Finding Text in Natural Scenes by Figure-Ground Segmentation

To appear in ICPR 2006

Huiying Shen and James Coughlan
Smith-Kettlewell Eye Research Institute
San Francisco, CA 94115
USA
{hshen,coughlan}@ski.org

Abstract

Much past research on finding text in natural scenes uses bottom-up grouping processes to detect candidate text features as a first processing step. While such grouping procedures are a fast and efficient way of extracting the parts of an image that are most likely to contain text, they still suffer from large amounts of false positives that must be pruned out before they can be read by OCR.

We argue that a natural framework for pruning out false positive text features is figure-ground segmentation, which we implement using a graphical model (i.e. MRF). The graphical model is "data-driven" in that the nodes of the graph correspond to the candidate text features. Since each node has only two possible states (figure and ground), and since the connectivity of the graphical model is sparse, we can perform rapid inference on the graph using belief propagation. We show promising results on a variety of urban and indoor scene images containing signs, demonstrating the feasibility of the approach.

1. Introduction

A large body of work addresses the problem of detecting and reading printed text, but so far this problem is considered solved only in the domain of OCR (optical character recognition). This domain is limited to the analysis of high-resolution, high-contrast images of printed text with little background clutter. The broader challenge of detecting and reading text in highly cluttered scenes, such as indoor or outdoor scenes with informational signs, is much more difficult and is a topic of ongoing research. The most straightforward solution is to search an image for occurrences of every letter in the alphabet, but in most cases this is impractical because of limited image resolution that impairs the visibility of individual letters, unknown font variability, varying lighting conditions and computational time limitations.

While some past work in text detection approaches the problem by searching for individual letters [1], most approaches divide it two separate processes, text segmentation followed by reading the segmented text. We focus on the problem of segmentation in this paper, leaving the task of reading segmented text for future research.

Many text segmentation algorithms employ deterministic, bottom-up processes for grouping text features into candidate text regions using features such as edges, color or texture [12, 7, 5, 4]. A recently developed and very effective algorithm [2] based on a statistical framework uses a cascade of filters trained from a labelled data set of natural scenes containing text. These filters are chosen by Adaboost to maximally distinguish between patches of text and non-text background. Our goal is to combine the best aspects of both types of algorithms by using bottom-up grouping processes in tandem with a statistical model to make corrections to these grouping processes.

Our approach combines a bottom-up search for likely text features, based on grouping sets of simple, rapidly detected features, with a statistical framework that rectifies mistakes made in the bottom-up process. This framework is based on graphical models that are "data-driven" in that their structure and connectivity is determined by the set of candidate text features detected. Such a model represents likely segmentations of the candidate text features into figure (text) and ground (non-text), and provides a way of pruning out false candidates using the context of nearby candidates. Besides providing a natural framework for modeling the role of context in segmentation, another benefit of the graphical model framework is the ability to learn the model parameters automatically from labeled data (though we have not done this in our preliminary experiments).

Recent work related to ours also uses a graphical model

framework for text segmentation in documents [16] and (of greater relevance to our work) in natural scenes [15]. Unlike our approach, the latter work uses color to initiate the segmentation and requires images in which individual letters are clearly visible. By contrast, we have designed our algorithm to process grayscale images with letters that may be poorly resolved (see Figures 6-10). This allows us to segment text in images in which the letters appear small and/or process the images at coarser scales (which decreases the amount of computation required).

2 Grouping Edges into Text Features

We have devised a bottom-up procedure for grouping edges into composite features that are signatures of regions containing text, and which are uncommon in non-text regions. Speed is a major consideration, so we decided to use a very simple edge detector to provide the basic elements to be grouped. Edges are grouped into a hierarchy of more complex features, shown in Figure 1. Edge pairs, horizontal strokes, endstops and corners are the intermediate-level features; the higher-level features, which we call "sticks" and "boxes," are the text features output by the grouping procedure (see Figures 2,3 for examples on a text image). This output is subsequently processed to remove false positive text features using a graphical model, as described in Section 3.

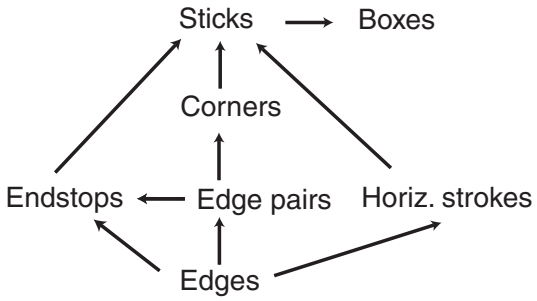

Figure 1. Bottom-up hierarchy of text features. Top-level features (sticks and boxes) are used to construct the graphical model.

Edges are detected (see Figure 2(a)) and classified into four orientations (up, down, left, right) using local minima/maxima of simple x and y derivatives of the image intensity. We assume the text is roughly horizontal and that it consists of light letters against a darker background; text of the opposite polarity is detected simply by reversing the contrast of the image.

Next, nearby horizontal/vertical edges of opposite polarities are grouped into edge pairs (Figure 2(b)). The polarities and range of allowed separation between the edges are chosen to be consistent with the cross-section of a typical character stroke (e.g. going left to right across a vertical stroke, the intensity increases and then decreases a few pixels to the right). This separation range equates to a particular range of scales; we have intentionally kept the separation distances small so that we can detect small letters and/or text in scenes viewed at coarse scales.

Edges are grouped into horizontal strokes (Figure 2(c)) using a greedy procedure that scans the rows of the image. A stroke begins with an edge pixel and continues as long as there is a smooth, roughly horizontal continuation of the edge (i.e. the vertical coordinates of the stroke pixels are allowed to increase or decrease by one pixel from one column to the next). Strokes are categorized as top or bottom depending on the polarity of the edges.

Edges and edge pairs are then grouped into top and bottom endstops (Figure 3(a)), representing ends of strokes. For instance, the bottom of the letter "T" is represented as a bottom endstop by grouping a vertical edge pair with the edge at the bottom of the stroke. The last category of intermediate-level features is corners (Figure 3(b)), which are defined by grouping vertical and horizontal edge pairs that are suitably aligned.

Figure 2. First three stages of feature grouping illustrated on road sign example. Top to bottom: (a) Edges. (b) Edge pairs: horizontal (yellow), vertical (cyan). (c) Horizontal strokes: top (red) and bottom (blue).

The final text candidate features extracted by this process are *sticks* and *boxes* (Figure 3(c)). A stick is a roughly vertical line segment whose height is approximately the height of a text letter. It is defined by two locations, a top and bottom. The top location of the stick is defined by the location of any of the following features: a top left or top right corner, a top endstop or (the center of) a top stroke; the bottom location of the stick is defined similarly in terms of "bottom" features. Any pair of these top and bottom features defines a stick, as long as the distance between the pair is

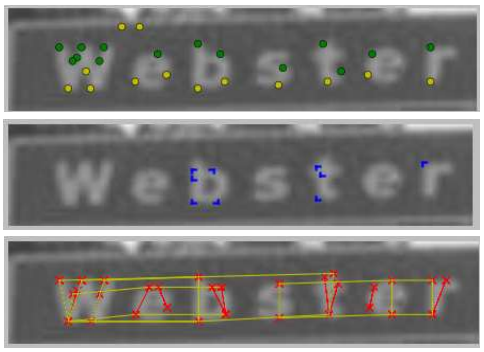

Figure 3. Last three stages of feature grouping illustrated on road sign example. Top to bottom: (a) Endstops: top (green) and bottom (yellow). (b) Corners. (c) Sticks (red) and boxes (yellow).

within a certain range, and the stick is oriented within 45 degrees of the vertical.

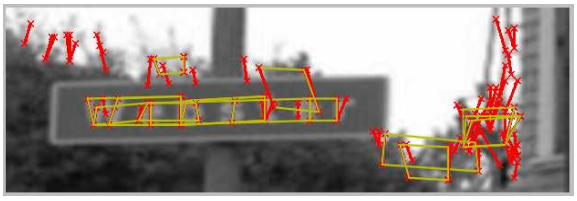

Figure 4. Scene clutter example. Figure shows previous road sign with local scene context. Sticks (red) and boxes (yellow) shown as before. Note numerous false positives in background.

Any two sticks that are sufficiently compatible define a quadrilateral-shaped feature called a box. Here compatibility means that the bottoms of the sticks are within a few rows of each other, and that the sticks have roughly the same height.

The rationale for generating stick and box features is that it is a rapid procedure for extracting structures from an image that reflect the regularity of near-vertical elements occurring in text. In particular, the box features often correlate with the characteristic space occupied by individual letters. Relatively few regions of the non-text background generate many stick and box features; in outdoor urban scenes the most common false-positive regions consist of structures such as tree branches and linear structures such as windows (see Figure 4 for examples).

Not only are the relatively few false-positive sticks and boxes, but these false positives tend to cluster in different ways, and with less regularity, than true positives. In the next section we describe a method for segmenting the true positives based on a graphical model that exploits these differences between true and false positives.

3 Data-Driven Graphical Model for Segmentation

We approach text detection in cluttered scenes as a figure-ground segmentation problem. The output of the grouping process described in the previous section produces a substantial number of false-positive text candidates (stick or box features), but relatively few false negatives. The true positives tend to cluster into regular structures, differently from the false positives, which are distributed more randomly. Thus it is natural to segment the text features into two groups: figure, representing true positives, and ground, for false positives.

Our approach draws on ideas from work on object-specific figure-ground segregation [11], which uses normalized cuts to perform grouping, and from work on clustering using graphical models [9]. We use affinity functions to measure the compatibilities of pairs of elements as potential figure candidates and construct a graphical model to represent a figure-ground process. We call this graphical model *data-driven* because the structure of the model depends on the data: each text feature (stick or box) defines one node of the graph, and the compatibility of any pair of candidates is expressed by the edge connecting the corresponding nodes.

Each node in the graph has two possible states, figure or ground. The graphical model defines a probability distribution on all possible combinations of figure-ground labels at each node. We use belief propagation to estimate the marginal probabilities of these labels at each node; any node with a sufficiently high marginal probability of belonging to the figure is designated as figure.

3.1 Figure-Ground Segmentation

We construct a figure-ground segmentation model using a data-driven graphical model with pairwise interactions representing compatibilities between elements. Our framework is similar to that of the multi-scale graphical models in [8, 6], with the important difference that our graphical model has a data-driven structure, rather than defining a graph with nodes at each location in a pixel lattice. Moreover, rather than the typical nearest-neighbor connectivity on a lattice (each pixel has four or eight neighbors), our graph connects each pair of elements within a certain distance of each other, much like the connectivity used in normalized cuts segmentation [10]. The use of rich text features rather than individual pixels as graph nodes, and the

longer spatial range of connectivity that results, allows our model to perform effective segmentation without the need for a multi-scale model.

We define the graphical model for a general figure-ground segmentation process as follows. Each of the N features extracted from the image is associated with a graph node (vertex) s_i, where i ranges from 1 through N. The data characterizing each feature is denoted by $\vec{d_i}$, which includes any relevant information about the feature.

Each node s_i can be in two possible states, 0 or 1, representing ground and figure, respectively. The probability of any labelling of all the nodes is given by the following expression:

$$P(s_1, \ldots, s_N) = 1/Z \prod_{i=1}^{N} B_i(s_i) \prod_{<ij>} C_{ij}(s_i, s_j)$$

This is the expression for a pairwise MRF, where $B_i(s_i)$ is the unitary potential function, $C_{ij}(s_i, s_j)$ is the binary potential function and Z is a normalization factor. (We note that the probability is conditioned on all data $\vec{d_1}, \ldots \vec{d_N}$, which we omit for brevity. Similarly, the notation for the unary and binary potentials also neglects this dependence on the data.) The notation $< ij >$ represents the set of all pairs of features i and j that are close enough to each other to be directly connected in the graph. $B_i(s_i)$ represents a *unitary* factor reflecting the likelihood of feature s_i belonging to the ground or figure, independent of the context of other nearby features. $C_{ij}(s_i, s_j)$ is the *compatibility function* between features i and j, which reflects how the relationship between two features influences the probability of assigning them to figure-ground.

The choice of unitary and compatibility functions is key to the effectiveness of the graphical model. These functions may be chosen by trial and error, as in the current application, or by maximum likelihood learning. An example of this kind of learning is in [9], in which compatibilities (binary potentials) learned from labeled data are used to construct graphical models for clustering. However, for our preliminary results on text segmentation, simple trial and error sufficed for choosing suitable unitary and compatibility functions.

3.2 Graphical Model for Text Segmentation

In this subsection we discuss the graphical model for text segmentation in more detail. Each stick or box feature gives rise to one node in the graph. Direct connections in this graph are only made between features within a fixed radius that are of *different* types, i.e. between a stick and a box, but not between two sticks or between two boxes.

The unitary potentials are defined to reward sticks and boxes for figure membership according to how closely they conform to their ideal form. For a stick s_i, the unitary potential is $B(s_i) = F_U^{stick}$ if $s_i = 1$ and $B(s_i) = 1$ if $s_i = 0$. Here

$$F_U^{stick} = e^{-|x_T - x_B|/|y_T - y_B|} 2^{N_c}$$

where (x_T, y_T) and (x_B, y_B) are the top and bottom coordinates of the stick, and N_c is the number of corners that define it (i.e. 0, 1 or 2). The first factor rewards verticality of the stick, and the second rewards the presence of corners, which provide evidence for the stick belonging to a text region.

For a box s_i, the unitary potential is $B(s_i) = F_U^{box}$ if $s_i = 1$ and $B(s_i) = 1$ if $s_i = 0$. Here

$$F_U^{box} = e^{-|h_1 - h_2|/|h_{avg}|} 2^{N_c}$$

where h_1 and h_2 are the heights of the sticks that comprise the box, and $h_{avg} = (h_1 + h_2)/2$ is the average of the stick heights. The first factor rewards the similarity of the stick heights, and the second rewards the presence of corners as before.

The binary potential is defined between a stick and a box, and rewards the consistency of the two features that is evidence for them belonging to the same text region. For a stick s_i and box s_j, the binary potential is $C(s_i, s_j) = F_B$ if $s_i = 1$ and $s_j = 1$, and $C(s_i, s_j) = 1$ otherwise. Here

$$F_B = e^{-E} R$$

where E is an error measure that is the sum of three terms:

$$E = K_1 |h_1 - h_{avg}^{Box}| + K_2 \min_{i=2,3} |x_1 - x_i| + K_3 [|m_1 - m_3| + |m_2 - m_3|]$$

Here h_1 is the height of the stick and h_{avg}^{Box} is the average height of the box (i.e. the average height of the two sticks in the box); x_i represents the values of the x locations of the three stick bottoms (so x_1 is for the lone stick, x_2 is for the left stick of the box and x_3 is for the right stick of the box); and $m_1 = (y_1 - y_2)/(x_1 - x_2)$, $m_2 = (y_2 - y_3)/(x_2 - x_3)$, $m_3 = (y_1 - y_3)/(x_1 - x_3)$ represent slopes (to enforce approximate horizontality of the stick bottoms). K_1, K_2 and K_3 are constants.

Finally, the factor R is a constant that rewards the appropriate ratio of horizontal and vertical edge pairs in the region defined by the lone stick and the more distant stick of the box. Let N_h and N_v be the number of horizontal and vertical edge pairs, respectively, in this region. Then $R = 5$ if $N_v/(N_v + N_h) > T_R$ for an appropriate threshold T_R (in text regions the majority of edge pairs are usually vertical), and $R = 1$ otherwise.

Once the graph has been defined in this way, we run belief propagation [13] in an asynchronous schedule that

sweeps the entire graph a few times. Then the unitary beliefs, i.e. estimates of $P(s_i = 1)$, are used to decide if each feature belongs to the figure or ground. If $P(s_i = 1) > 0.9$ then we decide "figure". The results of this figure-ground labeling procedure are shown in Figure 5.

Figure 5. Eliminating background states using belief propagation. Top: stick and box features. Bottom: remaining figure states after belief propagation. Note that the false positives are eliminated.

4 Results

We ran our algorithm with the same exact settings and parameter values for all of the following images (see Figures 6-10). The algorithm (written in interpreted Python code) took about half a minute to process 640 x 500 images on a standard laptop computer. Note the algorithm's ability to handle considerable amounts of scene clutter. In Figure 7 we demonstrate the algorithm's ability to detect text of two different polarities. Also note that the algorithm handles a limited range of scales; in order to detect other text in these figures we would need to run the algorithm at different scale ranges. The algorithm's robustness to non-uniform lighting conditions is shown in Figure 8. Figure 9 demonstrates that the algorithm is selective for finding the Latin (e.g. English) alphabet because the edge grouping procedure was designed specifically for English text. Finally, in Figure 10 notice that the algorithm finds two false positives in a building region, which contains periodic structures resembling text at a local scale.

Figure 6. Experimental results. Note algorithm's ability to handle large amounts of scene clutter.

5 Conclusions

We have demonstrated the feasibility of a novel algorithm for segmenting text in natural scenes. The algorithm consists of a grouping procedure to generate text feature hypotheses, and a graphical model for segmenting these hypotheses into figure and ground. The results show that the algorithm is capable of segmenting text in a variety of scenes with few false positives.

Future work will focus on converting the code to C++ (which we expect to speed up the algorithm by an order of magnitude) using manually segmented images of text to learn the appropriate graphical model potentials, and on accurately framing the detected text into a box-shaped region which can be read by OCR. In many cases it will be important to undo distortion from perspective viewing [3] in order to better segment, frame and read the text. Finally, we note that the OCR stage will have the added benefit of ruling out some false positives which cannot be ruled out earlier.

6 Acknowledgments

The authors were supported by the National Institute on Disability and Rehabilitation Research (grant no.

Figure 7. Polarity selectivity. Top: algorithm finds light text on dark background. Bottom: by inverting the contrast of input image, text of opposite polarity is detected.

Figure 8. Robustness to non-uniform lighting, in this case due to shadows falling on sign.

H133G030080), the NSF (grant no. IIS0415310) and the National Eye Institute (grant no. EY015187- 01A2).

References

[1] S. Gold, A. Rangarajan, C.P. Lu, S. Pappu and E. Mjolsness, New Algorithms for 2D and 3D Point Matching: Pose Estimation and Correspondence, Pattern Recognition, 31(8):1019-1031, 1998.

[2] X. Chen and A. L. Yuille. “Detecting and Reading Text in Natural Scenes.” CVPR 2004.

[3] X. Chen, J. Yang, J. Zhang and A. Waibel. “Automatic Detection of Signs with Affine Transformation.” Proceedings of WACV2002. Orlando, December, 2002.

[4] J. Gao and J. Yang. “An Adaptive Algorithm for Text Detection from Natural Scenes.” CVPR 2001.

Figure 9. Algorithm is tailored to finding English text, and does not detect Chinese text.

Figure 10. Most common false positives: periodic structures such as rows of windows resemble text at a local scale.

[5] H. Li, D. Doermann and O. Kia. Automatic text detection and tracking in digital videos. IEEE Transactions on Image Processing, 9(1):147–156, January 2000.

[6] X. He, R. S. Zemel and M. A. Carreira-Perpinan. “Multiscale Conditional Random Fields for Image Labeling.” CVPR 2004.

[7] A.K. Jain and B. Tu. “Automatic Text Localization in Images and Video Frames.” Pattern Recognition. 31(12), pp 2055-2076. 1998.

[8] S. Kumar and M. Hebert. “Man-Made Structure Detection in Natural Images using a Causal Multiscale Random Field.” CVPR 2003.

[9] N. Shental, A. Zomet, T. Hertz and Y. Weiss. “ Pairwise Clustering and Graphical Models.” NIPS 2003.

[10] J. Shi and J. Malik. ”Normalized Cuts and Image Segmentation.” IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(8), 888-905, August 2000.

[11] S. X. Yu and J. Shi. “Object-Specific Figure-Ground Segregation.” CVPR 2003.

[12] V. Wu, R. Manmatha, and E. M. Riseman. Finding Text In Images. Proc. of the 2nd intl. conf. on Digital Libraries. Philadaphia, PA, pages 1–10, July 1997.

[13] J.S. Yedidia, W.T. Freeman, Y. Weiss. “Bethe Free Energies, Kikuchi Approximations, and Belief Propagation Algorithms”.2001. MERL Cambridge Research Technical Report TR 2001-16.

[14] A. L. Yuille, “Deformable Templates for Face Recognition”. *Journal of Cognitive Neuroscience.* Vol 3, Number 1. 1991.

[15] D.Q. Zhang and S.F. Chang, “Learning to Detect Scene Text Using a Higher-Order MRF with Belief Propagation.” CVPR 04.

[16] Y. Zheng, H. Li and D. Doermann, “Text Identification in Noisy Document Images Using Markov Random Field.” Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003).