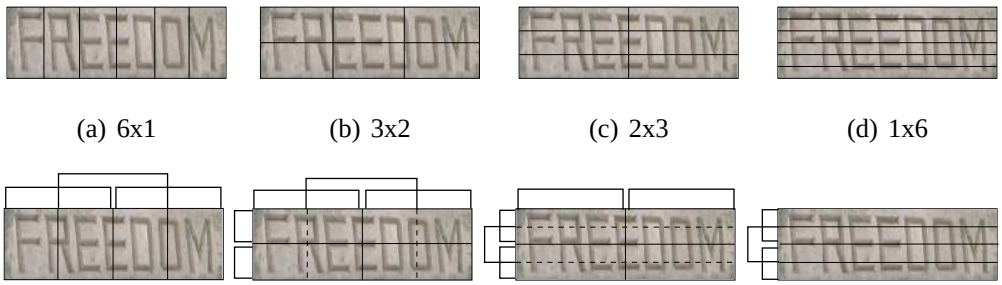


Figure 13: Some possible arrangements of blocks and cells that result in an R-HOG descriptor with six HOGs. Solid and dashed lines inside the image are the cell boundaries; the external brackets show the blocks.

overlapping cells, which could be grouped into disjoint blocks in several ways. Arrangements (e)

and (h) have two cells per block; note that the two central cells are duplicated in the final descriptor. Arrangements (f) and (g) have single-cell blocks that overlap by half a cell.

We tested many possible cell and block arrangements with and without overlapping blocks. The DET curves for some combinations of n_x and n_y with $n_b = 12$ are shown in Figure 14. Note that the counts n_x and n_y include overlapping cells so that the descriptor always consists of $n_x n_y$ HOGs. As mentioned in section 3.4, we concluded from these experiments that arrangements with two or more blocks, overlapping or not, are not advantageous for R-HOG. We have found that for the same descriptor size $N = n_x n_y n_b$ and number of bins, a single block is always better. Moreover, we concluded that, for the same descriptor size, the best choice is always $n_x = 1$, that is, a grid of n_y horizontal stripes. These conclusions were confirmed by numerous tests with the other two datasets and with different bin counts ($n_b = 6, 12, 18$ and 36).

A parallel series of tests with our T-HOG classifier gave entirely similar results, confirming that $n_x = 1$ is always the best choice for any descriptor size.

4.6. Performance as function of descriptor size

Having established that the best cell arrangement for R-HOG is always a single block divided into disjoint horizontal stripes, we performed another series of tests to analyze the influence of the number of stripes n_y and the number of bins per stripe n_b on the R-HOG classifier accuracy. Namely, we tested all combinations of $n_y = 1, 2, \ldots, 8, 12$ and $n_b = 4, 5, \ldots, 18, 24, 36$, with n_x fixed at 1. Figure 15 shows the results of these experiments for $N \leq 250$. Configurations are identified by the notation $n_x \times n_y{:}n_b$. From these tests, we concluded that a longer R-HOG descriptor generally gives better results. However, the advantage is very small for N greater than 100. In particular, no improvement was seen when N increased beyond 250. We also concluded that the R-HOG's accuracy improves dramatically as n_y increases from 1 to 3, improves more gradually until n_y is 7 or so, and remains the same thereafter. These conclusions were found to hold for all three datasets.

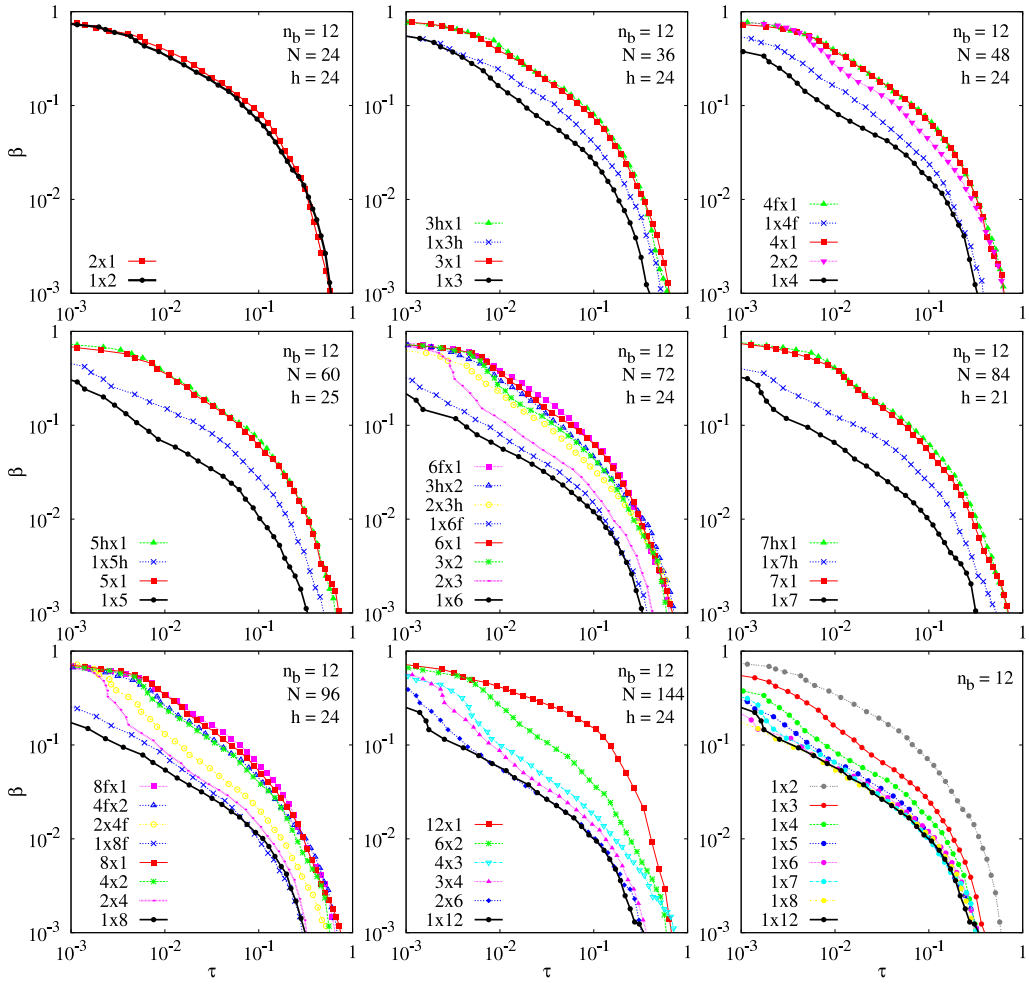


Figure 14: DET curves (mean of 10 random partitions) of the R-HOG classifier, for various cell and block arrangements, on the ICDAR dataset X_1, B_1. In each plot, except the last one, all grid configurations give the same descriptor size $N = n_x n_y n_b$. The last plot compares the best combinations of the eight previous plots.

In figure 15 (bottom), the black dots represent the optimal combinations of n_y and n_b, the only ones that are worth using for any specified descriptor size N. Configurations that fall above the solid staircase line (blue dots) are fully dominated by optimal ones, in the sense that the latter provides equal or better performance with equal or smaller N. There appears to be no simple formula for the optimal parameters, partly because n_y and n_b are constrained to be divisors of N.

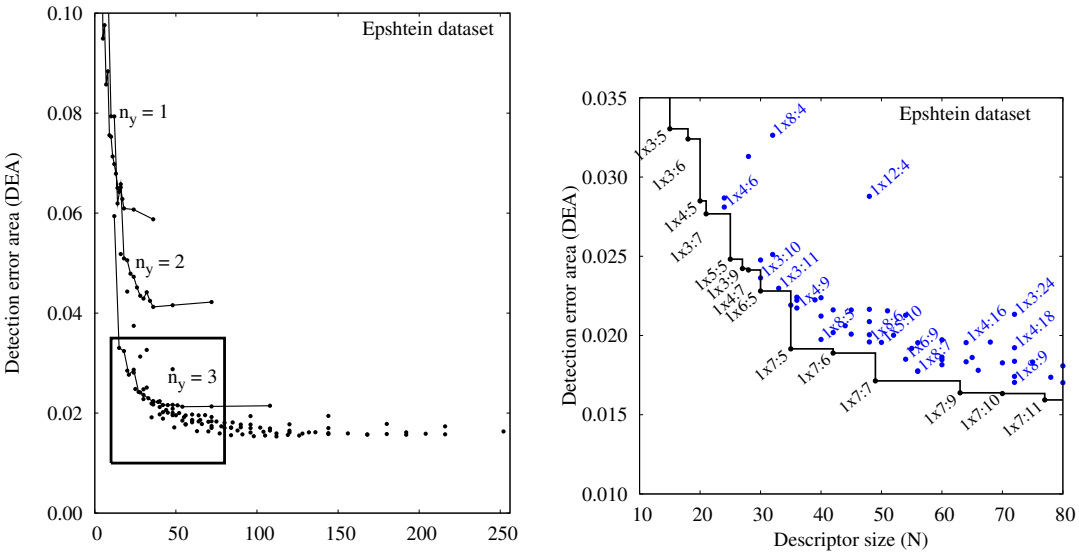


Figure 15: Detection error area A (mean of 10 random partitions) of R-HOG as a function of descriptor size $N = n_x n_y n_b$, for various combinations n_y and n_b, with $n_x = 1$. In the left plot, arrangements with the same n_y (1, 2 or 3) and increasing n_b are connected by lines. The outlined region is magnified in the right plot. The staircase curve connects the optimal configurations (black dots).

Furthermore, the optimal configurations for the other two datasets are slightly different.

A similar series of tests were performed to determine the best combination of n_y and n_b for the T-HOG classifier. We found that the optimum combinations for each N were generally the same as those of R-HOG (see the next section).

4.7. Comparison of T-HOG vs. R-HOG

Figures 16 and 17 compare the accuracy of the R-HOG and T-HOG classifiers, in the optimal n_y and n_b configurations, for each descriptor size N and for each of the three datasets. As we can see, the T-HOG significantly outperforms R-HOG in all cases. For example, a T-HOG with about 20 features has a performance similar to an R-HOG with 80 or more features.

Table 2 gives detailed data for two cell grid and bin count combinations (1×4:5, $N = 20$, and 1×7:9, $N = 63$), selected among the optimum combinations of figures 16 and 17. According

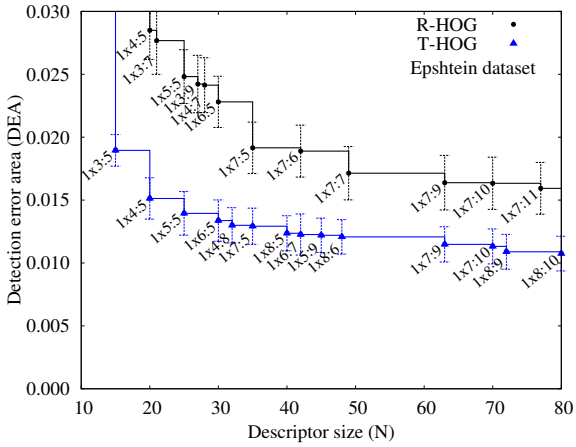


Figure 16: Comparison between optimal configurations of R-HOG and T-HOG. The error bars show the standard deviation over 10 random partitions of (X_i, B_i) on the Epshtein dataset.

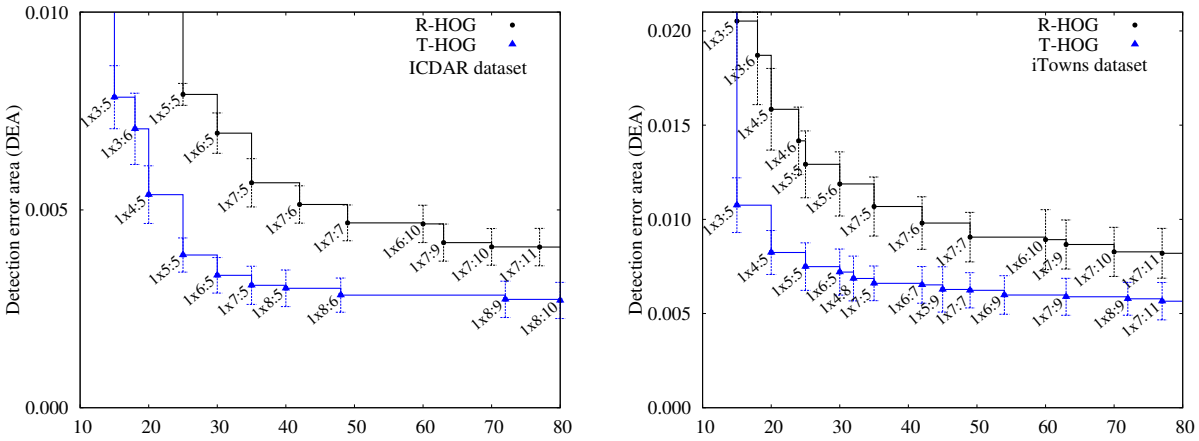


Figure 17: Comparison between optimal configurations of R-HOG and T-HOG. The error bars show the standard deviation over 10 random partitions of (X_i, B_i) on the ICDAR and iTowns datasets, respectively. Note that the vertical scale is different in each plot.

to Student's table for $2L - 2 = 18$ degrees of freedom, the smallest t value in the table, 5.44, corresponds to a risk $\alpha < 10^{-4}$.

Figure 16 shows that, for both classifiers, the *ICDAR-derived* dataset is significantly easier than the other two. Presumably this is due to the fact that most ICDAR images are digitized 2D

$n_x \times n_y$:n_b	Dataset	R-HOG		T-HOG		t-test
		$\boldsymbol{\mu}(A)$	$\boldsymbol{\sigma}(A)$	$\boldsymbol{\mu}(A)$	$\boldsymbol{\sigma}(A)$	
1×4:5	ICDAR	0.0109	0.0010	0.0054	0.0007	14.73
	iTowns	0.0158	0.0022	0.0082	0.0012	10.19
	Epshtein	0.0285	0.0023	0.0151	0.0016	15.10
1×7:9	ICDAR	0.0042	0.0005	0.0029	0.0005	5.81
	iTowns	0.0087	0.0013	0.0059	0.0010	5.44
	Epshtein	0.0164	0.0022	0.0120	0.0014	6.14

Table 2: Statistics of R-HOG and T-HOG classifiers for two optimal cell configurations.

documents, whereas the iTowns and Epshtein images are photos of 3D urban scenes.

4.8. Blurred vs. *hard-edged cells*

Finally, we performed another series of tests to quantify the contribution of blurred cell boundaries to the T-HOG performance. Detailed data for two specific configurations (1×4:5, $N = 20$, and $1\times7:9$, $N = 63$) on the iTowns dataset are shown in table 3. According to Students t-test, the improvement is significant (at risk level $\alpha = 0.05$) for the $1\times4:5$ descriptor ($t = 8.42$), but not for the 1×7:9 descriptor ($t = 1.47$).

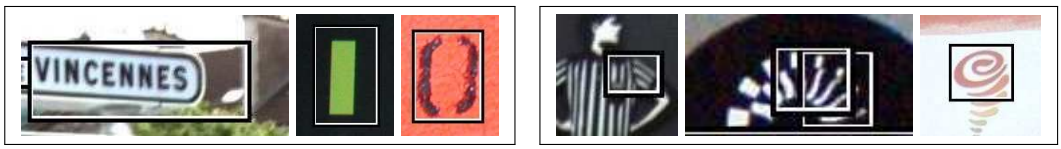
$n_x \times n_y$:n_b	Dataset	Sharp		Blurred		t-test
		$\boldsymbol{\mu}(A)$	$\boldsymbol{\sigma}(A)$	$\boldsymbol{\mu}(A)$	$\boldsymbol{\sigma}(A)$	
1×4:5	iTowns	0.0130	0.0014	0.0082	0.0012	8.42
1×7:9	iTowns	0.0065	0.0008	0.0059	0.0001	1.47

Table 3: Statistics for two optimal T-HOG classifiers with sharp and blurred cells.

4.9. Limitations

Figure 18 shows some false negatives and false positives reported by the T-HOG classifier (in the 1×7:9 configuration) for the X_i and B_i datasets. False negatives are usually due to an inaccurate

detection of the candidate sub-image, or to those texts that are one or two characters long, obscured, or with incomplete characters. False positives are typically images with many line-like features in several orientations.



(a) False negatives (b) False positives

Figure 18: Examples of sub-images incorrectly classified by the T-HOG.

5. Applications

5.1. T-HOG as a post-filter to text detection

The motivating application for text classifiers such as T-HOG and R-HOG is the detection of text in photos and videos of arbitrary scenes [29, 30]. Specifically, the idea is to use the classifier to filter the output of a fast but "permissive" (high-recall and moderate-precision) detector.

To evaluate the suitability of T-HOG for this application we used the SnooperText detector of Minetto et al. [3], which was developed within the iTowns urban documentation and navigation project [26]. SnooperText uses a multiscale adaptive segmentation to locate candidate characters, which are selected and grouped into words and lines by geometrical criteria. Two critical parameters of SnooperText are the minimum size λ (in pixels) of the detected character regions in each scale, and the minimum number of characters per group (GOC). We found that the optimal values of these parameters, when SnooperText was used alone, were $\lambda = 10$ and $GOC = 3$. That is, only words with 3 or more characters were reported. These settings are denoted ST3 in what follows.

When SnooperText was used in combination with the R-HOG or T-HOG as a post-filter, we found that the optimum parameters were $\lambda = 5$ and $GOC = 2$, which increase the recall but

significantly reduce the precision. We denote these settings by ST2. For the T-HOG and R-HOG we used the optimal parameters specified in section 4.4, with the cell arrangement $n_x = 1, n_y = 7$, and $n_b = 9$, resulting in a descriptor of size $N = 63$. See figure 19.

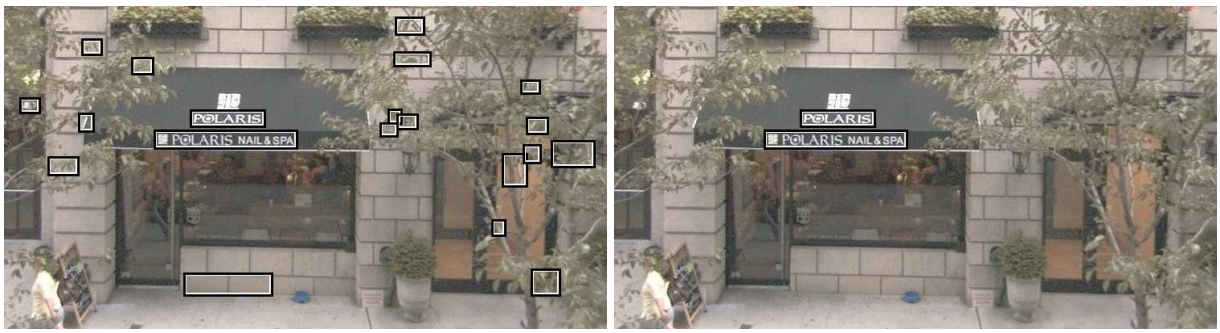


Figure 19: Output of the SnooperText detector with $\lambda = 5$ and $GOC = 2$ (left), and the same output after filtering with the T-HOG recognizer (right) on an image from the Epshtein collection.

5.1.1. Metrics for text detection

The standard metrics to compare text detection systems described in the literature are based on the ICDAR 2005 measure of similarity [25] between two rectangles r, s, and defined as $m(r, s) = S(r \cap s)/S(r \cup s)$ where $S(t)$ 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). The metric m is extended to a set of rectangles Z by the formula $m(r, Z) = \max\{m(r, s') : s' \in Z\}$. From this indicator one derives the ICDAR *precision* p and *recall* r scores [25]

$$p = \frac{\sum_{r \in E} m(r, T)}{|E|} \qquad r = \frac{\sum_{r \in T} m(r, E)}{|T|} \tag{7}$$

where T is the set of manually identified text regions in the input images, and E is the set of text regions reported by the detector. For ranking purposes, the ICDAR 2005 committee used the f *measure* [25] which is the harmonic mean of precision and recall $f = 2/(1/p + 1/r)$. There are several ways of averaging these metrics over a multi-image database. The approach used by

the ICDAR 2005 scoring program (method I) is to evaluate p, r and f separately for each image, and then compute the arithmetic mean of the f-scores over all images. Another approach (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. We note that the first method suffers from higher sampling noise and a negative bias compared to the other method. These points must be considered when comparing f values reported by different authors.

5.1.2. Results

We compared the performance of SnooperText alone and in combination with a text recognizer, either R-HOG or T-HOG, as a post-filter. We also compared them with several state-of-the-art detectors described in the literature. Specifically, we compared with the published scores of the detectors that entered the ICDAR 2003 and 2005 Challenges [25], and also with the detectors of Tian et al. [31] and Epshtein et al. [4], which had the highest f-scores reported in the literature (as of 2011).

The results for each of the three image collections are shown in tables 4–6. All p and r scores were computed with the ICDAR scoring program [25]. The scores in the f_I and f_{II} columns were averaged by methods I and II, respectively.

Note that T-HOG is a more effective post-filter for SnooperText than R-HOG; and that the best combination (ST2+T-HOG) is much better than the best results of SnooperText alone (ST3). Also note that the combination ST2+T-HOG is at least as effective as the best published methods on the ICDAR dataset, and outperforms the Stroke Width Transform (SWT) [4] results of Epshtein et al. on their own dataset. The largest difference between the R-HOG and T-HOG classifications is in the Epshtein dataset, where the text candidates are much harder to classify (see the vertical plot scales in figures 16 and 17).

To confirm the results of section 4.5, we tested the ST2+T-HOG combination with a smaller descriptor ($n_x = 1, n_y = 4$, $n_b = 5$, $N = 20$). The f-score was about 1 to 2% lower, on average,

System	p	r	f_I	f_{II}
ST2+T-HOG	0.73	0.61	0.65	0.67
ST2+R-HOG*	0.70	0.62	0.64	0.66
Yi and Tian [31]	0.71	0.62	0.62	0.66
Epshtein et al. [4]	0.73	0.60	0.66	0.66
ST3+T-HOG	0.72	0.57	0.62	0.64
ST3+R-HOG*	0.72	0.57	0.62	0.64
Hinnerk Becker†	0.62	0.67	0.62	0.64
ST3	0.64	0.59	0.59	0.61
Alex Chen†	0.60	0.60	0.58	0.60
ST2	0.42	0.65	0.47	0.51
Ashida†	0.55	0.46	0.50	0.50
HWDavid†	0.44	0.46	0.45	0.45
Wolf†	0.30	0.44	0.35	0.36
Qiang Zhu†	0.33	0.40	0.33	0.36
Jisoo Kim†	0.22	0.28	0.22	0.25
Nobuo Ezaki†	0.18	0.36	0.22	0.24
Todoran†	0.19	0.18	0.18	0.19
Full†	0.01	0.06	0.08	0.02

Table 4: Performances of various text detectors on the "testing" subset of the ICDAR image collection. The competitors of the ICDAR 2003 and 2005 Challenges are marked with †. For this table, the T-HOG and R-HOG classifiers were trained on the output of the ST2 detector applied to the ICDAR "training" subset. *Best R-HOG found using only horizontal cuts.

for the three image collections. We also tested the ST2+T-HOG combination with the sub-image divided into vertical stripes ($n_x = 7, n_y = 1$, $n_b = 9$, $N = 63$); the f-score was about 5 to 7% lower.

System	p	r	f_I	f_{II}
ST2+T-HOG	0.72	0.50	0.56	0.59
ST2+R-HOG*	0.70	0.49	0.54	0.58
ST3+T-HOG	0.72	0.43	0.51	0.54
ST3+R-HOG*	0.72	0.43	0.51	0.54
ST3	0.49	0.43	0.43	0.46
ST2	0.24	0.53	0.31	0.33

Table 5: Performances of SnooperText, with and without HOG post-filtering, on the whole iTowns image collection. For this table, the R-HOG and T-HOG classifiers were trained on the $X_1 \cup X_3$ and $B_1 \cup B_3$ datasets. *Best R-HOG found using only horizontal cuts.

System	p	r	f_I	f_{II}
ST2+T-HOG	0.59	0.47	0.49	0.52
ST2+R-HOG*	0.55	0.44	0.46	0.49
ST3+T-HOG	0.64	0.39	0.46	0.49
Epshtein et al. [4]	0.54	0.42	—	0.47
ST3+R-HOG*	0.62	0.37	0.43	0.46
ST3	0.46	0.42	0.41	0.44
ST2	0.19	0.54	0.25	0.28

Table 6: Performances of SnooperText, with and without HOG post-filtering, and of the Epshtein et al. detector on the whole Epshtein image collection. For this table, the R-HOG and T-HOG classifiers were trained on the $X_1 \cup X_2$ and $B_1 \cup B_2$ datasets. *Best R-HOG found using only horizontal cuts.

5.2. T-HOG as a text detector

Any text recognizer can also be used on its own as a sliding-window text detector. Namely, the recognizer is applied to a sufficiently large set of sub-regions in the input image, and the sub-regions with the largest scores are returned as the output.

Figure 20 shows the result of such a text detector, using the T-HOG+SVM recognizer, with a

window of fixed size (24 by 72 pixels) sliding over the whole image. Note the high selectivity of the recognizer. For this test, we trained the SVM classifier using the set U of positive "ground-truth" sub-regions provided by the ICDAR Challenge team [25], and a set V of negative random sub-regions of the ICDAR images disjoint from the set U and about three times its size.

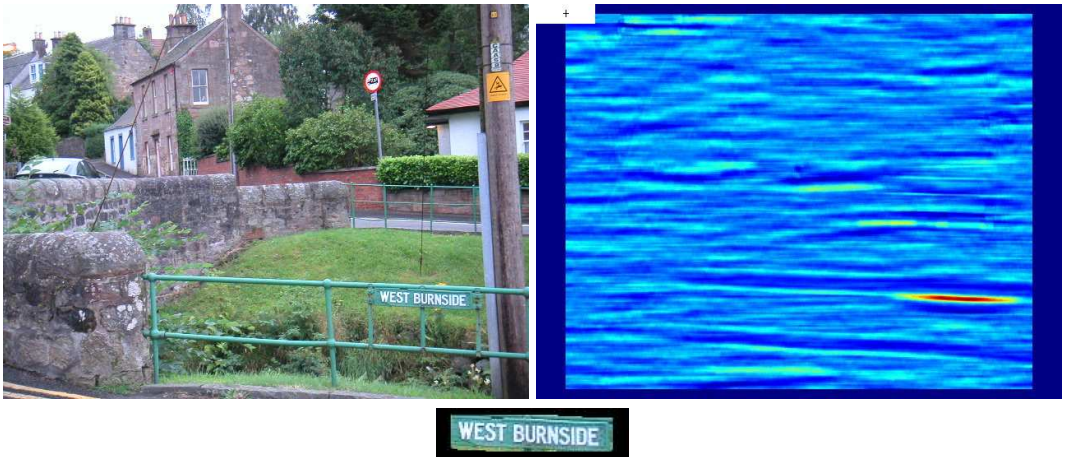


Figure 20: Top left: image PICT0031 from ICDAR dataset (640×480 pixels). Top right: the output of the T-HOG+SVM sliding window classifier, encoded as the color of the central pixel (warm tones for positive output, cold tones for negative). The white rectangle at the top left corner shows the size of the sliding window. Bottom: the union of the 100 windows with the highest scores.

Text of variable size can be detected by running this algorithm on several reduced versions of the input image, in a multi-scale fashion. However, this brute-force approach to text detection is extremely expensive, since the number of windows that need to be analyzed is very large. For this reason we did not evaluate its accuracy or compare it to other detectors.

5.3. T-HOG as a detection post-filter in OCR algorithms

OCR algorithms designed for unstructured 3D urban environments are of great interest to systems as the Google's Street View and the iTowns projects, which aim to extract (offline) any textual

information present in the images, such as street and traffic signs, store names, and building numbers. With this information the user can then make textual queries to retrieve images semantically relevant to him. However, standard OCR algorithms developed for scanned documents perform very poorly on photos of 3D scenes. See figure 21(top). Much better results are obtained by filtering the false positives with T-HOG. See figure 21(bottom).

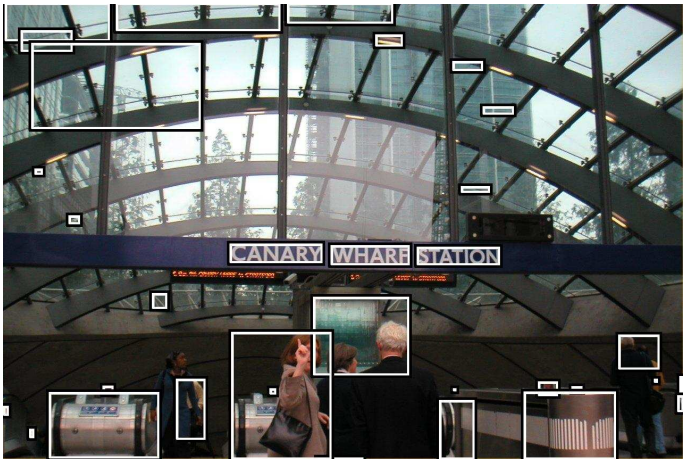
6. Conclusions

In this paper we describe extensive experiments with Dalal and Triggs's multiple HOG descriptor (R-HOG) and SVM classification for the text/non-text discrimination problem. These experiments showed that the optimum cell configuration, for any descriptor size, consists of horizontal bands in a single weighting and normalization block. Splitting the sub-image by vertical cuts is never cost-effective. In retrospect, this conclusion makes sense, given the nature of the 'object' to be classified – a single line of text of arbitrary length. Through these experiments, we also determined the best values for the number of cells n_y and the number of bins n_b, for each descriptor size $N = n_y \times n_b$. In particular, we found that increasing N beyond 100 has practically no effect on classification accuracy.

We then defined another multiple HOG descriptor, the T-HOG, whose cells have blurred boundaries defined by overlapping Gaussian weight functions. An exhaustive series of experiments confirmed that the best cell arrangement for the T-HOG text classifier is also a stack of horizontal bands. These tests also showed that the T-HOG classifier consistently outperforms R-HOG at text/non-text discrimination, for any descriptor size N.

Finally, we described the use of T-HOG in three text-related applications. First, we described the use of T-HOG as a post-filter for a high-recall, low-precision text detector, and showed that the combination is at least as good as the best text detectors reported in the literature. We also showed that T-HOG is better than R-HOG for this application. Second, we described the use of T-HOG in

Tesseract text detection



Tesseract OCR

```
-3’5
3?
HM! :==" ’F"1-’
-W *
’?!
’—
I
I
L4-
-=w
CANARY WHARF STATION
\ ’1! ~
rf. -  *5.  ’ *
’ . ’ %\  \}\ -:@\|\W X
":1
M M ,,;’1
.’_&M
```

Tesseract text detection + T-HOG

Tesseract OCR

```
CANARY WHARF STATION
```

Figure 21: Top left: the text detection of a top publicly available OCR (Tesseract). Top right: OCR of Tesseract with few readable words and a lot of noise due to the false detections. Note that it is hard, in an urban context, to filter the strings with a dictionary since noise regions can be converted to words with meaning. Bottom left: the text detection output of Tesseract after filtering the candidates with the T-HOG classifier. Bottom right: the OCR after filtering.

a sliding-window text detector, and gave anecdotal evidence of its accuracy. Third, we described the benefits of T-HOG in a well-known OCR software.

References

[1] R. Minetto, Text-HOG (T-HOG)., http://www.dainf.ct.utfpr.edu.br/~rminetto/projects/thog.html (2012).

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

[3] R. Minetto, N. Thome, M. Cord, J. Fabrizio, B. Marcotegui, Snoopertext: A multiresolution system for text detection in complex visual scenes, in: Proc. IEEE Int. Conf. on Image Processing - ICIP, 2010, pp. 3861–3864.

[4] B. Epshtein, E. Ofek, Y. Wexler, Detecting text in natural scenes with stroke width transform, in: Proc. Int. Conf. on Computer Vision and Pattern Recognition - CVPR, IEEE Computer Society, 2010, pp. 886–893.

[5] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: Proc. Int. Conf. on Computer Vision and Pattern Recognition - CVPR, 2005, pp. 886–893.

[6] W. Zhang, G. Zelinsky, D. Samaras, Real-time accurate object detection using multiple resolutions, in: Proc. Int. Conf. on Computer Vision - ICCV, 2007, pp. 1–8.

[7] X. Chen, A. L. Yuille, Detecting and reading text in natural scenes, in: Proc. Int. Conf. on Computer Vision and Pattern Recognition - CVPR, Vol. 2, IEEE Computer Society, Los Alamitos, CA, USA, 2004, pp. 366–373.

[8] S. M. Hanif, L. Prevost, Text detection and localization in complex scene images using constrained AdaBoost algorithm, in: Proc. Int. Conf. on Document Analysis and Recognition - ICDAR, IEEE Computer Society, Washington, DC, USA, 2009, pp. 1–5.

[9] Y.-F. Pan, X. Hou, C.-L. Liu, A robust system to detect and localize texts in natural scene images, in: Proc. Int. Workshop on Document Analysis Systems, IEEE Computer Society, Washington, DC, USA, 2008, pp. 35–42.

[10] X. Wang, L. Huang, C. Liu, A new block partitioned text feature for text verification, IEEE Computer Society, Los Alamitos, CA, USA, 2009, pp. 366–370.

[11] G. Louloudis, B. Gatos, I. Pratikakis, C. Halatsis, Text line detection in handwritten documents, Pattern Recognition, Elsevier 41 (12) (2008) 3758–3772.

[12] Y. Leydier, F. Lebourgeois, H. Emptoz, Text search for medieval manuscript images, Pattern Recognition, Elsevier 40 (12) (2007) 3552–3567.

[13] I. Giannoukos, C.-N. Anagnostopoulos, V. Loumos, E. Kayafas, Operator context scanning to support high segmentation rates for real time license plate recognition, Pattern Recognition, Elsevier 43 (11) (2010) 3866–3878.

[14] N. Thome, A. Vacavant, L. Robinault, S. Miguet, A cognitive and video-based approach for multinational license plate recognition, Machine Vision and Applications 22 (2) (2011) 389–407.

[15] N. Sharma, U. Pal, M. Blumenstein, Recent advances in video based document processing: A review, in: Proc. IAPR International Workshop on Document Analysis Systems, IEEE Computer Society, Los Alamitos, CA, USA, 2012, pp. 63–68.

[16] D. Chen, J.-M. Odobez, H. Bourlard, Text detection and recognition in images and video frames, Pattern Recognition, Elsevier 37 (3) (2004) 595–608.

[17] 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: Proc. Int. Conf. on Pattern Recognition - ICPR, Vol. 2, 2004, pp. 679–682.

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

[19] M. Zhao, S. Li, J. Kwok, Text detection in images using sparse representation with discriminative dictionaries, Image and Vision Computing, Elsevier 28 (12) (2010) 1590–1599.

[20] P. Shivakumara, W. Huang, T. Q. Phan, C. L. Tan, Accurate video text detection through classification of low and high contrast images, Pattern Recognition, Elsevier 43 (6) (2010) 2165–2185.

[21] M. Anthimopoulos, B. Gatos, I. Pratikakis, A two-stage scheme for text detection in video images, Image and Vision Computing, Elsevier 28 (9) (2010) 1413–1426.

[22] T. Ojala, M. Pietikäinen, T. Mäenpää, Multiresolution gray-scale and rotation invariant texture classification with local binary patterns, IEEE Trans. on Pattern Analysis and Mach. Intell. 24 (2002) 971–987.

[23] M. D. Fairchild, Color Appearance Models, Wiley-IST Series in Imaging Science and Technology, Chichester, UK, 2005.

[24] D. Picard, N. Thome, M. Cord, An efficient system for combining complementary kernels in complex visual categorization tasks, in: Image Processing (ICIP), 2010 17th IEEE International Conference on, IEEE, 2010, pp. 3877–3880.

[25] S. Lucas, ICDAR 2005 Text locating competition results, in: Proc. Int. Conf. on Document Analysis and Recognition - ICDAR, Vol. 1, 2005, pp. 80–84.

[26] A. N. de Recherche, The iTowns project, `http://www.itowns.fr` (2009).

[27] A. Martin, G. Doddington, T. Kamm, M. Ordowski, M. Przybocki, The DET curve in assessment of detection task performance, in: Proc. Eurospeech, 1997, pp. 1895–1898.

[28] K. Turkowski, Filters for common resampling tasks, in: A. S. Glassner (Ed.), Graphics Gems, Academic Press Professional, Inc., San Diego, CA, USA, 1990, pp. 147–165.

[29] R. Minetto, N. Thome, M. Cord, J. Stolfi, F. Precioso, J. Guyomard, N. J. Leite, Text Detection and Recognition in Urban Scenes, in: International Conference on Computer Vision (ICCV): Workshop on Computer Vision for Remote Sensing of the Environment, 2011, pp. 227–234.

[30] R. Minetto, N. Thome, M. Cord, N. J. Leite, J. Stolfi, Snoopertrack: Text Dectection and Tracking for outdoor Videos, in: 18th IEEE International Conference on Image Processing (ICIP 2011), 2011.

[31] C. Yi, Y. Tian, Text string detection from natural scenes by structure-based partition and grouping, IEEE Trans. on Image Processing 20 (9) (2011) 2594–2605.