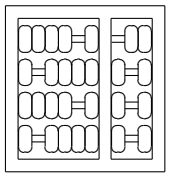


INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

Fuzzy Histogram of Oriented Gradients to characterize Single Line Text Regions

R. Minetto *N. Thome* *M. Cord*
N.J Leite *J. Stolfi*

Technical Report - IC-11-10 - Relatório Técnico

May - 2011 - Maio

Fuzzy Histogram of Oriented Gradients to characterize Single Line Text Regions

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

Abstract

In this work we discuss the use of the *histogram of oriented gradients* (HOG) descriptors as an effective tool for text description and recognition. Specifically, we propose a **F**uzzy **HOG**-based texture descriptor (F-HOG) that uses a partition of the image into three horizontal cells with fuzzy adaptive boundaries, to characterize single-line texts in outdoor scenes and video frames. The input of our algorithm is a rectangular image presumed to contain a single line of text in latin like characters. The output is a relatively short (54-features) descriptor that provides an effective input to an SVM classifier. Tests show that F-HOG is more accurate than Dalal and Triggs original HOG-based classifier using a 54-features descriptor, and comparable to their best classifier (which uses a 108-features descriptor) while being half as long.

1 Introduction

In this work we discuss the use of the *histogram of oriented gradients* (HOG) [5] for the purpose of recognizing lines of text in digital images and video frames. Specifically, we describe a novel text descriptor, F-HOG, that efficiently and accurately characterize single-line texts.

The input of F-HOG algorithm is a rectangular image presumed to contain a single line of text in latin like characters. The output is a relatively short (54-component) descriptor that is fed to a *support vector machine* (SVM) classifier.

The F-HOG descriptor is the concatenation of three HOGs computed for three slices of the image, with fuzzy adaptive boundaries. Fuzzy adaptive boundaries are important for outdoor images, which often have text with non-uniform, rotated, tilted or curved baselines, mixed lower and upper case characters, or inaccurate bounding boxes.

We compared our F-HOG descriptor to the HOG-based descriptors defined by Dalal and Triggs [5]. Our tests show that the F-HOG is more accurate than their best 54-features descriptor, and comparable to their best one (108-features) descriptor (both consistent with our model).

We expect that F-HOG can be useful in many real time applications, such as text detection, text tracking and OCR. We used it as a hypothesis-validation filter (to remove non-text regions) in a

*This work was partially supported by FAPESP (Phd grant 07/54201-6), CAPES/COFECUB 592/08 and ANR 07-MDCO-007-03. (1) University of Campinas, UNICAMP, Av. Albert Einstein, 1251, Campinas-SP, Brazil. (2) Universite Pierre et Marie Curie, UPMC-Sorbonne Universities, LIP 6, 4 place Jussieu, 75005, Paris, France. Contact: rodrigo.minetto@gmail.com.

recent published text detector [2]. The performances obtained after the filtering were comparable to an state of the art text detector [6].

1.1 Statement of the problem

We consider here images obtained from a physical scene or artifact by a scanner or digital camera opposed to synthetic images generated by typesetting or computer graphics software. A *text object* is any part of the physical scene carrying a string of two or more letters that are easily recognizable and readable in the captured image. We are primarily concerned with texts written in the Roman alphabet or any of its variants, in plain print-style (rather than cursive or ornate) fonts, with fairly "normal" geometry (straight baseline, on smooth surfaces that are flat or moderately curved). See figure 1.

Figure 1: Urban scene with recognizable text objects.

Our method assumes that the candidate text object has been identified and its image has been bounded by a rectangle. Futhermore, it assumes that the text consists of a single multi-character line, as in figures 2(b,c). Isolated characters and multiline text blocks, as in figures 2(a,d) should be joined or split into separate lines or words.

(a) (b) (c) (d)

Figure 2: Alternative segmentations of text objects: (a) characters, (b) words, (c) lines and (d) blocks.

Finally, our method is designed for texts that contain a normal mix of characters. It may perform poorly for isolated letters or for texts that consist of many similar letters.

1.2 Descriptor outline

Our solution builds upon the general-purpose texture descriptor known as histogram of oriented gradients (HOG), developed by Dalal and Triggs [5]. The HOG descriptor is based on the idea 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 the strokes that make up text letters) then the strongest gradients tell the orientations of the edges of those shapes. HOG-based

descriptors have been successfully used for the recognition of pedestrians [5], objects [19] and for text detection [8, 13, 17, 18].

The first step of F-HOG algorithm is to extract and normalize the candidate text region. In this step we scale the candidate region to a fixed height h, maintaining its original aspect ratio. The height h should be sufficiently large to retain the identities of the characters, while suppressing spurious detail. We also convert the image from color to gray scale, since the human eye-brain uses only the brightness channel to recognize character shapes [7]. We then apply a mean-and-variance normalization to the image values (with a Gaussian-like weight kernel) in order to remove artifacts due to non-uniform illumination.

As observed by Chen and Yuille [3], the top, middle and bottom parts of Roman characters have distinctive distributions of edge directions. Therefore, in the F-HOG algorithm we split the normalized image into three horizontal slices. For each pixel we compute the local image gradient, and we build a histogram of the gradient directions for each slice. Furthermore, the slices are not sharp-edged but "fuzzy", defined by smoothly varying weight functions w_0, w_1, w_2. The complete descriptor is the concatenation of these three HOGs. See figure 3.

The remainder of the paper is organized as follows. In section 2 we discuss some previous work. In section 3 we present the basic HOG and our F-HOG. In section 4 we experimentally compare F-HOG to the Dalal and Triggs descriptors. Finally in section 5 we state the conclusions.

2 Previous work

In 2004 Chen and Yuille [3] described a text line detector that uses 2D histograms of image intensity and gradient norm computed separately for top, middle and bottom of the text region, as well as for more complex slices subdivisions of the image. The Chen and Yuille descriptor has 79 features (composed by 5 distinct types of features based on derivatives, histogram, edges, etc.).

In 2008 Pan et al. [13] described a text line detector that partitions the image into 14 blocks, as proposed by Chen and Yuille, but they compute for each block a 4-bin HOG complemented by a 2×3 array of *local binary patterns* [12] (LBP). Their complete descriptor has a total of 140 features. They claimed that the HOG alone do not discriminate between text and busy backgrounds such as vegetation or fences (a claim that we found to be exaggerated, see section 3).

Others HOG-based text recognizers have been proposed in 2009 by Hanif and Prevost [8] for single-line text blocks and Wang et al. [17] for isolated Chinese and Roman characters as well as single-line text blocks. Both methods partition the text area into blocks in vertical as well as horizontal cuts. The use of vertical cuts was apparently borrowed from the Dalal and Triggs paper [5] on pedestrian recognition. Vertical cuts may be justifiable for isolated characters, but they do not appear to be useful for multicharacter texts (especially if the vertical cuts ignores the character boundaries), while they significantly increase the size of the descriptor. Hanif and Prevost's descriptor has 151 features (16 blocks each with a 8-bin HOG, supplemented by 7 mean difference and 6 standard deviation features over 16 blocks). The descriptor of Wang et al has 80 features (8 blocks with 8-bin HOG and 1 mean difference feature and 1 standard deviation).

In 2010 Zhang and Kasturi [18] described an approach to identity characters edges. They noted that the edges of a stroke typically generates pairs of closely spaced parallel edges with opposite orientation. So, they build a 4-bin HOG from the orientations of edges obtained by the Canny

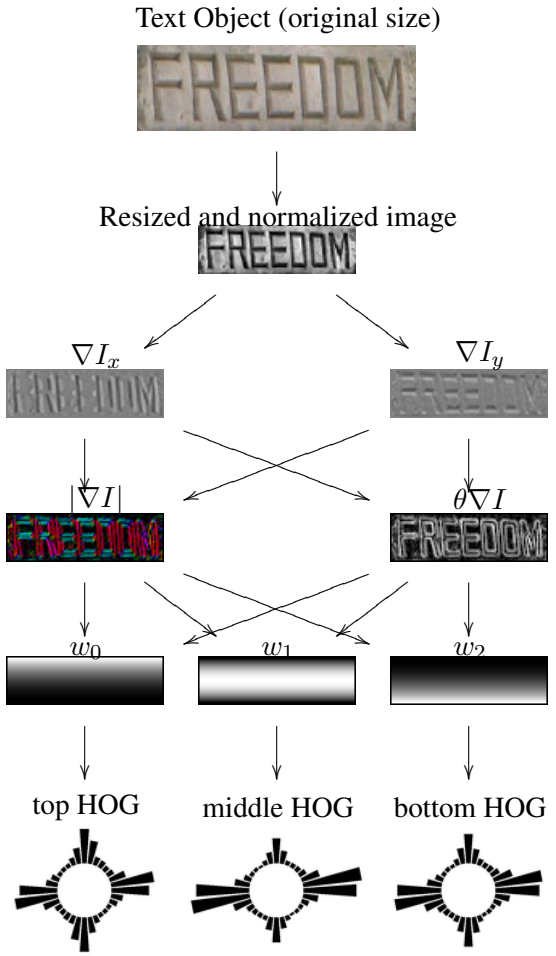


Figure 3: Fuzzy HOG text descriptor scheme. For clarity, the histograms are duplicated in the plots so as to cover the full range 0 to 2π.

detector, and require that bins corresponding to opposite orientations be nearly equal. Is important to note that there are many algorithms [15, 9, 14] that are designed to work as *text detectors* rather than *text classifier*. These algorithms take as input a whole image and return a list of sub-regions that are likely to contain text; therefore they are not directly comparable to text classifiers like F-HOG, which assume that the candidate regions are given as input. A recent example of text detector is the algorithm by Epshtein et al [6] in 2010. They use gradient orientation over image edges to determine a "local stroke width" at each pixel and then gather pixels with similar stroke widths into regions which are likely to be characters.

3 Histogram of oriented gradients

3.1 The basic HOG descriptor

To compute the HOG descriptor of an image, one computes the local gradient ∇I at each pixel. Then one builds a histogram of the directions $\theta(\nabla I)$ of the gradients, quantized into a small number of bins. For text recognition, one often considers opposite directions as equivalent (so that they range over $[0, \pi]$ radians rather than $[0, 2\pi]$). The contribution of each pixel to the histogram has weight $|\nabla I|$, so that the random gradient directions that occur in flat parts of the image do not contribute to the histogram. The histogram is scaled so that the sum of all entries is 1.

Figure 4 shows the HOGs of a few isolated letters. These HOGs have 8 bins, each with $\pi/8$ radians wide, centered at orientations $k\pi/8$ for $k = 0, 1, \ldots, 7$.

Note that the HOG gives the predominant orientation of the letter strokes. In particular the histogram of a rounded letter like 'O' is almost uniform over the whole range $[0, \pi]$, while that of 'I' has a spike at the directions perpendicular to the letter's stem.

3.2 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. Therefore, in many applications, the candidate region is partitioned into an array of *cells*, and a HOG is computed separately for each cell. The concatenation of those HOGs is taken to be the descriptor of the full region.

For single-line text images, formed by Roman characters, it makes sense to analyze separately the distributions of edge directions in the top, middle and bottom parts. As shown in figure 5, it is expected that all three parts contain mostly vertical or horizontal strokes, so that the gradients orientations are predominantly 0 (or 180) and 90 (or 270) degrees. The top and bottom parts should contain a larger proportion of horizontal strokes, so that the gradients there are pointing mostly in the vertical direction. The midlle part is expected to have a larger proportion of vertical strokes. In all three parts we expected to have a small amount of diagonal strokes from letters such as 'A', 'V', 'Y', etc, and from rounded parts of letters such as 'O', 'S', 'B', etc.

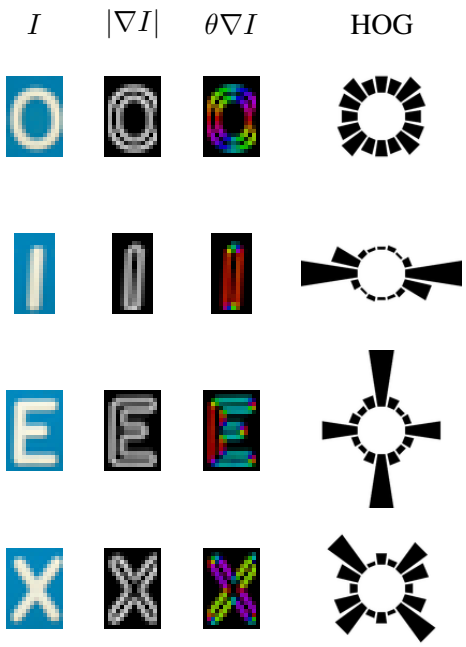


Figure 4: HOGs of some isolated letters. Note that each HOG histogram was duplicated to cover the full range 0 to 2π.

3.3 Fuzzy cells

However, if the cells are defined by sharp boundaries, the HOG may change drastically with small vertical displacements of the text inside its bounding box. To avoid this problem, we use "fuzzy" cells, defined by *weight functions* w_0, w_1 and w_2. The weight w_k of each cell is a function of the relative vertical coordinate

$$z = \frac{y - y_{\text{top}}}{y_{\text{bot}} - y_{\text{top}}} \tag{1}$$

where y is the vertical coordinate of the pixel, and y_{top} and y_{bot} are the estimated y coordinates of the top and bottom contours of the text in the image. The value of $w_k(z)$ is a number that vary smoothly between 0 and 1. When computing the histogram for cell number k, each pixel (x, y) of the normalized text image is assumed to have mass $|\nabla I(x, y)| w_k(z)$.

Recall that each HOG is normalized to unit sum. Therefore, only the shape of each weight function w_k is important. Scaling each w_k by any positive factor will have no effect on the final HOG.

This model is a generalization of hard-edged cells. These can be emulated by defining each w_k to be the appropriate step function. See figure 6. For fuzzy cells we tested different sets of weight functions (Gaussian bell functions, Hann, etc). The best found consists of clipped and scaled

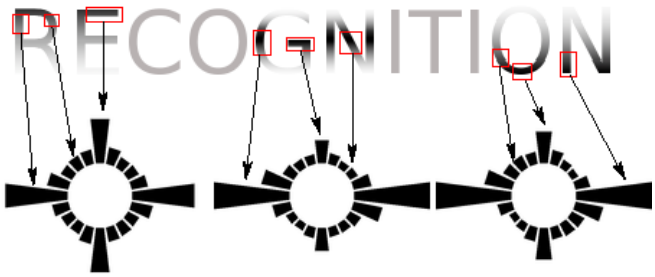


Figure 5: From left to right we have the top, middle and bottom HOGs for the text "RECOGNITION". The arrows show the contribution of specific letters strokes to the final descriptor.

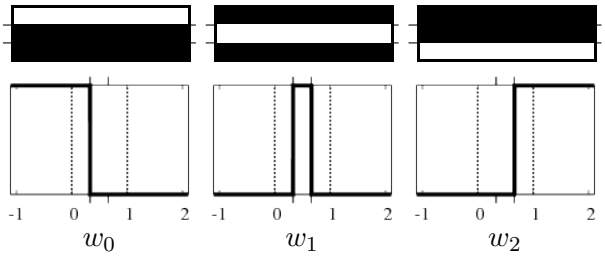


Figure 6: The step weight functions for hard-edged cells. The tic marks and the dotted lines show the ordinates $y_{\text{top}}, y_{\text{bot}}$.

Bernstein polynomials. Namely, for $n+1$ horizontal stripes, we use

$$w_k(z) = \begin{cases} 1 & \text{if } k = 0 \text{ and } z \leq 0 \\ 1 & \text{if } k = n \text{ and } z \geq 1 \\ \beta_k^n(z)/\beta_k^n(k/n) & \text{if } 0 < z < 1 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where

$$\beta_k^n(z) = \binom{n}{k} z^k (1-z)^{n-k} \tag{3}$$

for $k = 0, 1, \ldots, n$. See figure 7.

3.4 Variable text-baselines

Outdoor scenes often have text regions whose 'top' and 'bottom' edges vary along the line. Thus, we adapt the cell boundaries to account for such texts, e.g. with non-uniform, rotated, tilted or curved baselines, with mixed lower and upper case characters or with incorrect bounding. See figure 8. To handle these problems we chose the text baselines $y_{\text{top}}(x)$ and $y_{\text{bot}}(x)$ of equation (1) separately for each column x of the image, so as to fit the local text boundaries. Namely, for each y along a column x we compute a 'local text contents' value $v(y)$ that is the variance of the image inside a

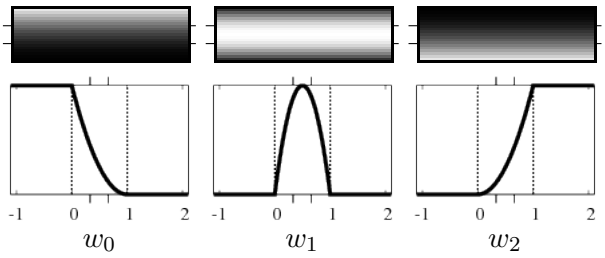


Figure 7: The Bernstein weight functions for $n = 2$.

Figure 8: Three text samples with upper and lower case letters, tilted and curved baselines.

$1 \times t$ window centered at pixel (x, y). We take the ordinates $y_{\text{top}}(x)$ and $y_{\text{bot}}(x)$ for that column as the 20% and 80% percentiles of the v function. See figure 9. We then adjust the fuzzy cell weight

Figure 9: The local baselines $y_{\text{top}}(x)$ and $y_{\text{bot}}(x)$ for the three text samples of figure 8 (black lines).

functions to vary between $[y_{\text{top}}, y_{\text{bot}}]$.

Figure 10 shows the Bernstein weight functions for the examples of figure 8. Note that the areas above y_{top} and bellow y_{bot} are wholly included in the respective HOGs.

However, if we expected some noise involving the text regions, Gaussian weight functions may be the most appropriated. See figure 11. Note that the top and bottom exclude parts of the background that are well above y_{top} or well below y_{bot}.

Figure 12 shows some HOGs to non-text images. Figure 13 shows the HOG insensitiveness to text line splitting, incorrect horizontal placement of bounding box, and mixed fonts (note that in figure 13 the HOG's to different images are very similar). Plots using F-HOG.

4 Experiments

4.1 Datasets

For training and evaluation, we used sets of text regions X_i and $\hat{X}_i$ and non-text "background" regions B_i and $\hat{B}_i$ extracted from three image collections:

1. the 2005 ICDAR challenge collection [11], consisting of 499 color images, captured with different digital cameras and resolutions, of book covers, road signs, household objects, posters,

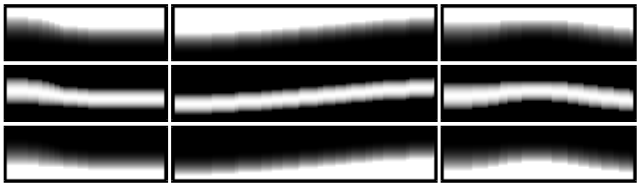


Figure 10: Bernstein weight functions to the variable baselines shown in figure 9.

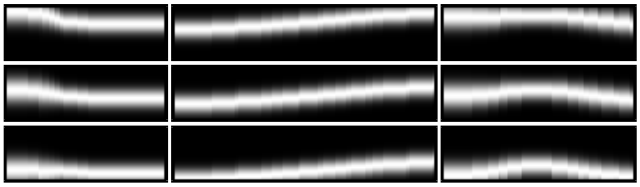


Figure 11: Gaussian weight functions to the variable baselines shown in figure 9.

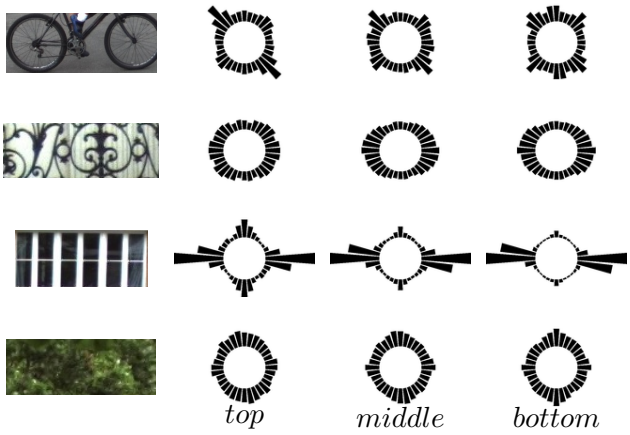


Figure 12: HOG of some non-text images.

etc.. The 'training' sets of the ICDAR challenge provided our set X_1 of text-containing regions. We eliminated from X_1 any regions that contained single characters. We also extracted from the ICDAR training images a set of background (non-text) regions B_1, which were the false positives returned by an experimental text detector algorithm $\hat{D}$ [2]. See figure 14.

2. the iTowns Project collection [1], consisting of several thousand 1080×1920 color images of Parisian façades taken by a camera-equipped vehicle (similar to Google's Street View images). The text detector algorithm $\hat{D}$ was applied to a few hundred images to obtain a set of candidate regions which we manually sorted into text X_2 (the true positives) and non-text B_2 (the false positives). See figure 14.

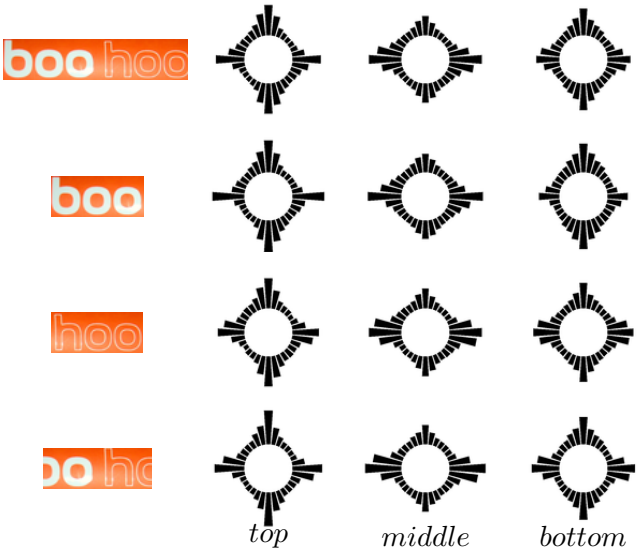


Figure 13: Effect of horizontal displacement. ICDAR image.

3. the Epshtein et al. benchmark [6] with 307 color images ranging from 1024×1360 to 1024×768 pixels of urban scenes taken with hand-held cameras. We did not use this dataset for training so there is no X_3 or B_3.

For each collection we also obtained a publicly available positive test set $\hat{X}_i, i = 1, 2, 3$. [1, 6, 11]. We excluded from Epshtein et al test dataset ($\hat{X}_3$) about 69 regions which had vertical or severely tilted text. For each collection we also generated a test set $\hat{B}_i$ of randomly chosen non-text regions ("background") (B) disjoint from the $\hat{X}_i$ regions. See table 1.

		X	B
Training	ICDAR + iTowns ($X_1 \cup X_2$, $B_1 \cup B_2$)	4918	4047
Testing	ICDAR ($\hat{X}_1, \hat{B}_1$)	1107	5535
	iTowns ($\hat{X}_2, \hat{B}_2$)	1039	5825
	Epshtein ($\hat{X}_3, \hat{B}_3$)	1912	9905

Table 1: Our training and testing datasets.

Figure 14: Positives (top) $(X_1 \cup X_2)$ and negatives (bottom) $(B_1 \cup B_2)$ training samples. From ICDAR and iTowns training sets.

4.2 Metrics

To evaluate the performance in each test we counted the number of true and false positives and true and false negatives (TP, FP, TN and FN). We then computed the well-known precision (p) and recall (r) metrics:

$$p = \frac{TP}{TP + FP} \qquad r = \frac{TP}{TP + FN} \tag{4}$$

We also use the combined f-metric [10], the weighted harmonic mean of p and r, defined as $f = (pr)/(\alpha r + (1 - \alpha)p)$ where α is a weighting coefficient (set to 0.5).

4.3 F-HOG Settings

For all tests we used an SVM classifier with Gaussian χ^2 kernel, and performed a cross-validation to optimize its standard deviation σ parameter. In each experiment (both in training and evaluation phases) we used the Lanczos interpolation (resampling) filter [16] to rescale the dataset images to the fixed height h maintaining the aspect ratio.

For our three stripe algorithm (F-HOG) the best f-performance was obtained with $h = 21$, 18 bins and L1 norm, yielding a 54-features descriptor. However using 9 bins we also have good results (drop of $\%1$ or $\%2$ in the $f-$performance on average) yielding a 27-features descriptor. The trade-off between performance and descriptor size will depend on the application. Finally, increasing the number of bins well above 18 does not bring a significant gain.

In figure 15 we show that the combination F-HOG/SVM clearly discriminated between text and non-text regions. See figure 15.

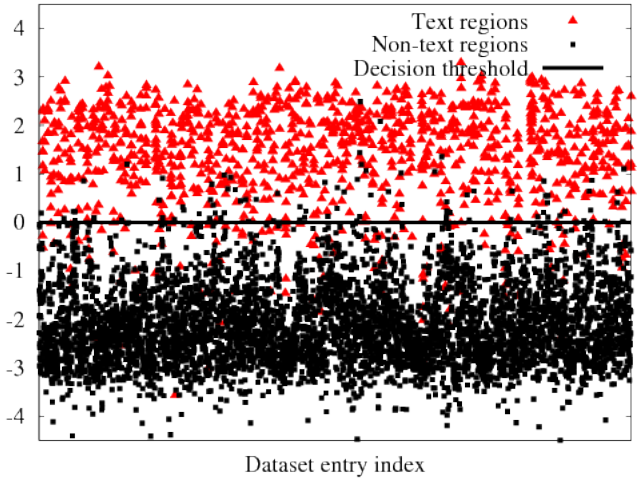


Figure 15: SVM output for the 1017 ICDAR 'testing' test regions ($\hat{X}_1$) in red and the 5535 non-text regions ($\hat{B}_1$) in black, using the F-HOG descriptor.

Figure 16 shows some false positives and false negatives reported by our F-HOG on the ICDAR testing dataset. Note that our algorithm was not designed to recognize single characters. Where our

error probability for randomly chosen non-text regions was about 1% (tests with 10^3, 10^4 and 10^5 randomly sampled non-text regions taken over the whole dataset).

(a) -3.46 (b) -0.02 (c) -0.06 (d) +1.22 (e) +0.01

Figure 16: Some ICDAR test regions incorrectly classified by the F-HOG and their SVM scores (from left to right 3 false negatives and 2 false positives).

4.4 Comparison

We compared our classifier with Dalal and Triggs's original code [4]. In an extensive series of tests, the best f-performance of their classifier was obtained with $h = 24$ (image resizing also with the Lanczos interpolation, using the same training and testing datasets, and with the same SVM configurations), 1×6 single-cell blocks, and 18 bins, yielding a descriptor with 108 features. When restricted to a three-cell partition, the best $f-$performance of Dalal's classifier was observed with $h = 21$ and 18 bins, yielding a 54-feature descriptor. In both cases we used L1 histogram normalization and RGB SQRT image preprocessing. We will denote these two classifiers by Dalal-108 and Dalal-54, respectively. The results of the tests are shown in table 2. As can be seen, the $f-$performance of our F-HOG is significantly better than that of Dalal-54 and comparable to that of Dalal-108. Note that the Dalal-108 descriptor is also consistent with our model (horizontal slices).

According our tests, vertical splittings do not worth the trouble. They produced small descriptors with poorly performances or large descriptors without a substantial gain (e.g. a 5×5 Dalal descriptor, with or without overlap, with 18 bins, had a performance drop of 2% in the $f-$performance (on average) compared to Dalal-108).

4.5 Importance of baselines adjustment

To assess the importance of the adaptive baselines, we also trained and tested a variant of the F-HOG classifier with this step turned off (namely, with $y_{\text{top}} = 0$ and $y_{\text{bot}} = h$). The results are shown in table 2 (row F-HOG-NB). As it can be seen, the baseline adjustment contributed between 1 and 5% to the p, r and f scores of F-HOG.

4.6 Importance of fuzzy cell weights

To assess the importance of fuzzy boundaries, we also trained and tested another variant of our classifier with sharp cells borders (that is, with the 'step' weight functions), still with adaptive baselines. As seen in row F-HOG-NF of table 2, fuzzy cell boundaries contribute about 5% to the f-score of FHOG.

Method	Descriptor	ICDAR			iTowns			Epshtein		
	size	p	r	f	p	r	f	p	r	f
Dalal-108	**108**	**0.92**	**0.87**	**0.90**	**0.94**	**0.84**	**0.89**	**0.93**	**0.83**	**0.88**
Dalal-54	54	0.77	0.82	0.80	0.83	0.84	0.83	0.71	0.86	0.78
F-HOG	**54**	**0.92**	**0.89**	**0.91**	**0.91**	**0.82**	**0.87**	**0.93**	**0.80**	**0.86**
F-HOG-NB	54	0.88	0.88	0.88	0.89	0.77	0.83	0.91	0.75	0.82
F-HOG-NF	54	0.87	0.87	0.87	0.86	0.79	0.82	0.84	0.77	0.80

Table 2: Scores of Dalal's classifiers (Dalal-108 and Dalal-54) and our classifier (FHOG) on the three datasets. Trained with the same sets. Rows FHOG-NB and FHOG-NF are variants of FHOG without adaptive baselines and without fuzzy cells, respectively.

4.7 Using the F-HOG as a text filter

We also tested the F-HOG as a hypothesis-validation filter in a real text detector algorithm $\hat{D}$. The performances of the text detector $\hat{D}$ on the Epshtein dataset ($\hat{X}_3$, $\hat{B}_3$) were $p = 0.38$ and $r = 0.43$. See one output example in figure 17 (top). Using the F-HOG as a filter we obtained $p = 0.55$ and $r = 0.40$. See figure 17 (bottom). These results are comparable to the ones obtained by Epshtein text detector [6] which were $p = 0.54$ and $r = 0.42$.

4.8 Using the F-HOG as a text detector

Finally, figure 18 shows the results of applying the F-HOG/SVM classifier alone as a text detector, by using it to classify a 21 by 63 pixel window, sliding over the whole image. However, to this purpose, it should be applied in a multi-scale fashion, together with some rules to define the text area, etc. (that was not the focus of this paper).

5 Conclusions

In this work we have discussed the use of the histogram of oriented gradients (HOG) for the purpose of text recognition. We also described a novel text descriptor, F-HOG, with only 54 features, that efficiently and accurately characterizes single-line texts by using fuzzy adaptive boundaries as well as the original Dalal and Triggs 108-features descriptor. Our tests also indicate that the F-HOG/SVM classifier may be an effective hypothesis-validation filter for text detectors, or a text detector by itself.

References

[1] iTowns ANR project. http://www.itowns.fr.

[2] Suppressed.

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

[4] N. Dalal. INRIA Object Detection and Localization Toolkit, 2008. http://pascal.inrialpes.fr/soft/olt/.

(a)

(b)

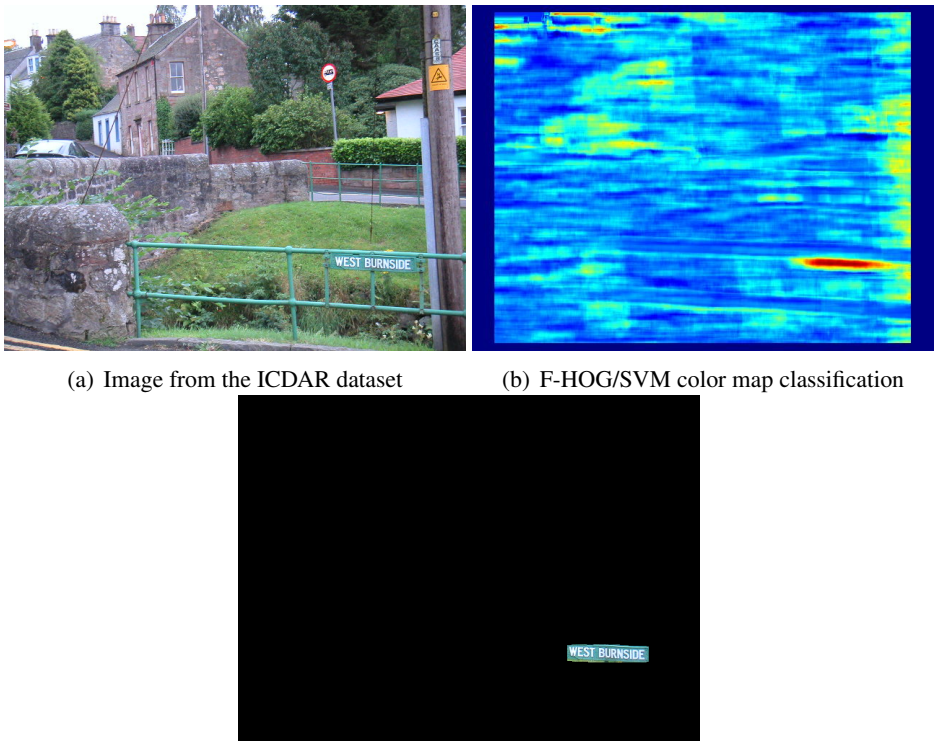
Figure 17: (a) Output of the text detector $\hat{D}$ on the image 289 from the Epshtein dataset. (b) The same after filtering with F-HOG classifier.

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

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

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

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



(a) Image from the ICDAR dataset (b) F-HOG/SVM color map classification

(c) Best regions

Figure 18: (a) Image PICT0031 (640×480 pixels) from ICDAR dataset; (b) The output of the F-HOG/SVM classifier applied to a sliding 21×63 pixel window, where the hot tones indicates positive output (the regions most 'text-like') and the cold tones indicate negative output; (c) The union of the 100 windows with best scores.

[9] W. Kim and C. Kim. A new approach for overlay text detection and extraction from complex video scene. *IEEE Transactions on Image Processing (TIP)*, 18:401–411, 2009.

[10] S. Lucas. Icdar 2005 text locating competition results. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 80–84 Vol. 1, 2005.

[11] S. M. Lucas. Text locating competition - icdar (2003-2005). http://algoval.essex.ac.uk:8080/icdar2005/.

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

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

[14] P. Shivakumara, W. Huang, T. Quy Phan, and C. Lim Tan. Accurate video text detection through classification of low and high contrast images. *Pattern Recognition - Elsevier*, 43:2165–2185, June 2010.

[15] P. Shivakumara, T. Q. Phan, and C. L. Tan. A laplacian approach to multi-oriented text detection in video. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33:412–419, 2011.

[16] K. Turkowski. Graphics gems. Filters for common resampling tasks. pages 147–165. Academic Press Professional, Inc., San Diego, CA, USA, 1990.

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

[18] J. Zhang and R. Kasturi. Text detection using edge gradient and graph spectrum. *International Conference on Pattern Recognition (ICPR)*, 0:3979–3982, 2010.

[19] W. Zhang, G. Zelinsky, and D. Samaras. Real-time accurate object detection using multiple resolutions. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1–8, 2007.