Figure-Ground Segmentation Using Factor Graphs

Huiying Shen, James Coughlan and Volodymyr Ivanchenko

Smith-Kettlewell Eye Research Institute
San Francisco, CA 94115
USA

Abstract

Foreground-background segmentation has recently been applied [26,12] to the detection and segmentation of specific objects or structures of interest from the background as an efficient alternative to techniques such as deformable templates [27]. We introduce a graphical model (i.e. Markov random field)-based formulation of structure-specific figure-ground segmentation based on simple geometric features extracted from an image, such as local configurations of linear features, that are characteristic of the desired figure structure. Our formulation is novel in that it is based on factor graphs, which are graphical models that encode interactions among arbitrary numbers of random variables. The ability of factor graphs to express interactions higher than pairwise order (the highest order encountered in most graphical models used in computer vision) is useful for modeling a variety of pattern recognition problems. In particular, we show how this property makes factor graphs a natural framework for performing grouping and segmentation, and demonstrate that the factor graph framework emerges naturally from a simple maximum entropy model of figure-ground segmentation.

We cast our approach in a learning framework, in which the contributions of multiple grouping cues are learned from training data, and apply our framework to the problem of finding printed text in natural scenes. Experimental results are described, including a performance analysis that demonstrates the feasibility of the approach.

Key words: figure-ground segmentation, belief propagation, factor graphs, text detection
PACS:

1 Introduction

Originally proposed as a generic process for segmenting the foreground of a scene from the background, figure-ground (foreground-background) segmentation has recently been successfully applied [26,12] to the detection and segmentation of specific objects or structures (i.e. targets) of interest from the background. Standard techniques such as deformable templates [27] are poorly suited to finding some targets, such as printed text, stripe patterns, vegetation or buildings, particularly when the targets are regular (e.g. quasi-periodic) or texture-like structures with widely varying extent, shape and scale.

In these cases it seems more appropriate to group target features into a common foreground class (at least as an initial segmentation step to precede further processing), rather than directly attempt to find a detailed correspondence between a prototype and the target in the image, as is typically done with deformable template and shape matching techniques.

Our graphical model-based approach to figure-ground segmentation is an outgrowth of earlier work on figure-ground segmentation applied to finding crosswalks in traffic intersections [5] and builds on more recent work in [19] and [9]. The approach emphasizes the use of the *geometric* relationships of features extracted from an image as a means of grouping the target features into the foreground. In contrast with related MRF techniques [28] for classifying individual image patches into small numbers of categories, our approach seeks to make maximal use of geometric features extracted from images, rather than raw pixel information. Geometric information is generally more intuitive to understand than filter-based feature information, and it may also be more appropriate when lighting conditions are highly variable.

We formulate our approach in the general case of figure-ground sementation and apply it to the problem of finding printed text in natural scenes. Experimental results are described, including a performance analysis that demonstrates the feasibility of the approach.

2 Grouping with Factors

In this paper we describe the use of factor graphs as a natural framework for grouping (i.e. segmenting) features in an image. Any grouping process analyzes relationships among features in deciding how to group them; these relationships are interactions among features that reflect their compatibility for inclusion into the same group. In much past work on grouping and segmentation, such as normalized cuts [22,26] and graphical-model based typical cuts

[20] (all of which inspired our approach), these interactions are pairwise measures that measure the similarity (or affinity) of two features. Our approach is similar to that of [20] in that it constructs a Markov random field with one binary-valued (0 or 1) node variable for each feature to be grouped, and uses belief propagation to decide how to group the features (two features are assigned to the same group if the probability that they are jointly assigned to the same binary labels is above a threshold); however, their approach imposes a fundamental symmetry between the two possible states of each node variable, whereas in our approach the two possible states represent figure (state 1) or ground (state 0), which are not symmetric. The object-specific figure-ground technique in [26] imposes the same type of figure-ground asymmetry as our work does, but their work was applied to specific targets such as telephones and mugs and it is unclear how this technique would generalize to texture-like patterns with highly variable numbers of elements.

The above techniques all use *pairwise* measures that measure the similarity (or affinity) of two features. However, many clustering problems necessitate the use of higher-order interactions; for instance, the problem of grouping points on a 2-D plane into lines requires an interaction defined on triplets of points, since every pair of points is trivially collinear. Some recent work [1] has investigated hypergraph partitioning techniques for handling these higher-order interactions.

Factor graphs [11] are graphical models designed to express interactions of any order (generalizing the pairwise interactions often used in graphical models, i.e. Markov random fields), and may be used for formulating simple and efficient grouping algorithms. We apply this formulation to the problem of object-specific figure-ground segmentation, which is how we cast the problem of text detection.

In the next section we introduce factor graphs, and in subsequent sections we describe how a particular functional form of factor graph appropriate for figure-ground segmentation is suggested by a maximum entropy model of grouping.

2.1 Factor Graphs

Factor graphs [11] are representations of probability distributions of many variables that can be expressed as a product of several factors, where each factor is an explicit interaction among a subset of these variables. (For a comprehensive treatment of factor graphs, see Chapter 8 of [2], which can be downloaded at http://research.microsoft.com/~cmbishop/PRML/ .) Fig. 1 shows an example of a factor graph depicted in a graphical format. Each

square node represents a factor, or interaction, among one or more variables, depicted by circles, and the topology of the factor graph indicates how the joint distribution of all variables factors.

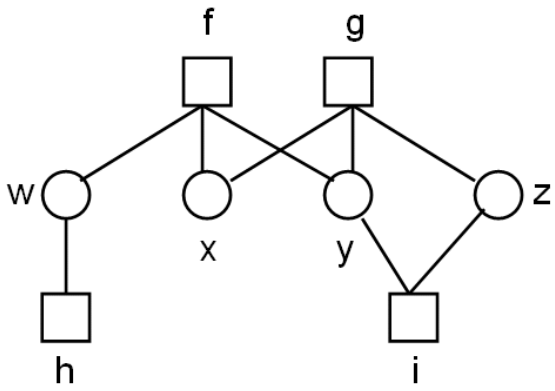

Fig. 1. Factor graph. This graph represents a distribution on four variables w,x,y,z (drawn as circles) using four factors f, g, h, i (drawn as squares). Edges connect factors with the variables they influence. The joint distribution represented by this factor graph is $P(w, x, y, z) = f(w, x, y)g(x, y, z)h(w)i(y, z)$.

The *arity* of a factor is the number of variables that interact in the factor. An arity-1 factor is called a *unitary* factor, and an arity-2 factor is sometimes referred to as a *pairwise interaction*. For the example shown in Fig. 1, factor h is arity-1, factor i is arity-2 and factors f and g are arity-3. In our figure-ground segmentation application, one factor arises from each constraint (measurement), and its arity equals the number of features (nodes) included in the measurement constraint.

The factor graph framework is an extremely general framework that is convenient for representing a large variety of probabilistic models. As we will see in a later section, inference on factor graphs is made tractable by approximate techniques such as belief propagation, which we use in our text-finding algorithm.

2.2 Theoretical Motivation: Maximum Entropy

In this section we discuss the theoretical motivation for the functional form of the model we use for figure-ground segmentation, which is inspired from a simple maximum entropy probability model. As we will see, this functional form is naturally described using a factor graph.

The motivation for our maximum entropy model is that, given a collection of features to be segmented into figure or ground, evidence for how to assign (segment) features to figure or ground arises from considering *relationships*

among two or more features. (Depending on the nature of the segmentation task, there may also be evidence for *individual* features belonging to figure or ground that is independent of other features.) The goal is to combine noisy evidence pertaining to groups of features in such a way that we can decide the likelihood that any *individual* feature should be assigned to figure or ground. Our approach is to construct a joint probability distribution of the assignments of all the features that is consistent with the evidence from groups of features. The joint distribution is chosen to be the least biased distribution that is consistent with the evidence; we enforce this "minimal-bias" criterion using a maximum entropy approach [17].

We now present the details of our maximum entropy model of figure-ground segmentation. We are given a collection of n features (i.e. nodes) in an image, each of which has an unknown assignment to figure or ground, denoted by x_i for $i = 1, 2, \ldots, n$, where $x_i = 0$ represents assignment to the ground state and $x_i = 1$ represents assignment to the figure state. (Image data is also associated with each feature, such as its pixel coordinates.) Certain subsets of nodes are selected as *candidates to be grouped into the figure*, and we assume that for each subset (with at least two nodes) there is a way to estimate the probability that all nodes in the subset belong to the figure. Such a probability measure is based on the image data associated with the features; for instance, in the collinearity example in Section 2.4, the probability that a subset of three feature points belongs to the figure would be a monotonically increasing function of the degree of collinearity and proximity of the feature points.

We can regard the probability measure associated with each feature subset as a measurement that constrains the joint distribution of all nodes given the measurement data, which yields the posterior distribution $P(x_1, x_2, \ldots, x_n|data)$. (In this section, conditioning on "data" means that we are conditioning on some or all such measurement data; in later sections we will use more precise notation to refer to individual pieces of measurement data.) We can then show that the form of $P(x_1, x_2, \ldots, x_n|data)$ having the maximum entropy distribution consistent with these constraints will be a product of multiple factors, one factor corresponding to each constraint.

We demonstrate this fact with an illustrative example (see Fig. 2); it is straightforward to extend the analysis to an arbitrary collection of features and feature subsets. In this example there are four nodes, x_1, x_2, x_3, x_4, and two subset constraints: $P(x_1 = 1, x_2 = 1, x_3 = 1|data) = p_1$ and $P(x_3 = 1, x_4 = 1|data) = p_2$. The joint distribution we are seeking, $P(x_1, x_2, x_3, x_4|data)$, has entropy equal to $-\sum_X P(X|data) \log P(X|data)$, where $X = (x_1, x_2, x_3, x_4)$. The problem is a constrained optimization problem, and the solution can be obtained using Lagrange multipliers:

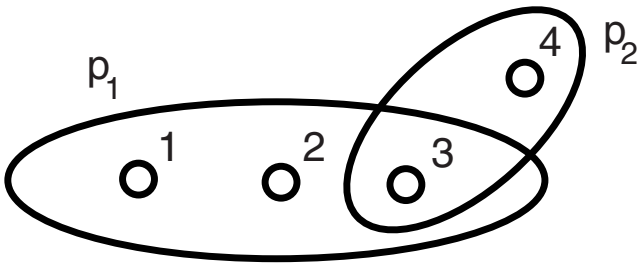

Fig. 2. Example of probability measure applied to subsets of features (i.e. candidate groupings). Each feature (node) is depicted by a circle, labeled 1 through 4; the two ovals depict candidate groupings of features. Values p_1 and p_2 denote the probability that all the features within each oval belong to the figure.

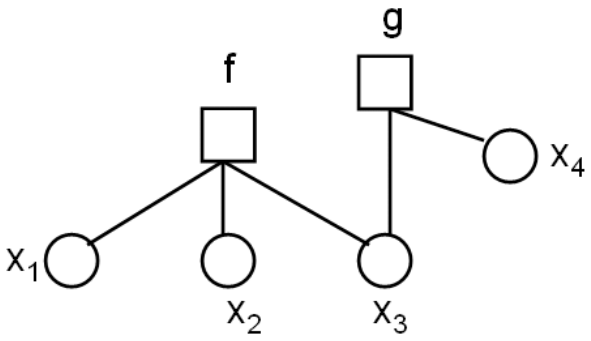

Fig. 3. Recasting grouping example from Fig. 2 as a factor graph. The four features have unknown states, denoted x_1,x_2,x_3 and x_4, each of which equals 0 or 1, representing figure or ground, respectively. Factor $f(x_1, x_2, x_3)$ results from a measurement applied to features 1, 2 and 3; factor $g(x_3, x_4)$ results from a measurement applied to features 3 and 4.

$$
\begin{aligned}
E = & -\sum_X P(X|data) \log P(X|data) \\
& +\lambda_1 (\sum_{x_4} P(x_1 = 1, x_2 = 1, x_3 = 1, x_4|data) - p_1) \\
& +\lambda_2 (\sum_{x_1, x_2} P(x_1, x_2, x_3 = 1, x_4 = 1|data) - p_2) \\
& +\tau (\sum_X P(X|data) - 1)
\end{aligned} \tag{1}
$$

The Lagrange multipliers λ_1 and λ_2 enforce the two subset constraints, and τ enforces the fact that $P(X|data)$ is a normalized distribution. The expression for $\partial E / \partial P(X|data)$ is:

$$
\partial E / \partial P(X|data) = -\log P(X|data) - 1 + \lambda_1 x_1 x_2 x_3 + \lambda_2 x_3 x_4 + \tau \tag{2}
$$

where we notice that $x_1 x_2 x_3 = 1$ only when $x_1 = 1$, $x_2 = 1$ and $x_3 = 1$. If we

set $\partial E/\partial P(X|data) = 0$, then we get the following solution:

$$P(X|data) = \frac{1}{Z} e^{\mu_1 x_1 x_2 x_3} e^{\mu_2 x_3 x_4} \tag{3}$$

The key point to notice is that $P(X|data)$ contains one factor (term) for each constraint, which in this case we label as $f(x_1, x_2, x_3) = e^{\mu_1 x_1 x_2 x_3}$ and $g(x_3, x_4) = e^{\mu_2 x_3 x_4}$. The values μ_1 and μ_2 are parameters that must be chosen so that $P(X|data)$ satisfies the two subset constraints (for details on how to estimate these parameters, see [20]); the value Z is a constant chosen to make $P(X|data)$ properly normalized. The other important point is that the factors have a special form that appears throughout this paper: *each factor has one value if all of its arguments are in the figure state and another value if one or more the arguments are in the ground state.*

This model is naturally formulated as a factor graph, since the joint posterior probability $P(X|data)$ is expressible as a product of the factors $f(.)$ and $g(.)$ (neglecting the normalization constant Z that is independent of X). In the next section we will develop a simpler version of the model, whose parameters can be easily learned from training data. Factor BP will then be used to determine how to label individual features (figure or ground), which we describe in Section 3.

2.3 Figure-Ground Segmentation Using Factor Graphs

In the previous subsection, all measurements were marginal probabilities that certain features (nodes) all belonged to figure. However, it is rare that we are given measurements that give direct access to such marginals – instead, it is much more likely that we have various *cues* that indirectly reflect such marginal probabilities. For instance, if we want to find those features in an image that group into a (roughly) straight line, an important cue will be the degree of collinearity among triplets of features. Thus, we will use the same functional form of the maximum entropy model in Sec. 2.2, given by Eq. 3, to construct a simpler model that uses arbitrary cues rather than direct measurements of marginal probabilities.

We now express this simpler model for the concrete example given in Sec. 2.2. Two cues are measured in this example, C_{123} and C_{34}, which are scalar quantities that reflect the likelihood that the corresponding node subsets $\{1, 2, 3\}$ and $\{3, 4\}$ should be grouped into the figure. (Note that these are arbitrary cues, *not* direct measurements of marginal probabilities denoted by p_1 and p_2 in the previous section.) Then, if we assume conditional independence of the

two measured cues, we have the following expression for the joint likelihood:

$$P(data|X) = P(C_{123}|X)P(C_{34}|X) = P(C_{123}|x_1, x_2, x_3)P(C_{34}|x_3, x_4) \quad (4)$$

Following the form of Eq. 3, we will impose the requirement that the individual likelihood functions $P(C_{123}|x_1, x_2, x_3)$ and $P(C_{34}|x_3, x_4)$ will each depend only on *whether the states they are conditioned on are all figure or not.* In other words, $P(C_{123}|x_1, x_2, x_3) = P_{on}(C_{123})$ if $x_1x_2x_3 = 1$ and $P(C_{123}|x_1, x_2, x_3) = P_{off}(C_{123})$ if $x_1x_2x_3 = 0$.

Having defined the likelihood functions $P_{on}(.)$ and $P_{off}(.)$ for each cue, we *learn* them from training data, rather than choosing arbitrary expressions and parameters for them. (In our application on finding text, we use histograms learned from training data to represent these distributions, but other ways of learning and representing the distributions may be more appropriate for other applications.)

The model is used to perform inference by estimating the MAP (maximum a posterior), where the posterior is given by Bayes rule:

$$P(X|data) = P(X)P(data|X)/P(data) \quad (5)$$

Note that maximizing the posterior with respect to X is equivalent to maximizing $P(X)P(data|X)$, since $P(data)$ is independent of X.

As before, $X = (x_1, x_2, \ldots, x_N)$, where N is the number of nodes (features) in the image to be grouped. Assume that there is a cue for each triple (i, j, k) of neighboring features, C_{ijk}, where $i < j < k$. (It is straightforward to generalize this model to multiple cues with arbitrary arities; here we consider just one cue of arity three.) Then the expression that needs to be maximized to estimate the MAP is:

$$P(X)P(data|X) = P(X) \prod_{(ijk)} P(C_{ijk}|x_i, x_j, x_k) \quad (6)$$

where $\prod_{(ijk)}$ denotes a product over all feature triples such that $i < j < k$.

Note that $\prod_{(ijk)} P(C_{ijk}|x_i, x_j, x_k)$ can be re-expressed as the following:

$$[\prod_{(ijk)} P_{off}(C_{ijk})][\prod_{(ijk):x_ix_jx_k=1} P_{on}(C_{ijk})/P_{off}(C_{ijk})] \quad (7)$$

where the restriction $x_ix_jx_k = 1$ in the product ensures that only those triples whose nodes all belong to figure are included. Since the term $\prod_{(ijk)} P_{off}(C_{ijk})$

is independent of X, the MAP can be determined by maximizing the following expression:

$$R(X) = P(X) \prod_{(ijk):x_i x_j x_k = 1} P_{on}(C_{ijk}) / P_{off}(C_{ijk}) \tag{8}$$

$R(X)$ is proportional to the posterior distribution (i.e. it is unnormalized, and the constant of proportionality is the normalization factor, which depends on the data). Assuming a simple i.i.d. prior, such as $P(X) = \prod_{i=1}^{N} P_i(x_i)$, where $P_i(x_i)$ equals α if $x_i = 0$ and $1 - \alpha$ if $x_i = 1$ (e.g. enforcing a preference that each node belong to ground rather than figure if $\alpha > 1/2$), $R(X)$ can be represented by a factor graph with factors of arity one and arity three (for the prior and likelihood, respectively).

Taking logarithms, an equivalent way of estimating the MAP is to maximize the following function:

$$\log R(X) = \sum_i \log P_i(x_i) + \sum_{(ijk)} x_i x_j x_k \log[P_{on}(C_{ijk}) / P_{off}(C_{ijk})] \tag{9}$$

where the product $x_i x_j x_k$ ensures that the sum is taken only over those triples whose nodes all belong to figure. We will use factor graph BP to estimate the MAP of this function, as described in Sec. 3.

2.4 Example of Figure-Ground Segmentation

As a simple example of a factor graph for figure-ground segmentation, consider the problem of segmenting a collection of points in an image (Fig. 4(a)), where the figure consists of points that lie on smooth, nearly straight lines, and the ground consists of all other points. We will define factors that express the degree of collinearity among points; since any two points define a line, we will construct arity-3 factors with three points, which are the smallest factors for which collinearity is a meaningful concept. In this example, given point features i,j and k in the image, we define the collinearity C_{ijk} of the points as follows:

$$C_{ijk} = \left| \frac{(\vec{p}_1 - \vec{p}_2) \cdot (\vec{p}_2 - \vec{p}_3)}{||\vec{p}_1 - \vec{p}_2||\,||\vec{p}_2 - \vec{p}_3||} \right| \tag{10}$$

where $\vec{p}_i$ denotes the (x, y) coordinates of point feature i in the image. Note that C_{ijk} equals 1 when the three points are exactly collinear and 0 when they define a right triangle; points that are approximately collinear will have values less than, but close to, 1.

We measure the distribution of C_{ijk} from a database of training examples, where $P_{on}(C_{ijk})$ and $P_{off}(C_{ijk})$ are the conditional distributions for triples of points that are on and off (respectively) smooth lines. Specifically, we manually label those features in our database that we deem as figure, and all other features are assumed to belong to ground. Then we search for all triples of points that are sufficiently close to one another (there is no need to include triples whose points span half of the image, for example). To cut down on the potentially huge number of triples that can be composed in images with hundreds or more feature points, we may further exclude (i.e. prune) those triples for which C_{ijk} is too close to 0. Of the triples thus formed, any triple composed of points all labeled as figure is used in the calculation of a histogram representing $P_{on}(C_{ijk})$, and all others are used for the histogram representing $P_{off}(C_{ijk})$.

Finally, we estimate an i.i.d. prior on X by empirically counting what proportion of points in the training database belong to figure (see the discussion of the prior just before Eq. 9).

Having learned P_{on} and P_{off}, we can now infer the most likely segmentation of point features in a new image by finding the value of X that maximizes $\log R(X)$ in Eq. 9. As long as the same pruning procedure is used in learning and in inference, we expect that the pruning of factors discussed above is unlikely to change the maximizer of $\log R(X)$ by a significant amount. (The estimate of P_{off} will automatically reflect the population of triples that survives pruning; conversely, very few triples belonging to figure will be pruned, which means that pruning will have a much smaller effect on the estimation of P_{on}.) Indeed, such a pruning procedure is essential for making learning and inference fast enough to be tractable.

Fig. 4 shows a schematic example of a figure-ground segmentation performed using a simple model of this form. The maximizer of $\log R(X)$ was estimated using factor BP (belief propagation), which is discussed in Sec. 3. Fig. 4(a) shows the point features input to the algorithm and Fig. 4(b) shows the output, with figure points drawn as red circles (all the rest are classified as ground).

3 Belief Propagation for Factor Graphs

Belief propagation (BP) is a standard procedure [25,2] for performing inference on graphical models such as MRFs and factor graphs. A graphical model specifies the joint distribution of all the variables in the model; a fundamental challenge in graphical models is to find the most likely values of all the variables (e.g. the MAP estimate). A naive way of accomplishing this is to exhaustively search over all combinations of variables, and to find the combination which

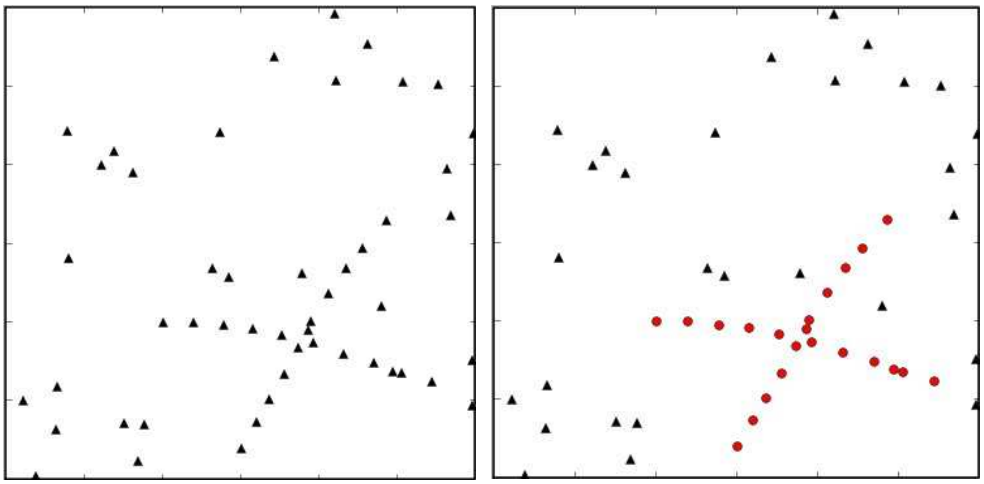

Fig. 4. Simple factor graph example. (a) Collection of points in an image to be segmented. (b) Points segmented as figure are shown as red circles.

is assigned the highest global probability by the graphical model. Such an approach is clearly intractable because of the exponentially large number of such combinations that must be considered. BP is a fast iterative procedure for estimating the marginal properties of individual variables, which may be used to estimate the most likely combination of all variables. Specifically, BP estimates either the marginal probabilities of each variable or some sort of score indicating which states are most likely for each variable.

The main idea behind BP is that different factors and variables "talk" to each other at each iteration, passing "messages" to their neighbors with their estimates of the neighbors' likely states. After enough iterations, this series of "conversations" is likely to converge to a consensus, at which time the marginal estimates are fully determined. While BP provides only approximate solutions in general, it converges in a finite number of iterations to the exact solution when the graphical model has no loops (i.e. cycles), which is the case for graphical models representing Markov chains or any graphical models whose connectivity is tree-shaped. Moreover, empirical studies have established [15] that in practice, BP often converges to good (if somewhat sub-optimal) solutions even in the presence of loops.

While most work in computer vision using BP has been done for pairwise MRFs [23,4,21,7], BP is readily extended to factor graphs [11]. Here we present a brief overview of factor graph BP, using notation similar to that of [11]. (A good introduction to factor graph BP is contained in Chapter 8 of [2], which can be downloaded at http://research.microsoft.com/~cmbishop/PRML/) We describe the "max-product" version of factor BP, which provides scores indicating which states are most likely for each variable, rather than the "sum-product" version, which estimates marginal probabilities for each variable. Not only does max-product BP provide a direct estimate of the MAP, but it is also

computationally more efficient.

In factor BP, messages are sent from factors to variables, indicating (at any iteration) the current estimate of the likelihood of each state for that variable according to the factor. (In standard treatments of BP [11] there are also messages sent from variables to factors, but since these are in turn expressed in terms of the opposite messages – from factors to variables – we will substitute these expressions anywhere the messages from variables to factors appear.) The fundamental operation of BP is the *message update equation*, which defines how to update messages at each iteration, and which one iterates enough times until the messages converge (though there is no guarantee of convergence in loopy graphs). Then the messages are used to calculate the *beliefs*, which provide scores indicating which states are most likely for each variable.

We express the max-product factor BP algorithm in the log domain, where it is understood that factor functions here are the log of the factors defining the joint probability of the factor graph, as in Sec. 2.3. In this domain, the message update equation is as follows:

$$m_{f\to x}(x) \leftarrow \max_{\sim\{x\}}\Big(f(X) + \sum_{y\in n(f)\setminus\{x\}} \sum_{h\in n(y)\setminus\{f\}} m_{h\to y}(y)\Big) \tag{11}$$

where X is the set of arguments of function f, and $\sim\{x\}$ denotes the set of all arguments of f except for x, which we term the set of "siblings" of x under factor f (see Fig. 5(a)). Also, $n(f)$ denotes all neighboring variables of factor f (i.e. all variables directly influenced by f), and $n(y)$ denotes all neighboring factors of variable y (i.e. all factors that directly influence y).

The message update equation is illustrated in Fig. 5(b). The first sum in the update equation is over all siblings of variable x. For each sibling y, the second sum includes all messages from factors *other than* f that flow into the sibling.

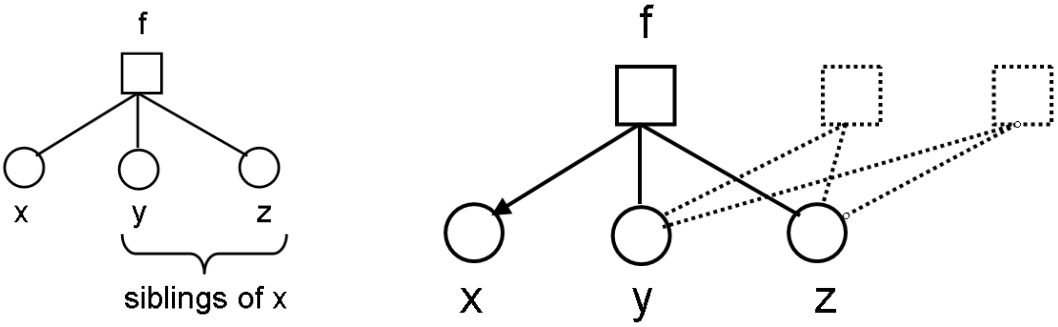

Fig. 5. (a) The siblings of a node variable x under factor f are the other variables included in the factor, in this case y and z. (b) The message update $m_{f\to x}(x)$, indicated by the solid line with the arrow, is calculated by summing all messages from other factors flowing to the siblings of x, shown as dashed lines.

Note that, for each factor f and neighboring variable x, updating all messages $m_{f\to x}(x)$ has worst-case complexity $O(|S|^M)$, where M is the number of variables coupled by factor f and $|S|$ is the number of allowed states of each of the M variables (assuming the state spaces are the same size for each variable). This is because Eq. 11 must be iterated for each value of x on the left-hand side, and for each value of x the max must be evaluated over the remaining $M-1$ sibling variables $\sim\{x\}$.

The order in which messages are updated defines the *message update schedule.* In an asychronous schedule (which is commonly used on serial computers), one message (i.e. specified by one factor and one variable) is updated at a time according to Eq. 11 at each iteration. An entire message "sweep" refers to a sequence of message updates, such that every possible message in the graph is updated once. Generally speaking, message convergence requires several sweeps, depending on the connectivity of the factor graph and the precise nature of the probabilities encoded by it.

Because of a simplification that arises from our figure-ground application, we use instead a fully synchronous schedule, in which all messages are updated in parallel (see Sec. 3.1 for details).

Once the messages have converged to some value (which, in general, we can only hope happens after enough message updates), we can calculate the belief function for each node:

$$b(x) = \sum_{f\in n(x)} m_{f\to x}(x) \tag{12}$$

In max-product BP, the belief is a function with the following property: the state that maximizes the belief of a node is an estimate of the node's state in the most likely global configuration of states across the entire graphical model (i.e. the MAP estimate if the graphical model is interpreted as representing a posterior distribution).

3.1 Special Case: Factor BP for Figure-Ground

In this section we consider the behavior of factor BP for the case of a factor graph we have devised for performing figure-ground segmentation. Since we are working in the log domain, our factor graph represents the log posterior (unnormalized) given in Eq. 9. We re-express this equation as follows:

$$\log R(X) = G\sum_i x_i + \sum_{(ijk)} x_i x_j x_k L(C_{ijk}) \tag{13}$$

where we define $L(C_{ijk}) = \log[P_{on}(C_{ijk})/P_{off}(C_{ijk})]$. As a result, the factor corresponding to triple (i, j, k) is $f(x_i, x_j, x_k) = x_i x_j x_k L(C_{ijk})$. As before, we assume that the prior is i.i.d. (i.e. each feature has a prior bias for ground, independent of other features), where $G < 0$ is a constant less than 0 expressing a bias in favor of ground. (The prior need not be normalized here, since the expression is for the log of the unnormalized posterior.)

Let us assume that we initialize all messages to 0, as is standard in max-product BP. We now analyze a synchronous (parallel) message update schedule applied to this initial condition. It is straightforward in this case to determine the message values after one sweep (i.e. each message has been updated exactly once). For all messages flowing from arity-1 factors (e.g. corresponding to the prior), the expression is:

$$m^{(1)}_{f\to x}(x = 1) = G \tag{14}$$

$$m^{(1)}_{f\to x}(x = 0) = 0 \tag{15}$$

since the variables receiving messages from these factors have no siblings. Here the superscript (1) indicates the value of the messages after one sweep.

All other messages (i.e. greater than arity-1) have the following form after one sweep:

$$m^{(1)}_{f\to x}(x = 1) = \max(0, L_f) \tag{16}$$

$$m^{(1)}_{f\to x}(x = 0) = 0 \tag{17}$$

where L_f denotes the value of $L(C_{ijk})$ when factor f is specified by the triple (i, j, k). Notice that no message values appear on the right-hand side of these equations: all messages on the left-hand side are being updated in parallel, and any messages referred to on the right-hand side of Eq. 11 have been set to 0.

If we use the messages obtained after just one sweep, the beliefs have a simple expression. Since only the difference between beliefs for the figure and ground states matters, we define $B_x = b_x(x = 1) - b_x(x = 0)$. The expression for B_x is then:

$$B_x = \sum_{f\in n(x)} \max(0, L_f) + G \tag{18}$$

A value of $B_x > 0$ means that the estimated optimal state of x is 1 (figure), and $B_x \leq 0$ indicates ground.

Finally, we note that in a previous version of this work [19], we constructed a similar factor graph for which $L_f > 0$ for all factors, and no prior was used. In this case we showed that one sweep of factor BP guaranteed convergence. The difference here is that L_f and the prior are learned from training data, and so the resulting values of L_f can be positive or negative. As a result, convergence is no longer guaranteed in one sweep; however, in our experience with the text finding algorithm convergence is attained in comparatively few sweeps (10 or fewer), and just a few sweeps suffice for good performance of the algorithm.

4 Application: Finding Text

We now describe a specific application of our figure-ground segmentation framework to the problem of finding text. In this application, a bottom-up procedure is used for grouping edges into composite features that are signatures of regions containing text. The next subsection describes how these features are constructed, and subsequent subsections explains how the features are grouped into factors and how the resulting factor graph is used to detect the presence of text features.

4.1 Constructing Features

We use a very simple edge detector to provide the basic elements to be grouped. First, the image is blurred slightly and decimated by a factor of three in each dimension, yielding a size of 648×864 pixels. Two kinds of edges are detected, corresponding to local maxima or minima of the horizontal and vertical image intensity derivatives. The edges are grouped into line segments, which are approximately straight and fully connected sequences of edge pixels (with between 3 and 20 pixels, which sets an appropriate range of scales for text detection). There are two kinds of line segments, those that are oriented (approximately) vertically and those that are oriented (approximately) horizontally. Vertical segments that are sufficiently close together and have opposite polarity are grouped into "weakly matched vertical edges", shown in Fig. 6(a). "Weakly matched horizontal edges" are determined in a similar way (see Fig. 6(b)). As the figure suggests, weakly matched edges are features designed to be prevalent along the borders of letter strokes.

Next we prune the set of weakly matched vertical segments to obtain our final features, "anchored vertical segments" (see Fig. 7). An anchored vertical feature is a weakly matched vertical segment whose topmost or bottommost pixel lies sufficiently close to the leftmost or rightmost pixel of a weakly matched horizontal segment. By "sufficiently close" pixels we mean that they are either

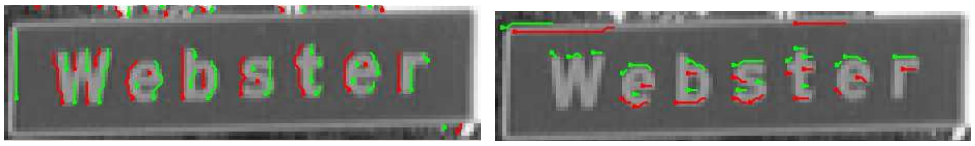

Fig. 6. Edge features used to construct text features shown on cropped image of street sign. (a) Weakly matched vertical edge segments. (b) Weakly matched horizontal edge segments. Edges shown in red and green to indicate opposite polarities.

identical or one pixel is one of the eight nearest neighbors of the other.

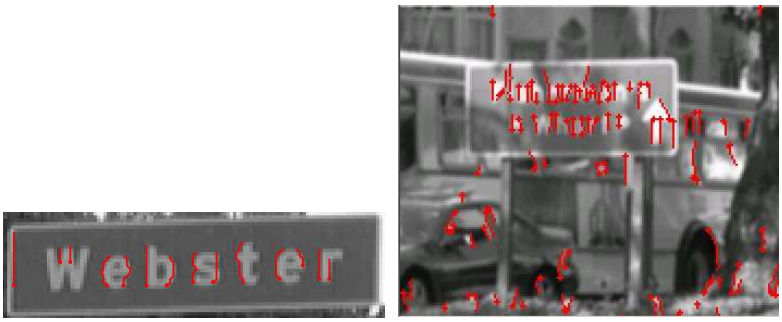

Fig. 7. Anchored verticals. (a) Anchored verticals shown for image in previous figure. (b) Same as (a) but shown for entire image. Note density and regularity of anchored verticals in text region, where bottoms and tops tend to be at the same level. By contrast, anchored verticals are scattered sparsely and irregularly throughout rest of the image.

As Fig. 7 shows, anchored verticals have a distribution on text regions that is significantly different from the distribution outside of text regions. Anchored verticals are distributed densely on text regions, and their bottoms and tops tend to be aligned to the same level. By contrast, outside of text regions, anchored verticals are distributed more sparsely and irregularly. We will exploit this differential distribution of anchored verticals (as well as other cues) to segment out text regions in an image.

4.2 Grouping Cues and Factor Construction

Having constructed a set of useful anchored vertical features that have a distinctive signature in text regions, we now proceed to construct a factor graph based on these features. In the factor graph, each anchored vertical is a variable node. Factor nodes are defined as triples of anchored verticals that may plausibly belong to one text region. As we have shown, one iteration of factor BP can be performed very simply, requiring minimal calculations (see Eq. 18). However, the search for factors is computationally intensive, since there are many possible triples of anchored verticals to consider in each image.

The search for factors is conducted by considering all triples of anchored verti-

cals that satisfy a number of criteria based on multiple cues. For any triple of anchored verticals, up to two possible factors can be formed: one from the tops of the three anchored verticals, and another from the bottoms. Here "top" and "bottom" refer to the upper and lower endpoints, respectively, of the anchored vertical. Once the factors are extracted from the image, multiple cues are used to define the "strenth" L_f of any factor f using learned distributions $P_{on}(.)$ and $P_{off}(.)$. An example of these learned distributions is shown in Fig. 8.

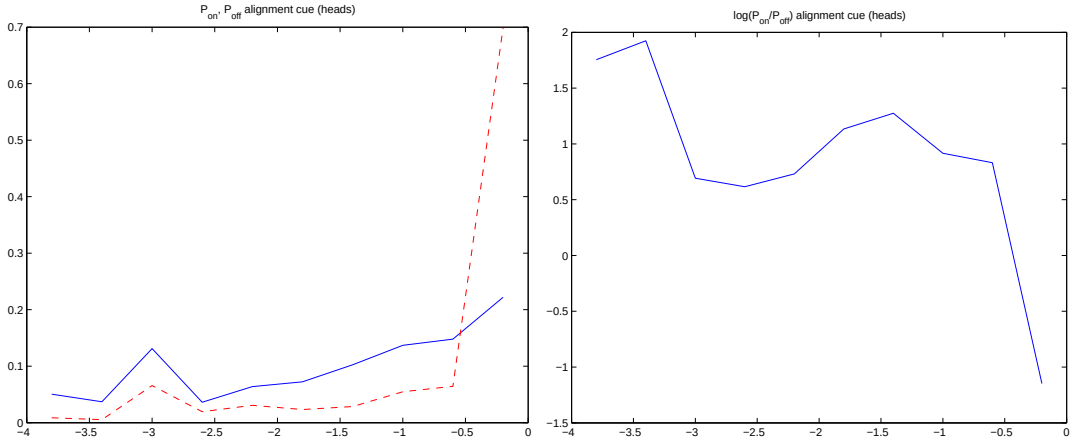

Fig. 8. (a) Conditional distributions $P_{on}(.)$ (blue) and $P_{off}(.)$ (red), measured by histogramming, for an alignment cue. (b) $logP_{on}(.)/P_{off}(.)$ for same cue.

Next we discuss the types of grouping cues used in our factor graph before defining them precisely. The following types of cues are used:

(a) Alignment: this reflects the fact that anchored vertical bottoms belonging to a group of text tend to lie along a straight line. (A similar tendency holds for anchored vertical tops, but to a lesser extent since upper and lower case letters have different heights.)

(b) Parallelism: neighboring anchored vertical segments tend to be roughly parallel to each other. However, when neighboring segments are *both* very well aligned *and* very highly parallel, this is a sign that the segments belong to a periodic structure (such as the rails of a fence or the cracks between bricks in a building) that is not text. To exclude such false positives, we have defined a joint (top) alignment and parallelism cue, which is the only two-dimensional vector cue we have used. Such a cue requires two-dimensional histograms to be learned from training data, but this cue is otherwise treated just like the other (one-dimensional scalar) cues.

(c) Orientation: most text of interest is oriented roughly horizontally. Therefore, we define an orientation cue that measures the (unsigned) orientation of the line connecting the leftmost and rightmost tops (or bottoms) of a segment triple.

(d) Color consistency: RGB color values are relatively consistent across the background of a typical text sign, since the background is usually a homogeneous color (different from the text letters themselves) and the illumination tends to be consistent across the background.

(e) Connectivity: this cue reflects the fact that anchored vertical segments are connected to more factors when they belong to text than when they belong to non-text. It is a "meta"-cue since it is calculated on the basis of factors due to all other cues.

In order to define the cues more precisely we define the following notation. The subscripts 1, 2 and 3 denote the three tops or bottoms of a triple of anchored verticals, where they are ordered from left to right across the image. Top or bottom locations will generically be referred to as pixel locations (x_i, y_i), where $i = 1$,2 or 3 (here x_i is the x-coordinate, not the figure/ground state of feature i). Line L_{13} is the line segment connecting (x_1, y_1) and (x_3, y_3).

Expressed in this notation, the following grouping cues are learned from training data and used to construct factor graphs.

(1) Bottom alignment: Let y^* be the y-coordinate of the point on L_{13} (defined in terms of bottoms) having x-coordinate equal to x_2. Then the bottom alignment cue is defined as $a_{bottom} = |y^* - y_2|$. (Perfect alignment would mean that (x_2, y_2) lies exactly on L_{13}, i.e. $a_{bottom} = 0$.)

(2) Joint parallelism and top alignment: this cue is defined as $\vec{J} = (p, a_{top})$. Here p is defined as $\max(|\theta_1 - \theta_2|, |\theta_2 - \theta_3|)$, where θ_i is the orientation of anchored vertical i. a_{top} is defined similarly to the bottom alignment cue above.

(3) Orientation of tops and of bottoms: given top (or bottom) locations (x_i, y_i), the overall orientation is defined by the orientation of the line connecting (x_1, y_1) and (x_3, y_3). The orientation is an unsigned number defined such that a horizontal line has an orientation of 0. The orientations of the tops and bottoms define two separate cues, ϕ_{top} and ϕ_{bottom}.

(4) Color consistency: denoting an RGB vector by $\vec{I} = (R, G, B)$, we define $\vec{I}_i^t$ and $\vec{I}_i^b$ to be the RGB values near the top (superscript "t") and bottom (superscript "b") of anchored vertical $i = 1, 2, 3$. Specifically, the RGB value is defined at the location 6 pixels above the top point and 6 pixels below the bottom point. Then the color consistency cue is defined as $C = \sum_i |\vec{I}_i^t - \vec{I}_i^b|$, where $|.|$ is the $L1$ norm.

(5) Connectivity: for each anchored vertical segment, this arity-1 cue is simply the total number of factors connected to the segment, denoted T.

These cues were selected by exploring a larger set of cues and choosing only

those cues with sufficient discriminating power (which we discuss at the end of this subsection).

Next we describe the selection process, which uses some of the grouping cues described above (as well as a few other cues). The first stage of the selection process removes all segments whose length is outside of a certain range (corresponding to the range of scales we are interested in). In addition, the color consistency of the segment all by itself (i.e. the value $|\vec{I}^t - \vec{I}^b|$, reflecting how well the RGB values match above and below the segment) is computed, and the segment is discarded if this value is above a threshold.

Having pruned many single anchored vertical segments in the first stage of the selection process, the goal of subsequent stages is to construct arity-3 factors. Arity-3 factors are built by combining (temporary) arity-2 factors, which are in turn constructed from pairs of segments that are sufficiently close together, and whose orientation (defined from a line connecting the midpoint of each segment) is sufficiently close to 0. We impose an ordering constraint to avoid double-counting: the first and second elements of each pair must be ordered from left to right.

Arity-3 factors are constructed by combining two arity-2 factors sharing a common segment. To construct an arity-3 factor, the following constraints must be satisfied: the three segments of the factor must be ordered from left to right; the bottom alignment cue of the three segments must be below a certain threshold; and the color consistency cue must be below a particular threshold. Once all candidate arity-3 factors have been computed, the connectivity cue is then computed for each segment, and all segments (and factors connected to the segment) are pruned for which the connectivity is sufficiently low. (All arity-2 factors are then deleted.)

These selection cues and thresholds were selected by trial and error to ensure that few "bad" factors were discarded from true text regions, while keeping the total number of factors in an image to a manageable number.

Empirical distributions $P_{on}(.)$ and $P_{off}(.)$ were first learned for all the grouping cues from a set of 71 training images as follows. Anchored vertical segments were extracted from each image, and those segments that lay on text were manually selected (by computer mouse). $P_{on}(.)$ and $P_{off}(.)$ were learned as 1-D histograms, separately for each of the grouping cues above (except for the joint 2-D cue $\vec{J}$, which was learned using a 2-D histogram). For each grouping cue, an ROC curve was computed to determine the strength of the cue in discriminating between text and non-text regions; cues demonstrated by the ROC curve to have poor discriminating power were not used. (Other measures of discrimination, such as the Kullback-Leiber distance and Chernoff information [6], would give similar but slightly different results.)

We note that the form of the selection process, in particular the choice of selection cues and thresholds, has a large effect on the resulting empirical distributions $P_{on}(.)$ and $P_{off}(.)$ of the grouping cues – in fact, these distributions are only defined relative to a specific selection process. For example, if a particular selection cue threshold is set strictly so that it removes many potential factors in the selection process, the resulting $P_{on}(.)$ and $P_{off}(.)$ distributions will have much less discriminating power than if the cue threshold is set loosely (because the strict selection threshold has already done much of the work of discriminating between on and off distributions).

4.3 Performing Inference with the Factor Graph

After all factors have been constructed, as described in the previous section, belief propagation is performed to perform inference with them. Any segment whose belief value indicates it is more likely to belong to figure than to ground is chosen as a candidate winner. For an example of this process, see Fig. 9, which shows the anchored vertical segments extracted in a typical input image, and Fig. 10 for the results of belief propagation applied to this image.

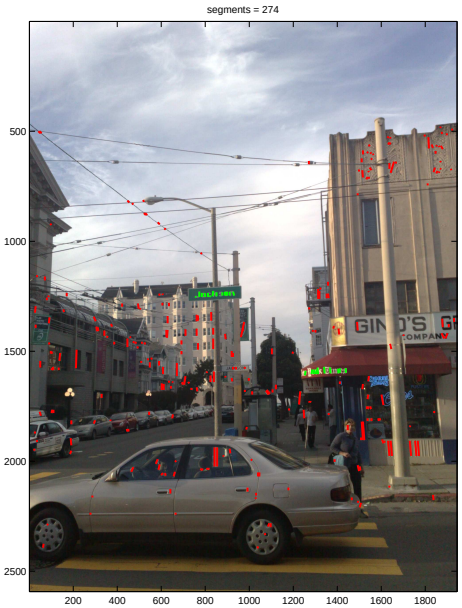

Fig. 9. Input image with anchored vertical segments colored to indicate correct (ground truth) labels: green means text and red means non-text.

Note that belief propagation correctly discards many, but not all, false positives, and detects most of the true text (while leaving some gaps).

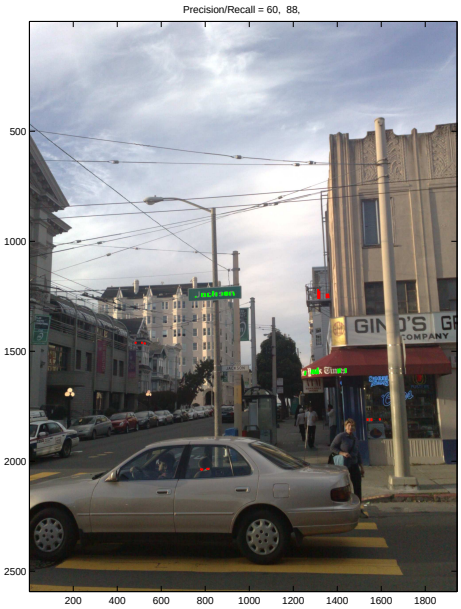

Fig. 10. Output before final processing step: all segments are shown that are designated as figure by belief propagation. Green segments are true positives and red segments are false positives.

A post-processing stage is then used to clean up the results of belief propagation, exploiting the fact that false positive segments tend to appear in isolated clusters, while true positives tend to be densely arrayed along lines of text. This post-processing stage consists of a simple convolution process, in which a sliding horizontal window (one pixel high by 100 pixels across) is applied across the image. At each location of the window, the number of candidate winner segments that it intersects is computed, and if this number is above a threshold then the window location (i.e. the center of the window) is classified as a winner. This process serves to remove isolated candidate winners and reinforce those that are densely arrayed along lines of text. See Fig. 11 for an example.

We show another result (also taken from one of the 40 test images), this time demonstrating the presence of a false positive and false negative text detection. Fig. 12 shows the input image with extracted segments; Fig. 13 shows the results of running BP on the image, which incorrectly removes some of the valid text segments and fails to remove some clusters of non-text segments. The final result is given in Fig. 14, which shows that an insufficient number of true positives were detected by BP to declare the presence of text, and that the chance alignment of features in the tree generated a false negative text detection.

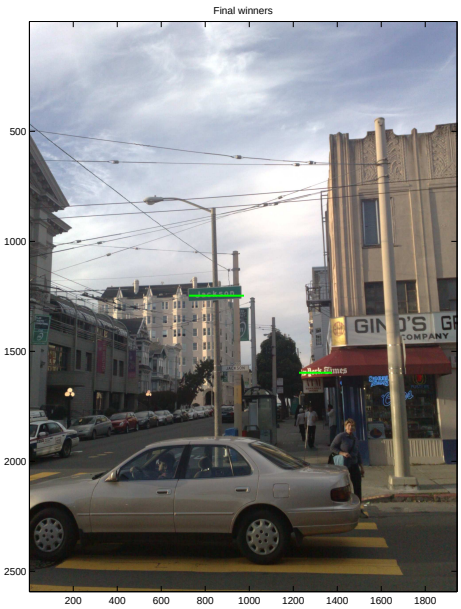

Fig. 11. Output after final processing step (convolution), with lines of text detected by algorithm shown as horizontal green lines.

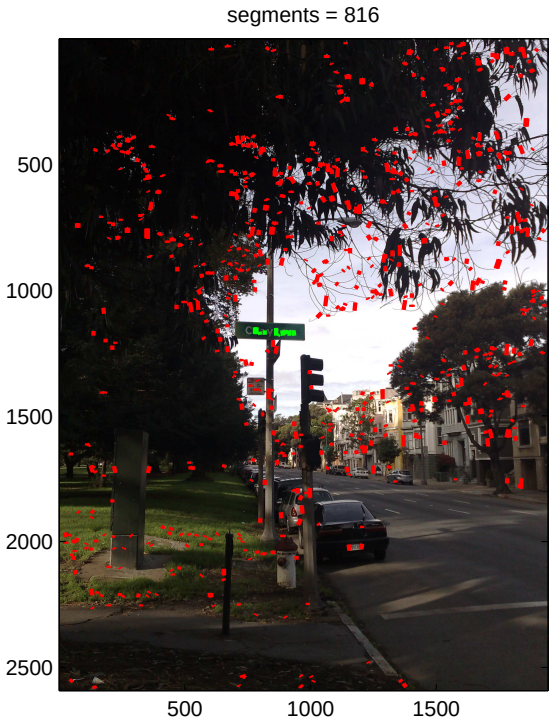

Fig. 12. A second image with anchored vertical segments colored to indicate correct (ground truth) labels: green means text and red means non-text.

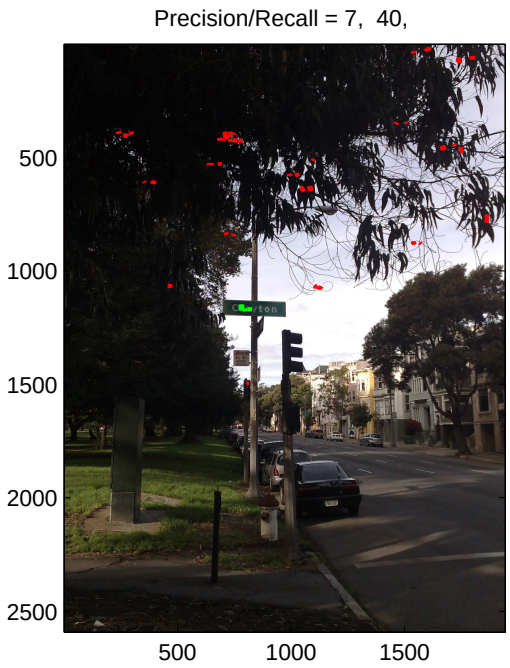

Fig. 13. Output before final processing step: all segments are shown that are designated as figure by belief propagation. Green segments are true positives and red segments are false positives.

5 Experimental Results

Our algorithm was programmed in unoptimized Matlab code, which took about one minute to search for the presence of text in each image. We trained our algorithm on 71 images and tested it on 40 images (different from the training images). In these 40 test images a total of 18142 anchored vertical segments were extracted, of which 767 belong to text and 17375 belong to non-text (based on manual inspection of the images). The results of belief propagation (before the convolution post-processing step) are that 647 out of 767 text segments were detected, with 540 false positives, corresponding to a true positive rate of 84 % and a false positive rate of 3 %.

We also evaluate the performance of the algorithm after the convolution post-processing step, quantifying detection performance at the level of entire lines of text (i.e. a street sign typically contains one line of text) rather than individual anchored vertical segments. Among the 40 test images there are 53 hits (true positive detections), 8 misses (false negatives) and 4 false alarms (false positives). (False alarms mainly came from building windows and railings.)

We note that there is some ambiguity in these performance statistics, since it is not always clear from examining an image how to distinguish between

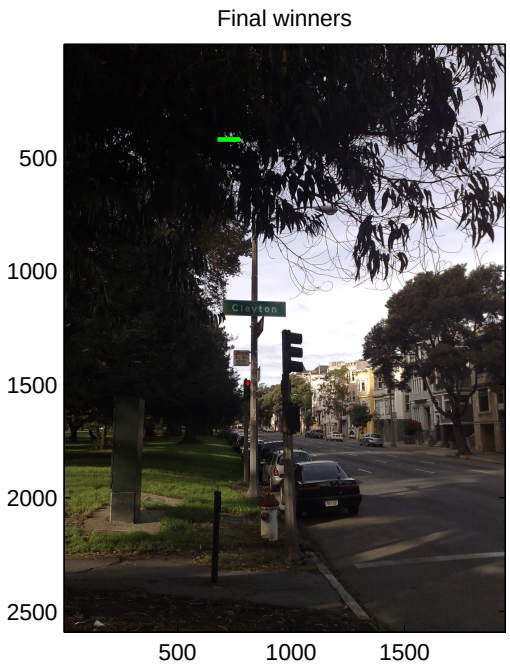

Fig. 14. Output after final processing step (convolution), with lines of text detected by algorithm shown as horizontal green lines. Notice presence of false positive and false negative text detection.

text and non-text (e.g. when the resolution is poor). Moreover, in some text regions too few anchored vertical segments are extracted, which obviously impairs performance; the solution to this problem is to improve the feature extraction procedure.

6 Discussion

We have presented a novel feature grouping framework, based on factor graphs, which emerges naturally from a simple maximum entropy model of figure-ground segmentation. The framework allows the use of multiple grouping cues, and the contributions of these cues are learned from training data. We have applied the framework to the problem of segmenting text in natural scenes, and experimental results demonstrate the feasibility of the approach.

Future work will focus on improving the reliability of text detection. The main limitation of the current approach is the use of anchored vertical segments, which are not reliably detected in some text regions (especially those containing letters with few vertical elements). One way to ameliorate this problem is to simply lower the threshold of detection so as to detect more segments; another way is to eliminate the requirement that anchored vertical segments

must lie near horizontal edge segments. Either solution will increase the number of true and false candidate segments, in which case additional cues may need to be strengthened in the selection process to decrease the total number of candidates, such as color consistency (which currently samples very few pixels but could easily be extended to sample more neighboring pixels). The connectivity cue might be strengthened to reward connections to segments lying to the left or right as opposed to segments lying above or below (since we are assuming the text is roughly horizontal). Another issue that may impair performance is the conditional independence assumption that underlies our approach (Eq. 4), which could be alleviated by adopting a conditional random field (CRF) framework [16].

In order for our algorithm to function as part of a practical system for finding and reading text, we will also have to use the text features output by it to determine appropriate bounding boxes to enclose the text, and use OCR to actually read the text. We note that the OCR stage will have the benefit of discarding some false positives which cannot be ruled out by our algorithm alone. Finally, the algorithm needs to be optimized for speed so that it can be ported to a camera cell (mobile) phone, as was accomplished in a previous version of this work [19].

7 Acknowledgment

The authors would like to acknowledge support from NIH grants EY015187-01A2 and 1 R01 EY018345-01, the National Institute on Disability and Rehabilitation Research grant H133E060001, and the Claire Giannini Fund.

References

[1] S. Agarwal, J. Lim, L. Zelnik-Manor, P. Perona, D. Kriegman, S. Belongie. "Beyond pairwise clustering." Computer Vision and Pattern Recognition (CVPR 2005). Volume 2. (2005) 838-845.

[2] C. Bishop. Pattern Recognition and Machine Learning. Springer. 2007.

[3] X. Chen and A.L. Yuille. "Detecting and reading text in natural scenes." Computer Vision and Pattern Recognition (CVPR 2004).

[4] J. Coughlan and S. Ferreira. "Finding Deformable Shapes using Loopy Belief Propagation." European Conference on Computer Vision (ECCV 2002). pp. 453-468. Copenhagen, Denmark. May 2002.

[5] J. Coughlan and H. Shen. "A fast algorithm for finding crosswalks using figure-ground segmentation." 2nd Workshop on Applications of Computer Vision, in conjunction with European Conference on Computer Vision (ECCV 2006).

[6] T.M. Cover and J.A. Thomas. Elements of Information Theory. Wiley Interscience Press. New York. 1991.

[7] P. Felzenszwalb and D. Huttenlocher. "Efficient Belief Propagation for Early Vision." Computer Vision and Pattern Recognition (CVPR 2004).

[8] J. Gao and J. Yang. "An adaptive algorithm for text detection from natural scenes." Computer Vision and Pattern Recognition (CVPR 2001).

[9] V. Ivanchenko, J. Coughlan and H. Shen. "Detecting and Locating Crosswalks using a Camera Phone." Fourth IEEE Workshop on Embedded Computer Vision, in conjunction with Computer Vision and Pattern Recognition (CVPR 2008). Anchorage, Alaska. June 2008.

[10] A. Jain and B. Tu. "Automatic text localization in images and video frames." Pattern Recognition 31(12) (1998) 2055-2076.

[11] F.R. Kschischang, B.J. Frey and H.-A. Loeliger. "Factor graphs and the sum-product algorithm." IEEETIT: IEEE Transactions on Information Theory 47 (2001).

[12] S. Kumar and M. Hebert. "Man-Made Structure Detection in Natural Images using a Causal Multiscale Random Field." Computer Vision and Pattern Recognition (CVPR 2003).

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

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

[15] K.P. Murphy, Y. Weiss and M.I. Jordan. "Loopy belief propagation for approximate inference: an empirical study." Uncertainty in AI. 1999.

[16] A. Quattoni, M. Collins and T. Darrell. "Conditional Random Fields for Object Recognition." Neural Information Processing Systems (NIPS 2004).

[17] A. Ratnaparkhi. "A simple introduction to maximum entropy models for natural language processing." Technical Report 97-08, Institute for Research in Cognitive Science, University of Pennsylvania. http://citeseer.ist.psu.edu/128751.html

[18] H. Shen and J. Coughlan. "Finding text in natural scenes by figure-ground segmentation." International Conference on Pattern Recognition (ICPR 2006). (4). (2006) 113-118.

[19] H. Shen and J. Coughlan. "Grouping Using Factor Graphs: an Approach for Finding Text with a Camera Phone." Workshop on Graph-based Representations in Pattern Recognition (GbR 2007, associated with The International Association for Pattern Recognition). June 2007. Alicante, Spain.

[20] N. Shental, A. Zomet, T. Hertz and Y. Weiss. "Pairwise clustering and graphical models." Neural Information Processing Systems (NIPS 2003).

[21] L. Sigal, M. Isard, B. H. Sigelman, M. J. Black. "Attractive People: Assembling Loose-Limbed Models using Non-parametric Belief Propagation." Neural Information Processing Systems (NIPS 2003).

[22] J. Shi and J. Malik. "Normalized cuts and image segmentation." IEEE Transactions on Pattern Analysis and Machine Intelligence 22(8) (2000) 888-905.

[23] J. Sun, H. Shum, and N. Zheng. "Stereo matching using belief propagation." European Conference on Computer Vision (ECCV 2002). pp. 510-524, 2002.

[24] V. Wu, R. Manmatha and E.M. Riseman. "Finding text in images." ACM DL. (1997) 3-12.

[25] J.S. Yedidia, W.T. Freeman and Y. Weiss,. "Understanding Belief Propagation and Its Generalizations". Exploring Artificial Intelligence in the New Millennium, ISBN 1558608117, Chap. 8, pp. 239-236, January 2003 (Science & Technology Books, also available as MERL Cambridge Research Technical Report TR TR2001-022)

[26] S.X. Yu and J. Shi. "Object-specific figure-ground segregation." Computer Vision and Pattern Recognition (CVPR 2003).

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

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

[29] D. Zhang and S. Chang. "Learning to detect scene text using a higher-order mrf with belief propagation." Computer Vision and Pattern Recognition (CVPR 2004).

[30] Y. Zheng, H. Li and D. Doermann. "Text identification in noisy document images using markov random field." International Conference on Document Analysis and Recognition. (2003).