Learning to Combine Bottom-Up and Top-Down Segmentation

Anat Levin Yair Weiss*
School of Computer Science and Engineering
The Hebrew University of Jerusalem
www.cs.huji.ac.il/ ∼{alevin,yweiss}

Abstract. Bottom-up segmentation based only on low-level cues is a notoriously difficult problem. This difficulty has lead to recent top-down segmentation algorithms that are based on class-specific image information. Despite the success of top-down algorithms, they often give coarse segmentations that can be significantly refined using low-level cues. This raises the question of how to combine both top-down and bottom-up cues in a principled manner.
In this paper we approach this problem using supervised learning. Given a training set of ground truth segmentations we train a fragment-based segmentation algorithm *which takes into account both bottom-up and top-down cues simultaneously*, in contrast to most existing algorithms which train top-down and bottom-up modules separately. We formulate the problem in the framework of Conditional Random Fields (CRF) and derive a feature induction algorithm for CRF, which allows us to efficiently search over thousands of candidate fragments. Whereas pure top-down algorithms often require hundreds of
fragments, our simultaneous learning procedure yields algorithms with a handful of fragments that are combined with low-level cues to efficiently compute high quality segmentations.

1 Introduction

Figure 1 (replotted from [2]) illustrates the importance of combining top-down and bottom-up segmentation. The leftmost image shows an image of a horse and the middle column show three possible segmentations based only on low-level cues. Even a sophisticated bottom-up segmentation algorithm (e.g. [12, 16]) has difficulties correctly segmenting this image.

The difficulty in pure low-level segmentation has led to the development of top-down, class-specific segmentation algorithms [3, 11, 22, 19]. These algorithms fit a deformable model of a known object (e.g. a horse) to the image - the shape of the deformed model gives an estimate of the desired segmentation. The right-hand column of figure 1 shows a top-down segmentation of the horse figure obtained by the algorithm of [3]. In this algorithm, image fragments from horses in a training database are correlated with the novel image. By combining together the segmentations of the fragments, the novel image is segmented. As can be seen, the top-down segmentation is better than any of the bottom-up segmentations but still misses important details.

* Research supported by the Israel Science Foundation, and by the EU under the DIRAC Project. EC Contract No.027787

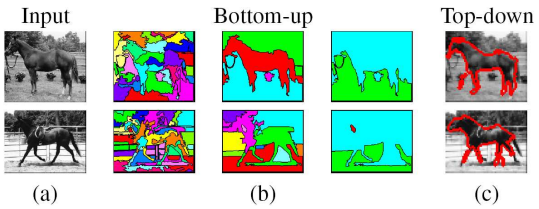
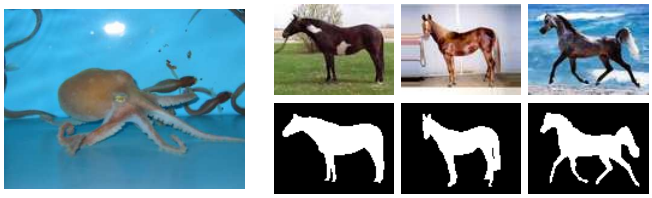


Fig. 1. The relative merits of the bottom-up and the top-down approaches, replotted from [2]. (a) Input image. (b) The bottom-up hierarchical segmentation at three different scales. (c) The top-down approach provides a meaningful approximation for the figureground segmentation of the image, but may not follow exactly image discontinuities.

In recent years, several authors have therefore suggested combining top-down and bottom-up segmentation [2, 21, 17, 6]. Borenstein et al. [2] choose among a discrete set of possible low-level segmentations by minimizing a cost function that includes a bias towards the top-down segmentation. In the *image parsing* framework of Tu et al. [17] object-specific detectors serve as a proposal distribution for a data-driven Monte-Carlo sampling over possible segmentations. In the *OBJ-CUT* algorithm [6] a layered pictorial structure is used to define a bias term for a graph-cuts energy minimization algorithm (the energy favors segmentation boundaries occurring at image discontinuities).

These recent approaches indeed improve the quality of the achieved segmentations by combining top-down and bottom-up cues at run-time. However, the training of the bottom-up and top-down modules is performed *independently*. In the work of Borenstein and colleagues, training the top-down module consists of choosing a set of fragments from a huge set of possible image fragments. This training is performed *without taking into account low-level cues*. In the image parsing framework [17], the top-down module are object detectors trained using AdaBoost to maximize detection performance. Again, this training is performed without taking into account low-level cues. In the OBJ-CUT algorithm, the training of the algorithm is based on a set of learned layered pictorial structures [6]. These learned models are then used to define a detection cascade (which calculates putative part locations by comparing the image to a small number of templates) and a bounding box for the relative part locations. Again, the choice of which templates to apply to a given images is performed independent of the low-level segmentation cues.

Figure 2(a) shows a potential disadvantage of training the top-down model while ignoring low-level cues. Suppose we wish to train a segmentation algorithm for octopi. Since octopi have 8 tentacles and each tentacle has multiple degrees of freedom, any top-down algorithm would require a very complex deformable template to achieve reasonable performance. Consider for example the top-down algorithm of Borenstein and Ullman [3] which tries to cover the segmentations in the dataset with a subset of image fragments. It would obviously require a huge number of fragments to achieve reasonable performance. Similarly, the layered pictorial structure algorithm of Kumar et al. [6] would require a large number of parts and a complicated model for modeling the allowed spatial configurations.



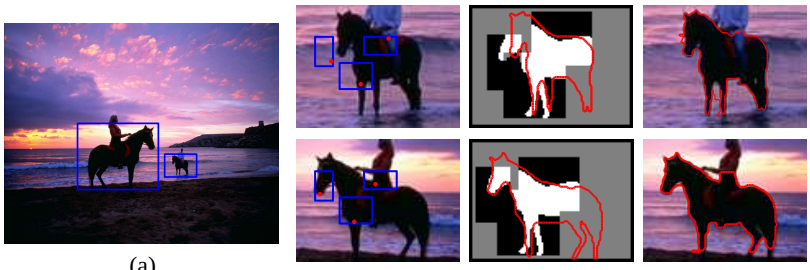
(a) (b)

Fig. 2. (a) Octopi: Combining low-level information can significantly reduce the required complexity of a deformable model. (b) Examples from horses training data. Each training image is provided with its segmentation mask.

While Octopi can appear in a large number of poses, their low-level segmentation can be easy since their color is relatively uniform and (depending on the scene) may be distinct from the background. Thus an algorithm that trains the top-down module while taking into account the low-level cues can choose to devote far less resources to the deformable templates. The challenge is to provide a principled framework for simultaneous training of the top-down and bottom-up segmentation algorithms.

In this paper we provide such a framework. The algorithm we propose is similar *at run-time* to the OBJ-CUT and the Borenstein et al. algorithms. As illustrated in figure 3, at run-time a novel image is scanned with an object detector which tries all possible subimages until it finds a subimage that is likely to contain the object (for most of the databases in this paper the approximate location was known so no scanning was performed). Within that subimage we search for object parts by performing normalized correlation with a set of fragments (each fragment scans only a portion of the subimage where it is likely to occur thus modeling the spatial interaction between fragment locations). The location of a fragment gives rise to a local bias term for an energy function. In addition to the local bias, the energy function rewards segmentation boundaries occurring at image discontinuities. The final segmentation is obtained by finding the global minimum of the energy function.

While our algorithm is similar at run-time to existing segmentation algorithms, the *training* method is unique in that it *simultaneously takes into account low-level and high-level cues*. We show that this problem can be formulated in the context of Conditional Random Fields [8, 7] which leads to a convex cost function for simultaneous training of both the low-level and the high-level segmenter. We use the CRFs formulation to derive a novel fragment selection algorithm, which allows us to efficiently learn models with a small number of fragments. Whereas pure top-down algorithms often require hundreds of fragments, our simultaneous learning procedure yields algorithms with a handful of fragments that are combined with low-level cues to efficiently compute high quality segmentations.



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

Fig. 3. System overview: (a) Detection algorithm applied to an input image (b) Fragments search range, dots indicate location of maximal normalized correlation (c) Fragments local evidence, overlaid with ground truth contour (d) Resulting segmentation contour

2 Segmentation using Conditional Random Fields

Given an image I, we define the energy of a binary segmentation map x as:

$$E(x;I) = \nu \sum_{i,j} w_{ij}|x(i) - x(j)| + \sum_{k} \lambda_k |x - x_{F_k,I}| \quad (1)$$

This energy is a combination of a pairwise low-level term and a local class-dependent term.

The low level term is defined via a set of affinity weights $w(i,j)$. $w(i,j)$ are high when the pixels (i,j) are similar and decrease to zero when they are different. Similarity can be defined using various cues including intensity, color, texture and motion as used for bottom up image segmentation [12]. Thus minimizing $\sum_{i,j} w_{ij}|x(i) - x(j)|$ means that labeling discontinuities are cheaper when they are aligned with the image discontinuities. In this paper we used 8-neighbors connectivity, and we set:

$$w_{ij} = \frac{1}{1 + \sigma d_{ij}^2}$$

where d_{ij} is the RGB difference between pixels i and j and $\sigma = 5 \cdot 10^4$.

The second part of eq 1 encodes the local bias, defined as a sum of local energy terms each weighted by a weight λ_k. Following the terminology of Conditional Random Fields, we call each such local energy term a feature. In this work, these local energy terms are derived from image fragments with thresholds. To calculate the energy of a segmentation, we shift the fragment over a small window (10 pixels in each direction) around its location in its original image. We select the location in which the normalized correlation between the fragment and the new image is maximal (see Fig 3(b)). The feature is added to the energy, if this normalized correlation is large than a threshold. Each fragment is associated with a mask fragment x_F extracted from the training set (Fig 9 shows some fragments examples). We denote by $x_{F,I}$ the fragment mask x_F placed over the image I, according to the maximal normalized correlation location. For each fragment we add a term to the energy function which penalizes for the number of

pixels for which x is different from the fragment mask $x_{F,I}$, $|x-x_{F,I}| = \sum_{i \in F} |x(i) - x_{F,I}(i)|$. Where $i \in F$ means the pixel i is covered by the fragment F after the fragment was moved to the maximal normalized correlation location (see Fig 3(c)).

Our goal in this paper is to learn a set of fragments $\{F_k\}$, thresholds and weights $\{\lambda_k\}$, ν that will favor the true segmentation. In the training stage the algorithm is provided a set of images $\{I_t\}_{t=1:T}$ and their binary segmentation masks $\{x_t\}_{t=1:T}$, as in figure 2(b). The algorithm needs to select features and weights such that minimizing the energy with the learned parameters will provide the desired segmentation.

2.1 Conditional Random Fields

Using the energy (eq. 1) we define the likelihood of the labels x conditioned on the image I as

$$P(x|I) = \frac{1}{Z(I)} e^{-E(x;I)} \quad \text{where:} \quad Z(I) = \sum_x e^{-E(x;I)}$$

That is, x forms a Conditional Random Field (CRF) [8]. The goal of the learning process is to select a set of fragments $\{F_k\}$, thresholds and weights $\{\lambda_k\}$, ν that will maximize the sum of the log-likelihood over training examples: $\ell(\vec{\lambda}, \nu; \vec{F}) = \sum_t \ell^t(\vec{\lambda}, \nu; \vec{F})$

$$\ell^t(\vec{\lambda}, \nu; \vec{F}) = \log P(x_t|I_t; \vec{\lambda}, \nu, \vec{F}) = -E(x_t; I_t, \vec{\lambda}, \nu, \vec{F}) - \log Z(I_t; \vec{\lambda}, \nu, \vec{F}) \quad (2)$$

The idea of the CRF log likelihood is to select parameters that will maximize the likelihood of the ground truth segmentation for training examples. Such parameters should minimize the energy of the true segmentations x_t, while maximizing the energy of all other configurations.

The CRF formulation has proven useful in many vision applications [7, 15, 14, 4, 5]. Below we review several properties of the CRF log likelihood:

1. For a given features set $\vec{F} = [F_1, ..., F_K]$, if there exists a parameter set $\vec{\lambda}^* = [\lambda_1^*, .., \lambda_K^*]$, ν^* for which the minimum of the energy function is exactly the true segmentation: $x_t = \arg\min_x E(x; I_t, \vec{\lambda}^*, \nu^*, \vec{F})$. Then selecting $\alpha\vec{\lambda}^*, \alpha\nu^*$ with $\alpha \to \infty$ will maximize the CRF likelihood, since: $P(x_t|I_t; \alpha\vec{\lambda}^*, \alpha\nu^*, \vec{F}) = 1$ (see [10]).
2. The CRF log likelihood is *convex* with respect to the weighting parameters λ_k, ν as discussed in [8].
3. The derivative of the log-likelihood with respect to the coefficient of a given feature is known to be the difference between the expected feature response, and the observed one. This can be expressed in a simple closed form way as:

$$\begin{aligned} \frac{\partial \ell^t(\vec{\lambda}, \nu; \vec{F})}{\partial \lambda_k} &= \frac{\partial \log P(x_t|I_t; \vec{\lambda}, \nu, \vec{F})}{\partial \lambda_k} \\ &= \sum_{i \in F_k} \sum_r p_i(r)|r - x_{F_k,I_t}(i)| - \sum_{i \in F_k} |x_t(i) - x_{F_k,I_t}(i)| \\ &= < |x_t - x_{F_k,I_t}| >_{P(x_t|I_t;\vec{\lambda},\nu,\vec{F})} - < |x_t - x_{F_k,I_t}| >_{Obs} \end{aligned} \quad (3)$$

$$\frac{\partial \ell^t(\vec{\lambda},\nu;\vec{F})}{\partial \nu} = \frac{\partial \log P(x_t|I_t;\vec{\lambda},\nu,\vec{F})}{\partial \nu}$$
$$= \sum_{ij}\sum_{rs} p_{ij}(r,s) w_{ij}|r-s| - \sum_{ij} w_{ij}|x_t(i)-x_t(j)|$$
$$= < |x_t(i)-x_t(j)| >_{P(x_t|I_t;\vec{\lambda},\nu,\vec{F})} - < |x_t(i)-x_t(j)| >_{Obs} \quad (4)$$

Where $p_i(r), p_{ij}(r,s)$ are the marginal probabilities $P(x_i = r|I_t;\vec{\lambda},\nu,\vec{F})$,$P(x_i = r, x_j = s|I_t;\vec{\lambda},\nu,\vec{F})$.

Suppose we are given a set of features $\vec{F} = [F_1, ...F_K]$ and the algorithm task is to select weights $\vec{\lambda} = [\lambda_1, .., \lambda_K]$, ν that will maximize the CRF log likelihood. Given that the cost is convex with respect to $\vec{\lambda}$, ν it is possible to randomly initialize the weights vector and run gradient decent, when the gradients are computed using equations 3,4. Note that gradient decent can be used for selecting the optimal weights, without computing the explicit CRF log likelihood (eq 2).

Exact computation of the derivatives is intractable, due to the difficulty in computing the marginal probabilities $p_i(r), p_{ij}(r,s)$. However, any approximate method for estimating marginal probabilities can be used. One approach for approximating the marginal probabilities is using Monte Carlo sampling, like in [4, 1]. An alternative approach is to approximate the marginal probabilities using the beliefs output of sum product belief propagation or generalized belief propagation. Similarly, an exact computation of the CRF log likelihood (eq 2) is challenging due to the need to compute the log-partition function $Z(I) = \int_x e^{-E(x;I)}$. Exact computation of $Z(I)$ is in general intractable (except for tree structured graphs). However, approximate inference methods can be used here as well, such as the Bethe free energy or the Kikuchi approximations [20]. Monte-Carlo methods can also be used. In this work we have approximated the marginal probabilities and the partition function using sum product tree-reweighted belief propagation [18], which provides a rigorous bound on the partition function, and has better convergence properties than standard belief propagation. Tree reweighted belief propagation is described in the Appendix.

2.2 Features Selection

The learning algorithm starts with a large pool of candidate local features.

In this work we created a $2,000$ features pool, by extracting image fragments from training images. Fragments are extracted at random sizes and random locations. The learning goal is to select from the features pool a small subset of features that will construct the energy function E, in a way that will maximize the conditional log likelihood $\sum_t \log P(x_t|I_t)$. Since the goal is to select a small subset of features out of a big pool, the required learning algorithm for this application is more than a simple gradient decent.

Let E_k denote the energy function at the k'th iteration. The algorithm initializes E_0 with the pairwise term and adds local features in an iterative greedy way, such that in each iteration a single feature is added: $E_k(x;I) = E_{k-1}(x;I) + \lambda_k|x - x_{F_k,I}|$. In each iteration we would like to add the feature F_k that will maximize the conditional

log likelihood. We denote by $L_k(F,\lambda)$ the possible likelihood if the feature F, weighted by λ, is added at the k'th iteration:

$$L_k(F,\lambda) = \ell(\vec{\lambda}_{k-1}, \lambda, \nu; \vec{F}_{k-1}, F) = \sum_t \log P\left(x_t | I_t;\ E_{k-1}(x_t; I_t) + \lambda |x_t - x_{F,I_t}|\right)$$

Straightforward computation of the likelihood improvement is not practical since in each iteration, it will require inference for each candidate feature and for every possible weight λ we may assign to this feature. For example, suppose we have 50 training images, we want to scan $2,000$ features, 2 possible λ values, and we want to perform 10 features selection iterations. This results in $2,000,000$ inference operations. Given that each inference operation itself is not a cheap process, the resulting computation can not be performed in a reasonable time. However, we suggest that by using a first-order approximation to the log likelihood, one can efficiently learn a small number of effective features. Similar ideas in other contexts have been proposed by [23, 9, 13].

Observation*: A first order approximation to the conditional log likelihood can be computed efficiently, without a specific inference process per feature.*

Proof:

$$L_k(F,\lambda) \approx \ell_{k-1}(\vec{\lambda}_{k-1}, \nu) + \lambda \left.\frac{\partial L_k(F,\lambda)}{\partial \lambda}\right|_{\lambda=0} \tag{5}$$

where

$$\left.\frac{\partial L_k(F,\lambda)}{\partial \lambda}\right|_{\lambda=0} = \sum_t < |x_t - x_{F,I_t}| >_{P(x_t|I_t;\vec{\lambda}_{k-1},\nu,\vec{F}_{k-1})} - < |x_t - x_{F,I_t}| >_{Obs} \tag{6}$$

and $\ell_{k-1}(\vec{\lambda}_{k-1}, \nu) = \sum_t \log P(x_t|I_t; E_{k-1})$. We note that computing the above first order approximation requires a single inference process on the previous iteration energy E_{k-1}, from which the local beliefs (approximated marginal probabilities) $\{b_{t,i}^{k-1}\}$ are computed. Since the gradient is evaluated at the point $\lambda = 0$, it can be computed using the $k-1$ iteration beliefs and there is no need for a specific inference process per feature.□

Computing the first order approximation for each of the training images is linear in the filter size. This enables scanning thousands of candidate features within several minutes. As evident from the gradient formula (eq 6) and demonstrated in the experiments section, the algorithm tends to select fragments that: (1) have low error in the training set (since it attempts to minimize $< |x_t - x_{F,I_t}| >_{Obs}$) and (2) are not already accounted for by the existing model (since it attempts to maximize $< |x_t - x_{F,I_t}| >_{P(x_t|I_t;\vec{\lambda}_{k-1},\nu,\vec{F}_{k-1})}$).

Once the first order approximations have been calculated we can select a small set of the features $F_{k_1} ... F_{k_N}$ with the largest approximated likelihood gains. For each of the selected features, and for each of a small discrete set of possible λ values $\lambda \in \{\lambda^1, ..., \lambda^M\}$, we run an inference process and evaluate the explicit conditional log likelihood. The optimal feature (and scale) is selected and added to the energy function E. The features selection steps are summarized in Algorithm 1.

Once a number of features have been selected, we also optimize the choice of weights $\{\lambda_k\}, \nu$ using several gradient decent steps. Since the cost is convex with respect to the weights a local optimum is not an issue.

Algorithm 1 : Features Selection

Initialization: $E_0(x_t; I_t) = \nu \sum_{ij} w_{ij}|x_t(i) - x_t(j)|$.
for k=1 to maxItr

1. Run tree-reweighted belief propagation using the $k-1$ iteration energy $E_{k-1}(x_t; I_t)$. Compute local beliefs $\{b_{t,i}^{k-1}\}$.
2. For each feature F compute the approximated likelihood using eq 5.
   Select the N features $F_{k_1}...F_{k_N}$ with largest approximated likelihood gains.
3. For each of the features $F_{k_1}...F_{k_N}$, and for each scale $\lambda \in \{\lambda^1, ..., \lambda^M\}$, run tree-reweighted belief propagation and compute the likelihood $L_k(F_{k_n}, \lambda^m)$
4. Select the feature and scale with maximal likelihood gain:

$$(F_{k_n}, \lambda^m) = \arg \max_{n=1:N,\ m=1:M} L_k(F_{k_n}, \lambda^m)$$

Set $\lambda_k = \lambda^m,\ F_k = F_{k_n}\ ,\ E_k(x; I) = E_{k-1}(x; I) + \lambda_k|x - x_{F_k,I}|$.

2.3 Bounding the log-likelihood gain

Consider the k'th feature selection iteration. In this iteration one is given an energy function $E_{k-1}(x; I)$, and tests the option of adding the energy function a feature (F, λ):

$$E_k(x; I) = E_{k-1}(x; I) + F(x; I)$$

where we use $F(x; I)$ as a shortcut for $\lambda|x - x_{F,I}|$. The new log-likelihood can be expressed as:

$$L_k(F, \lambda) = \ell_{k-1} + \sum_t -\ln < e^{-F(x_t;I_t)} >_{P_{k-1}(x_t,I_t)} - < F(x_t; I_t) >_{Obs}$$

The above formula is *exact*, and is easily derived from the fact that if $P_1(x) = \frac{1}{Z_1}e^{-E_1(x)}$ and $P_2(x) = \frac{1}{Z_2}e^{-E_2(x)}$, then the ratio of partition functions is $Z_2/Z_1 =< e^{E_1-E_2} >_{P_1}$.

This is closely related to the first-order approximation. Assuming that F is closely concentrated around its mean value (e.g. at small temperatures) one can replace the expectation and the log: $\ln < e^{-F} > \approx < -F >$, and get the familiar first order approximation:

$$L_k(F, \lambda) \approx \ell_{k-1} + \sum_t < F(x_t; I_t) >_{P_{k-1}(x_t,I_t)} - < F(x_t; I_t) >_{Obs}$$

A second result is that the first-order approximation is *a rigorous upper bound* on the log-likelihood:

$$-\ln < e^{-F} >_{P_{k-1}} \leq \ < F >_{P_{k-1}}$$

This follows from Jensen's inequality and can also be proven directly from the convexity of the log likelihood. Since the first order approximation is an upper bound, we know that we do not have to consider fragments whose first order approximation is small, as such fragments must have smaller likelihood gain.

3 Experiments

In our first experiment we tried to segment a synthetic octopus dataset. Few sample images are shown in Fig 4. It's clear that our synthetic octopi are highly non rigid objects. Any effort to fully cover all the octopi tentacles with fragments (like [2, 11, 6]), will require a huge number of different fragments. On the other hand, there is a lot of edges information in the images that can guide the segmentation. The first feature selected by our algorithm is located on the octopi head, which is a rigid part common to all examples. This single feature, combined with pairwise constraints was enough to propagate the true segmentation to the entire image. The MAP segmentation given the selected feature is shown in Fig 4.

We then tested our algorithm on three real datasets, of horses [3, 2], cars and cows [11]. We measured the percentage of mislabeled pixels in the segmented images on training and testing images, as more fragments are learned. Those are shown for the horses in Fig 5(a), for the cars in Fig 5(b), and for the cows in Fig 5(c). Note that after selecting 3 fragments our algorithm performs at over 95% correct on test data for the horse dataset. The algorithm of Borenstein et al. [2] performed at 95% for pixels in which its confidence was over 0.1 and at 66% for the rest of the pixels. Thus our overall performance seems comparable (if not better) even though we used far less fragments. The OBJ-CUT algorithm also performs at around 96% for a subset of this dataset using a LPS model of 10 parts whose likelihood function takes into consideration chamfer distance and texture and is therefore significantly more complex than normalized correlation.

In the horses, cars and cows experiments we rely on the fact that we are searching for a shape in the center of the window, and used an additional local feature predicting that the pixels lying on the boundary of the subimage should be labeled as background.

In Fig 6 we present several testing images of horses, the ground truth segmentation, the local features responses and the inferred segmentation. While low level information adds a lot of power to the segmentation process, it can also be misleading. For example, the image on the right of Fig 10 demonstrates the weakness of the low level information.

In Fig 7 we present segmentation results on cars test images, for an energy function consisting of 2 features, and in Fig 8 we present segmentation results on cows test images, for an energy function consisting of 4 features. The segmentation in the cows case is not as good as in the horses' case, especially in the legs. We note that in most of these examples the legs are in a different color than the cow body, hence the low-level information can not easily propagate labeling from the cow body to its legs. The low level cue we use in the work is quite simple- based only on the RGB difference between neighboring pixels. It's possible that using more sophisticated edges detectors [12] will enable a better propagation.

The first 3 horse fragments that were selected by the algorithm are shown in Fig 9. In Fig 10 we illustrate the first 3 training iterations on several training images. Quite a good segmentation can be obtained even when the response of the selected features does not cover the entire image. For example the first fragment was located around the horse's front legs. As can be seen in the first 3 columns of Fig 10, some images can be segmented quite well based on this single local feature. We can also see that the algorithm tends to select new features in image areas that were mislabeled in the

previous iterations. For example, in several horses (mainly the 3 middle columns) there is still a problem in the upper part, and the algorithm therefore selects a second feature in the upper part of the horse. Once the second fragment was added there are still several mislabeled head areas (see the 3 right columns), and as a result the 3rd fragment is located on the horse head.

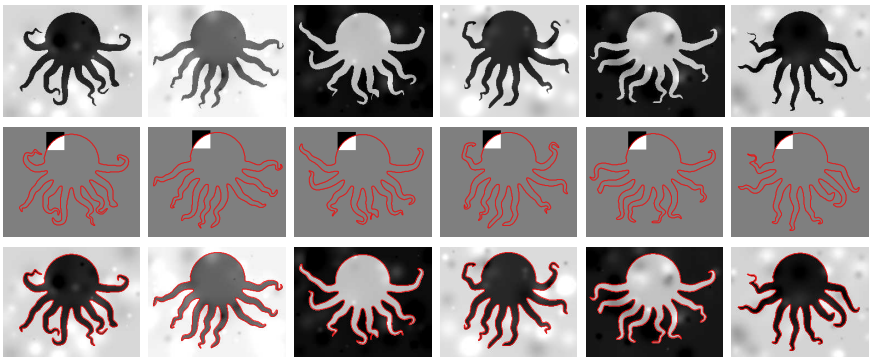


Fig. 4. Results on synthetic octopus data. Top: Input images. Middle: response of the local feature, with the ground truth segmentation contour overlaid in red. Bottom: MAP segmentation contour overlaid on input image.

4 Discussion

Evidence from human vision suggests that humans utilize significant top-down information when performing segmentation. Recent works in computer vision also suggest that segmentation performance in difficult scenes is best approached by combining top-down and bottom-up cues. In this paper we presented an algorithm that learns how to combine these two disparate sources of information into a single energy function. We showed how to formulate the problem as that of estimation in Conditional Random Fields. We used the CRF formulation to derive a novel fragment selection algorithm that allowed us to efficiently search over thousands of image fragments for a small number of fragments that will improve the segmentation performance. Our learned algorithm achieves state-of-the-art performance with a small number of fragments combined with very rudimentary low-level cues.

Both the top-down module and the bottom-up module that we used can be significantly improved. Our top-down module translates an image fragment and searches for the best normalized correlation, while other algorithms also allow rescaling and rotation of the parts and use more sophisticated image similarity metrics. Our bottom-up module uses only local intensity as an affinity function between pixels, whereas other algorithms have successfully used texture and contour as well. In fact, one advantage of the CRFs framework is that we can learn the relative weights of different affinity

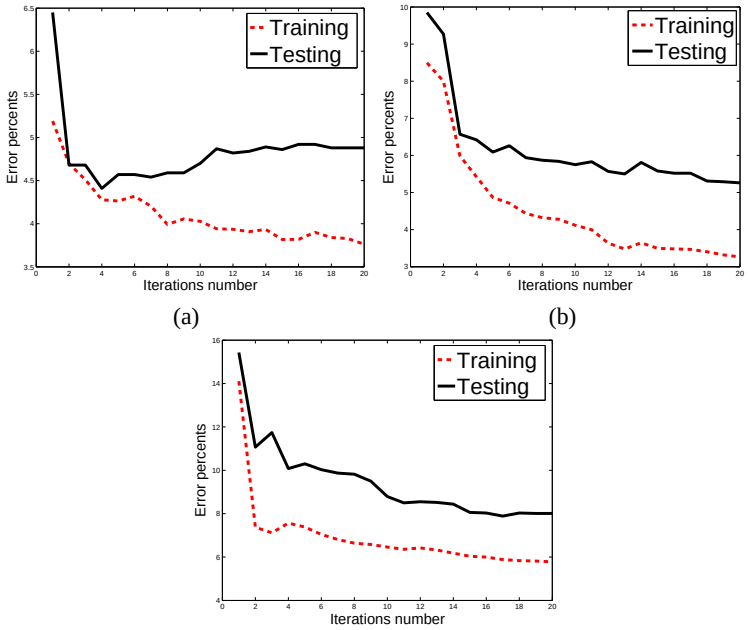


Fig. 5. Percents of miss-classified pixels: (a) Horses data (b) Cars data (c) Cows data. Note that after 4 fragments our algorithm performs at over 95% correct on test data for the horse dataset. These results are comparable if not better than [2, 6] while using a simpler model.

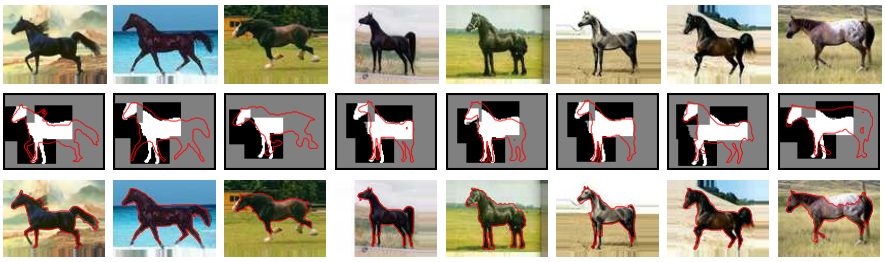


Fig. 6. Testing results on horses data. Top row: Input images. Second row: Response of the local features and the boundary feature, with the ground truth segmentation contour overlaid in red. Bottom row: MAP segmentation contour overlaid on input image.

functions. We believe that by improving both the low-level and high-level cues we will obtain even better performance on the challenging task of image segmentation.

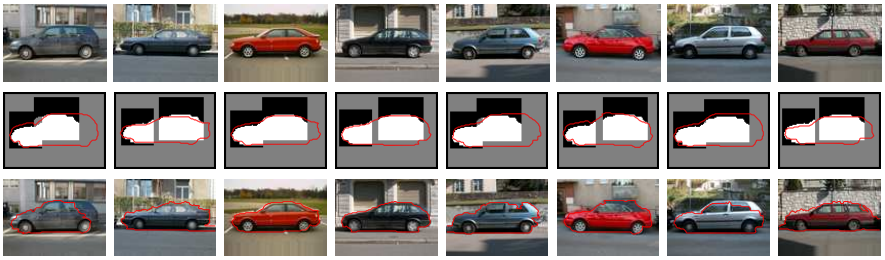


Fig. 7. Testing results on cars data. Top row: Input images. Second row: Response of the local features and the boundary feature, with the ground truth segmentation contour overlaid in red. Bottom row: MAP segmentation contour overlaid on input image.

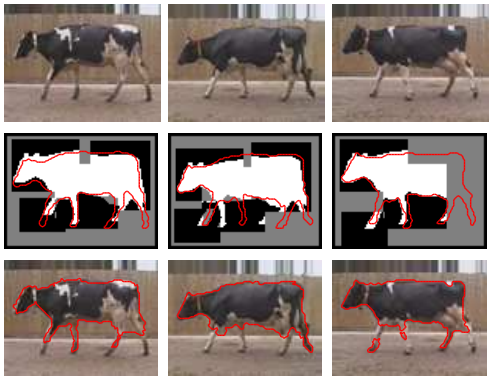


Fig. 8. Testing results on cows' data with 4 features. Top row: Input images. Second row: Response of the local features and the boundary feature, with the ground truth segmentation contour overlaid in red. Bottom row: MAP segmentation contour overlaid on input image.

Fig. 9. The first 3 horse fragments selected by the learning algorithm

5 Appendix: Tree-reweighted Belief Propagation and Tree-reweighted Upper Bound

In this section we summarize the basic formulas from [18] for applying tree-rewighted belief propagation and for computing the tree-rewighted upper bound.

For a given graph G, we let $\boldsymbol{\mu_e} = \{\mu_e | e \in E(G)\}$ represent a vector of edge appearance probabilities. That is, μ_e is the probability that the edge e appears in a spanning tree

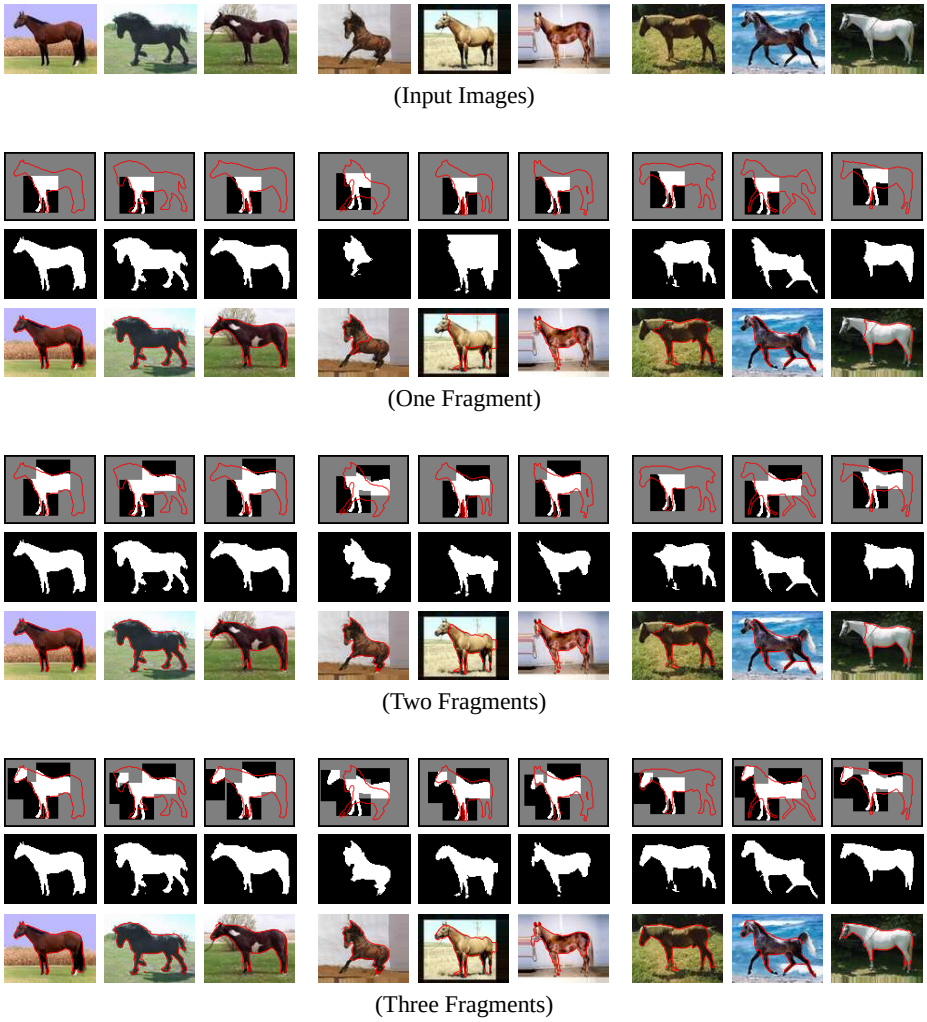


Fig. 10. Training results on horses data. For each group: Top row - response of the local features and the boundary feature, with the ground truth segmentation contour overlaid in red. Middle row - MAP segmentation. Bottom row - MAP segmentation contour overlaid on input image.

of the graph G, chosen under a particular distribution on spanning trees. For 2D-grid graphs with 4-neighbors connectivity a reasonable choice of edges distributions is $\boldsymbol{\mu_e} = \left\{\mu_e = \frac{1}{2} | e \in E(G)\right\}$ and for 8-neighbors connectivity, $\boldsymbol{\mu_e} = \left\{\mu_e = \frac{1}{4} | e \in E(G)\right\}$.

The edge appearance probabilities are used for defining a tree-rewighted massages passing scheme. Denote the graph potentials as: $\Psi_i(x_i) = e^{-E_i(x_i)}$, $\Psi_{ij}(x_i, x_j) = e^{-E_{ij}(x_i, x_j)}$, and assume $P(x)$ can be factorized as: $P(x) \propto \prod_i \Psi_i(x_i) \prod_{i,j} \Psi_{ij}(x_i, x_j)$. The tree-rewighted massages passing scheme is defined as follows:

1. Initialize the messages $m^0 = m^0_{ij}$ with arbitrary positive real numbers.
2. For iterations n=1,2,3,... update the messages as follows:

$$m^{n+1}_{ji}(x_i) = \kappa \sum_{x'_j} exp(-\frac{1}{\mu_{ij}} E_{ij}(x_i, x'_j) - E_j(x'_j)) \left\{ \frac{\prod_{k \in \Gamma(j) \setminus i} \left[m^n_{kj}(x'_j) \right]^{\mu_{kj}}}{\left[m^n_{ij}(x'_j) \right]^{(1-\mu_{ji})}} \right\}$$

where κ is a normalization factor such that $\sum_{x_i} m^n_{ji}(x_i) = 1$.

The process converges when $m^{n+1}_{ji} = m^n_{ji}$ for every ij.

Once the process has converged, the messages can be used for computing the local and pairwise beliefs:

$$b_i(x_i) = \kappa \ exp(-E_i(x_i)) \prod_{k \in \Gamma(i)} [m_{ki}(x_i)]^{\mu_{ki}} \tag{7}$$

$$b_{ij}(x_i, x_j) = \kappa \ exp(-\frac{1}{\mu_{ij}} E_{ij}(x_i, x_j) - E_i(x_i) - E_j(x_j)) \frac{\prod_{k \in \Gamma(i) \setminus j} [m_{ki}(x_i)]^{\mu_{ki}}}{[m_{ji}(x_i)]^{(1-\mu_{ij})}} \frac{\prod_{k \in \Gamma(j) \setminus i} [m_{kj}(x_j)]^{\mu_{kj}}}{[m_{ij}(x_j)]^{(1-\mu_{ji})}} \tag{8}$$

We define a pseudo-marginals vector $\vec{\mathbf{q}} = \{q_i, q_{ij}\}$ as a vector satisfying: $\sum_{x_i} q_i(x_i) = 1$ and $\sum_{x_j} q_{ij}(x_i, x_j) = q_i(x_i)$. In particular, the beliefs vectors in equations 7,8 are a peseudo-marginals vector. We use the peseudo-marginals vectors for computing the tree-rewighted upper bound.

Denote by θ the energy vector $\theta = \{E_i, E_{ij}\}$. We define an "average energy" term as: $\vec{\mathbf{q}} \cdot \theta = \sum_i \sum_{x_i} -q_i(x_i) E_i(x_i) + \sum_{ij} \sum_{x_i, x_j} -q_{ij}(x_i, x_j) E_{ij}(x_i, x_j)$. We define the single node entropy: $H_i(q_i) = -\sum_{x_i} q_i(x_i) \log q_i(x_i)$. Similarly, we define the mutual information between i and j, measured under q_{ij} as: $I_{ij}(q_{ij}) = \sum_{x_i, x_j} q_{ij}(x_i, x_j) \log \frac{q_{ij}(x_i, x_j)}{\left(\sum_{x'_j} q_{ij}(x_i, x'_j) \right) \left(\sum_{x'_i} q_{ij}(x'_i, x_j) \right)}$. This is used to define a free energy: $\mathcal{F}(\vec{\mathbf{q}}; \boldsymbol{\mu}_\mathbf{e}; \theta) \triangleq -\sum_i H_i(q_i) + \sum_{ij} \mu_{ij} I_{ij}(q_{ij}) - \vec{\mathbf{q}} \cdot \theta$.

In [18] Wainwright et al prove that $\mathcal{F}(\vec{\mathbf{q}}; \boldsymbol{\mu}_\mathbf{e}; \theta)$ provides an upper bound for the log partition function:

$$\log Z = \int_x exp(-\sum_i E_i(x_i) - \sum_{ij} E_{ij}(x_i, x_j)) \leq \mathcal{F}(\vec{\mathbf{q}}; \boldsymbol{\mu}_\mathbf{e}; \theta)$$

They also show that the free energy $\mathcal{F}(\vec{\mathbf{q}}; \boldsymbol{\mu}_\mathbf{e}; \theta)$ is minimized using the peseudo-marginals vector $\vec{\mathbf{b}}$ defined using the tree-rewighted messages passing output. Therefore the tighter upper bound on $\log Z$ is provided by $\vec{\mathbf{b}}$.

This result follows the line of approximations to the log partition function using free energy functions. As stated in [20], when standard belief propagation converges, the output beliefs vector is a stationary point of the bethe free energy function, and when generalized belief propagation converges, the output beliefs vector is a stationary point of the Kikuchi free energy function. However, unlike the bethe free energy and

Kikuchi approximations, the tree-rewighted free energy is *convex* with respect to the peseudo-marginals vector, and hence tree-rewighted belief propagation can not end in a local minima.

A second useful property of using the tree-rewighted upper bound as an approximation for the log partition function, is that computing the likelihood derivatives (equations 3-4) using the beliefs output of tree-rewighted massages passing, will result in *exact* derivatives for the upper bound approximation.

In this paper we used $\mathcal{F}(\vec{\mathbf{b}}; \boldsymbol{\mu}_{\mathbf{e}}; \theta)$ as an approximation for the log partition function, where $\vec{\mathbf{b}}$ is the output of tree-rewighted belief propagation. We also used the tree-rewighted beliefs $\vec{\mathbf{b}}$ in the derivatives computation (equations 3-4), as our approximation for the marginal probabilities.

References

1. A. Barbu and S.C. Zhu. Graph partition by swendsen-wang cut. In *Proceedings of the IEEE International Conference on Computer Vision*, 2003.
2. E. Borenstein, E. Sharon, and S. Ullman. Combining top-down and bottom-up segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshop on Perceptual Organization in Computer Vision*, June 2004.
3. E. Borenstein and S. Ullman. Class-specific, top-down segmentation. In *Proc. of the European Conf. on Comput. Vision*, May 2002.
4. X. He, R. Zemel, and M. Carreira-Perpinan. Multiscale conditional random fields for image labeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2004.
5. Xuming He, Richard S. Zemel, and Debajyoti Ray. Learning and incorporating top-down cues in image segmentation. In *ECCV*, 2006.
6. M. Pawan Kumar, P.H.S. Torr, and A. Zisserman. Objcut. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2004.
7. S. Kumar and M. Hebert. Discriminative random fields: A discriminative framework for contextual interaction in classification. In *Proceedings of the IEEE International Conference on Computer Vision*, 2003.
8. John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA, 2001.
9. John Lafferty, Xiaojin Zhu, and Yan Liu. Kernel conditional random fields: Representation and clique selection. In *ICML*, 2004.
10. Yann LeCun and Fu Jie Huang. Loss functions for discriminative training of energy-based models. In *Proc. of the 10-th International Workshop on Artificial Intelligence and Statistics (AIStats'05)*, 2005.
11. B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. In *Proceedings of the Workshop on Statistical Learning in Computer Vision*, Prague, Czech Republic, May 2004.
12. J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and texture analysis for image segmentation. In K.L. Boyer and S. Sarkar, editors, *Perceptual Organization for artificial vision systems*. Kluwer Academic, 2000.
13. Andrew McCallum. Efficiently inducing features of conditional random fields. In *UAI*, 2003.
14. A. Quattoni, M. Collins, and T. Darrell. Conditional random fields for object recognition. In *NIPS*, 2004.

15. X. Ren, C. Fowlkes, and J. Malik. Cue integration in figure/ground labeling. In *advances in Neural Information Processing Systems (NIPS)*, 2005.
16. E. Sharon, A. Brandt, and R. Basri. Segmentation and boundary detection using multiscale intensity measurements. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2001.
17. Z.W. Tu, X.R. Chen, A.L Yuille, and S.C. Zhu. Image parsing: segmentation, detection, and recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, 2003.
18. M. J. Wainwright, T. Jaakkola, and A. S. Willsky. Tree-reweighted belief propagation and approximate ml estimation by pseudo-moment matching. In *9th Workshop on Artificial Intelligence and Statistics*, 2003.
19. J. Winn and N. Jojic. Locus: Learning object classes with unsupervised segmentation. In *Proc. Int'l Conf. Comput. Vision*, 2005.
20. J. S. Yedidia, W.T. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51:2282–2312, 2005.
21. S.X. Yu and J. Shi. Object-specific figure-ground segregation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2003.
22. A. Yuille and P. Hallinan. Deformable templates. In *Active Vision, A. Blake and A. Yuille, Eds. MIT press*, 2002.
23. Song Chun Zhu, Zing Nian Wu, and David Mumford. Minimax entropy principle and its application to texture modeling. *Neural Computation*, 9(8):1627–1660, 1997.