Image Segmentation Using Bayesian Network and Superpixel Analysis

Mohammad Akbari Graduate School for Integrative Sciences and Engineering

Shahab Ensafi School of Computing

Jie Fu Graduate School for Integrative Sciences and Engineering

National University of Singapore

Abstract

Unsupervised image segmentation algorithms rely on a probabilistic smoothing prior to enforce local homogeneity in the segmentation results. Tree-structure prior is one such prior which allows important multi-scale spatial correlations that exist in natural images to be captured. This research presents a novel probabilistic unsupervised image segmentation framework called Irregular Tree-Structured Bayesian Networks (ITSBN) which introduces the notion of irregular tree structure. Our method, however, does not update the adaptive structure at every iteration, which drastically reduces the computation required. We derive a time-efficient exact inference algorithm based on a sum-product framework using factor graph. Furthermore, a novel methodology for the evaluation of unsupervised image segmentation is proposed. We also used deep learning approach to extract a set of features for each superpixels. By integrating non-parametric density estimation technique with the traditional precision-recall framework, the proposed method is more robust to boundary inconsistency due to human subjects.

1 Background and related works

Unsupervised image segmentation has long been an important subject of research in computer vision and image understanding. Probabilistic modeling of images provides a useful framework to conduct inference from images. There has been much interest in this area over recent years, and this has given rise to the development of a rich and varied suite of models and techniques. Numbered amongst these are wavelet models, elastic template matching, and one which has been particularly prominent is that of Markov Random Field (MRF) approaches. We will prepare an overview of some important points of these works in this section.

1.1 Fixed Architecture image models

Markov Random Field (MRF) is one of the most popular methods in image modeling. This class of model has seen widespread use in many areas, and in image segmentation these models have been used by authors such as Chellappa and Chatterie [18], Chellappa and Jain [19]. In MRF models a neighbourhood relation is firstly defined between pixels or groups of pixels in the image. A random variable designating the state is associated with each element in the image, and at a particular site the state value of this variable is not only given by the class of the pixel(s) to which it relates, but is also conditional on the states of its neighbours.

Thus the model considers the local correlations within an image, and produces a Gibbs distribution which can be explored by sampling methods [20].

However, the nature of MRF algorithms is such that they do not examine global effects directly, but only as a conspiracy of local effects, by virtue of the fact that the state of a node in the model is only influenced directly by those of its neighbors. This enables efficient parallel implementation of the algorithm but leads to a failure to capture explicitly less-localized effects such as a region of sky or a table in an image. MRFs are undirected graphs and non-hierarchical in structure which means that unlocalized or even global information cannot directly be considered. This is clearly disadvantageous in image segmentation.

Recently tree-structured belief networks (TSBNs) have been applied to image segmentation. Bouman and Shapiro [11] and Luettgen and Willsky [21] provide two such examples. These TSBNs use fixed balanced tree-structures which have the image (or an encoded representation of the image) instantiated at the leaf level of the network and an algorithm which propagates messages of belief about the status of the network to all the other nodes in the tree up to the root. The resulting equilibrium state, which occurs when all nodes have had their beliefs updated, provides the image segmentation.

1.2 Dynamic architecture image models

Models whose architecture can be dynamically altered provide a powerful means of addressing the image translation issue. The distinction between these and the fixed architecture models is that dynamic models don't merely instantiate hidden variables in a fixed structure conditionally on the supplied data, but they seek also to find relationships between substructures of these variables and are able to modify these relationships on the fly should subsequent data demand it.

Such issues in images have been considered by von der Malsburg [23] who proposed a model of deformable templates called the Dynamic Link Architecture (DLA). In this model the input image triggers feature detector cells which are then elastically matched to templates residing in cells in the layer above, by exploring dynamic links between the two layers. The templates are labeled graphs of objects expected to be found in the image, and the task is essentially one of labeled graph matching. Though dynamic in architecture this model is non-hierarchical with all object templates residing in a single layer.

Geman and Geman [20] introduce line processes as an extension of the basic MRF approach. These line processes (which also form an MRF) occupy the boundaries between pixels, and the connection of a number of these segments together produced the regions. Line processes are dynamically combined as edge elements which describe the boundaries between regions in the image. They perform this within a Bayesian framework and apply the model to an image restoration problem. This is an interesting model, but it still suffers from the disadvantages of MRF approaches in that inference is NP-hard.

A promising alternative to the above is to allow certain flexibility to the network architecture such that the sub-tree elements can be constructed in a way that their boundaries correspond directly to the natural boundaries in the image, producing unbalanced TSBNs. It is anticipated that such models would have the flexibility required to alleviate the "blockiness" experienced by balanced TSBNs and be invariant to image translation. Unbalanced tree structures have already been used by other areas of image modeling such as Meer and Connelly with an algorithmic approach. Modeling such structures within a Bayesian framework seems very attractive. Maintaining a tree structure without cross-connections would further allow the use of the attractive linear-time inference algorithms of Pearl.

Some work applying a hierarchical Bayesian model to images has already been undertaken by Utans and Gindi [25]. Their motivation was object recognition and, like von der Malsburg, sought a means of identifying particular objects in an image independent of position, scale and rotation. These may appear anywhere in the image and the task was to identify and match them against a previously constructed model of a single instance of the object. The hierarchy consists of three layers, with a single object match neuron at the top level connected to the second level object subparts whose labeling is based on the stored object structure. The lowest level neurons identify the individual components of the object in the given image, and the algorithm then tries to automatically label and connect these neurons to parents in the layer above. As in von der Malsburg [23], prior knowledge of the object is required and generalizing the approach to scenes containing multiple objects, including unknown and occluded types is a very difficult task.

2 Probabilistic modeling of images with tree

The main limitation of MRF models is that the inference is not computationally efficient. They also lack a hierarchical structure and do not provide a multi-scale interpretation of the image, which is highly desirable.

We concentrate on a newly emerging and promising image model called the Tree- Structured Belief Network (TSBN). Tree-structured belief networks provide a natural way of modeling images within a probabilistic framework. By this method a balanced tree-structured belief network is constructed with a single root node and the image is presented at the leaves. Inference can then be conducted by an efficient linear-time algorithm [10]. Fixed-structure TSBNs have been used by a number of authors as models of images such as Bouman and Shapiro [11]; Irving et al.[12]. They have an attractive multi-scale architecture, but suffer from problems due to the fixed tree structure, which can lead to very blocky segmentations.

Consider for instance the four level binary tree of Figure 1.1(a). The image applied in the example is a 1-d image of a black bar on a white background, and initially all the other nodes in the tree are uninstantiated. The structure and size of the TSBN directly determines the images it can deal with. The binary tree shown handles 1-d images, but more commonly quad-trees are used which better model the real 2-d images that are of interest.

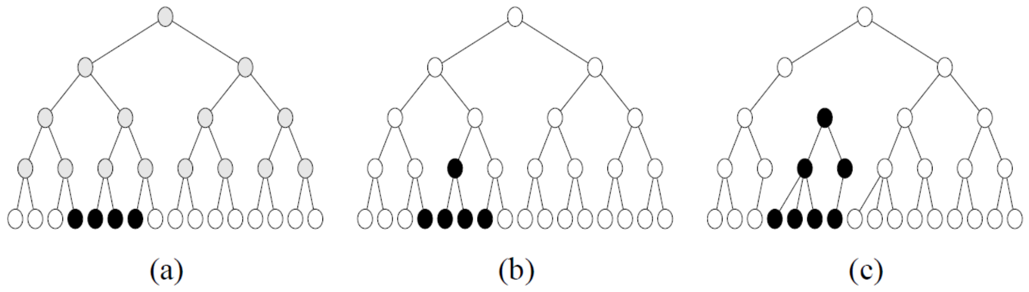

Figure 1: (a) balanced tree with image applied (b) resulting segmentation (c) an example of dynamic tree

The hierarchical structure of TSBNs naturally leads to coarser-scale representations of the image at successive levels. As well as providing a natural mechanism for all regions in the image to have some influence over each other and thus exert global consistency, there is also potential for using the segmentations given at higher levels in image coding applications.

However some problems arise when the natural boundaries in the image do not coincide with those of sub-trees in the TSBN. This effect is illustrated by Figure 1.1(b), where the black bar spans two sub-trees with roots at the third level. The resulting segmented images exhibit an undesirable blockiness as a consequence. The aim of this work is to attempt to find models which produce good representations of natural images and to use these models to improve on current image segmentation techniques. One strategy to overcome the fixed structure of TSBNs is to break away from the tree structure, and use belief networks with cross connections. However, this means losing the linear-time belief-propagation algorithms that can be used in trees (Pearl, 1988a) and using approximate inference algorithms.

TSBNs have many attractive properties and we believe that models based upon them, but having a dynamically adjustable tree structure to enable their boundaries to better reflect those of the image, provide a very promising starting point. Such models we have named dynamic trees (DTs) and one such tree produced for the toy image data is shown in the Figure 1.1(c).

Dynamic trees are a generalization of the fixed-architecture tree structured belief networks. Belief networks can be used in image analysis by grouping its nodes into visible units **X***v*, to which the data is applied, and having hidden units **X***h*, connected by some suitable configuration, to conduct a consistent inference. DTs set a prior *P(***Z***)*. Over tree structures **Z** which allows each node to choose its parent so as to find one which gives a higher posterior probability *P(***Z**,**X***h*| **X***v)* for a given image **X***v*. This effectively produces forests of TSBNs.

There are two essential components that make up tree architectures and the nodes and conditional probability tables (CPTs) in the given tree. There are a very large number of possible structures; in fact for a set of nodes created from a balanced tree with branching factor b and depth D (with the top level indexed by 1) there are $\prod_{d=2}^{D}(b^{(d-2)}+1)^{b^{d-1}}$ possible forest structures.

Our objective will be to obtain the maximum a posteriori (MAP) state from the posterior $P(Z|Xv) \propto P(Z)P(Xv|Z)$. For any Z it is possible to compute $P(Xv|Z)$ (using Pearl message passing) and P(Z). However, since the number of configurations of Z is typically very large it will usually be intractable to enumerate over them all and other approaches need to be adopted.

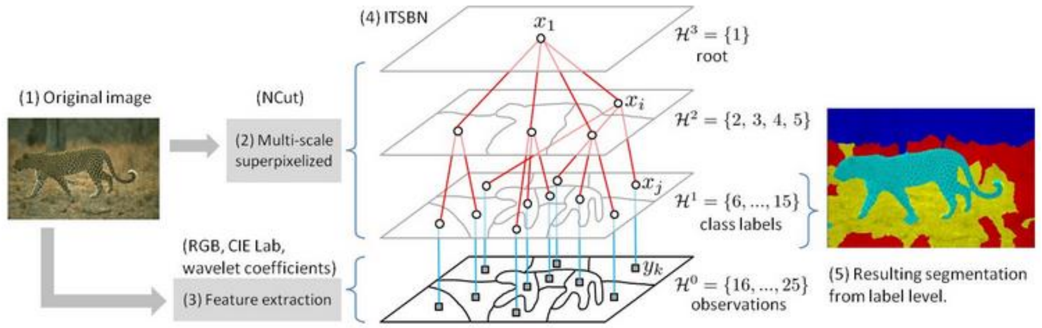
(4) ITSBN
x_1
$\mathcal{H}^3 = \{1\}$
root
(1) Original image
(NCut)
x_i
(2) Multi-scale superpixelized
$\mathcal{H}^2 = \{2, 3, 4, 5\}$
x_j
$\mathcal{H}^1 = \{6, ..., 15\}$
class labels
(RGB, CIE Lab, wavelet coefficients)
y_k
(3) Feature extraction
$\mathcal{H}^0 = \{16, ..., 25\}$
observations
(5) Resulting segmentation from label level.

Figure 2. The overview of Irregular Tree-Structured Bayesian Network (ITSBN) framework

Sampling by Markov Chain Monte Carlo (MCMC) techniques, or search using Simulated Annealing are possibilities which have been considered in [28], but the drawback is that they are slow. An alternative to sampling from the posterior $P(Z, Xh|Xv)$ is to use approximate inference. One possibility is to use a mean-field-type approximation to the posterior of the form $Q_z(Z)Q_{X_h}(X_h)$ and this is which have been considered in [14].

One another alternative is to find the best structural connectivity for Z and then using it to find the best possible labeling configuration for the tree-structure of the image segments. In this way, we will have an approximation of the best structure connectivity between nodes of the tree. Suppose that the structure connectivity is characterized by an $N \times N$ matrix adjacency matrix Z, where z_{ij} takes a values 1 if node j is the parent of node i. there is only one constraint which each node could be connected to a parent node in the directly adjacent layer. Therefore, a realization of structure matrix Z can have at most one entry equal to 1 in each row.

Suppose we have an image which is segmented by our method, see figure 2. Each node is a discrete random variable x_i of an image section i. This random variable takes one of C possible labels/classes. The probability which defines nodes j and its parents i has been labeled v and u respectively is $\theta_{ij}^{uv} = p(x_j = v|x_i = u)$

We can rewrite this equation by defining two indicator variables x_j, x_i which pick out the correct probabilities.

$$p(x_j = v|x_i = u) = \prod_{v=1}^{C}\prod_{u=1}^{C}[\theta_{ij}^{uv}]^{x_j^v x_i^u}$$

Note that this framework is unsupervised, hence the probabilistic model of a class c is not provided a priori, and the number of labels allowed for each input image must be defined before executing the algorithm. Estimating the appropriate number of classes for each image (the model selection problem) is explained in more detail in Section 3.

We introduce observed variables represented by shaded square-shaped nodes in the structure as illustrated in Fig. 1. Each observed random variable $y_e \in R_d$ of an image site $e \in V$ represents the relevant image features such as color or texture which take on continuous values. Extensive details on our choice of features will be discussed in the experimental results section. We model the feature vector y_e using a multivariate Gaussian distribution given as:

$$p(y_e|x_i) = \prod_{c=1}^{C} \mathcal{N}(y_e|\mu_c, \Lambda_c^{-1})^{x_{ic}}$$

Where $\mathcal{N}(y_e|\mu_c, \Lambda_c^{-1})^{x_{ic}}$ is the multivariate Gaussian distribution and μ_c, Λ_c^{-1} are mean and covariance matrices for class c respectively. Using the notation described above, we can now write the hidden labels collectively as $X = \{x_i\}_{i\in H}$ which H denotes the first hidden layers and the observed image features as $Y = \{y_e\}_{e\epsilon V}$. The log-likelihood of the complete data can be expressed as:

$$\log p(X,Y|Z,\theta) = \sum_{e\epsilon V}\sum_{i\epsilon H^1} z_{ei} \prod_{c} x_{ic} log\ \mathcal{N}(y_e|\mu_c, \Lambda_c^{-1})^{x_{ic}}$$

$$+ \sum_{l}(\sum_{j\epsilon H^1}\sum_{i\epsilon H} z_{ji} \sum_{v=1}^{c}\sum_{u=1}^{c} x_j^v x_i^u log\theta_{ij}^{uv})$$

Note that the connectivity structure matrix Z is assumed known from each input image, hence is excluded from the parameter set. In the next section, we derive an efficient inference and parameter estimation algorithm for ITSBN.

3 Inference and parameter Learning

In this section, we present a maximum likelihood estimation algorithm for ITSBN model parameter θ. For models with hidden variables, Expectation-Maximization (EM) [13] is a principled framework that alternates between inferring the posterior over hidden variables in the E-step, and maximizing the expected value of the complete data likelihood with respect to the model parameter in the M-step. More specifically, we infer the posterior probability over the hidden labels in the E-step and compute relevant sufficient statistics needed in the M-step for maximizing θ. Since exact inference on a tree structure graph is tractable, the objective function can be written as:

$$\mathcal{F}(\theta, \theta^{t-1}) \equiv \langle \log p(X,Y|Z,\theta)\rangle_{p(X|Y,Z,\theta^{t-1})}$$

Where $\langle f(x)\rangle_{q(x)}$ denotes the expectation of function f(x) respect to the distribution q(x). In M-step, by maximizing $\mathcal{F}$ with respect to θ, we derive close-form update equations for μ_c, Λ_c^{-1} and θ_l^{uv} as follows:

$$\mu_c = \frac{\sum_{e\epsilon V}\sum_{i\epsilon H^1} z_{ei}\langle x_i^c\rangle_{p(X|Y,Z,\theta^{t-1})} y_e}{\sum_{e\epsilon V}\sum_{i\epsilon H^1} z_{ei}\langle x_i^c\rangle_{p(X|Y,Z,\theta^{t-1})}}$$

$$\Lambda_c^{-1} = \frac{\sum_{e\epsilon V}\sum_{i\epsilon H^1} z_{ei}\langle x_i^c\rangle_{p(X|Y,Z,\theta^{t-1})}(y_e-\mu_c)\,(y_e-\mu_c)^T}{\sum_{e\epsilon V}\sum_{i\epsilon H^1} z_{ei}\langle x_i^c\rangle_{p(X|Y,Z,\theta^{t-1})}}$$

$$\theta_l^{uv} = \frac{\hat{\theta}_l^{uv}}{\sum_{\acute{u}} \hat{\theta}_l^{\acute{u}v}}$$

Where $\hat{\theta}_l^{uv} = \sum_{j\epsilon H^l}\sum_{i\epsilon H} z_{ji}\,\langle x_i^u x_j^v\rangle_{p(X|Y,Z,\theta^{t-1})}$ denotes the unnormalized class-transition CPT. To compactly explain our inference step, we express ITSBN in terms of a factor graph [4] using 2 types of nodes: 1) a variable node for each random variable x_i and 2) a factor node for each local function, namely CPT in this case. In the operation of sum-product algorithm, we compute a message from variable node x to factor node $f: m_{x\to f}(x)$, and message

from factor node f to variable node $f: m_{f\to x}(x)$, which can be expressed as:

$$m_{x\to f}(x) = \prod_{h\in n(x)\backslash\{f\}} m_{h\to f}(x)$$

$$m_{f\to x}(x) = \sum_{n(f)\backslash x} (f(n(x)) \prod_{y\in n(x)\backslash\{x\}} m_{h\to f}(x))$$

where $n(x)$ denotes the set of neighbors of a given node x, and $n(x)\{f\}$ denotes the remaining after f is removed from the set $n(x)$.

4 Building the tree

In this section, we discuss the methodology applied to each input image in order to create an irregular tree structure, where each level in the hierarchy can be regarded as a scale of visualization (coarse-to-fine representation). The details of over-segmentation algorithm and connecting superpixels across the adjacent level are discussed as followed.

4.1 Superpixel segmentation

There are many approaches to generate superpixels, each with its own advantages and drawbacks that may be better suited to a particular application. For example, if adherence to image boundaries is of paramount importance, the graph-based method of [4] may be an ideal choice. However, if superpixels are to be used to build a graph, a method that produces a more regular lattice, such as [1], is probably a better choice.

In this paper, we use the implementation in [15] due to its computational efficiency and homogeneity in the resulting superpixel output e.g. pixels with similar texture, color, and residing within the same object boundary are grouped into the same superpixel.

In our experiment, the number of hierarchical level L is fixed to 6 for every image, and the number of superpixels from level $\mathrm{l} = \mathrm{L}-2$ to leaf $\mathrm{l} = 0$ are approximately 5, 20, 50, 200 and 200 respectively. As mentioned earlier the root level contains only 1 node while the leaf level has the same number of nodes as level $\mathrm{l} = 1$. The same setting is applied to every image in the dataset.

4.2 Structure of tree

We describe a method to build an irregular tree structure from each input image and its corresponding multi-scale superpixels derived from the output of [15]. Two constraints are required in order to build such a tree in our experiment. First, each observed node $\mathrm{y_e}$ in the leaf level can connect to its parent in level $l = 1$ in a 1-to-1 transparent manner as shown in Fig. 1 since both levels are indeed derived from the same over-segmenting process. Second, each child node in level $l \in \{1 \dots L-2\}$ can only be connected to a parent in level $l + 1$. This regularization enables smoothness of the information transfer across the scale of image. The constraint is formulated formally as follows. For all $j \in H^l$, $i \in H^{l+1}$ and $l \in \{1 \dots L-2\}$ the connectivity $z_{ji} = 1$ when $i = i^*$ and otherwise $z_{ji} = 0$; when $i^* = argmax|S_i \cap S_j|$.
Where $|S_i \cap S_j|$ means the number of overlapped pixels when intersecting superpixel Si and Sj together. When regarding Sj as an arbitrary set, this idea can be extended to build a datadriven tree structure in other settings beyond superpixels, for instance block pixels, voxels, abstract image regions. Furthermore, the proposed method still works in the case where S_j is not a strict subset of S_{i^*} which makes the construction process quite robust and general.

5 Extracting features

In the bottom of the tree, we have a set of visible nodes which corresponds to image segments. We will use feature learning to extract image features from segments. In this section we discuss about feature learning and its advantage to our work.

Images are highly variable (because of viewpoint changes, shape variation, etc.) and high-dimensional (often hundreds of thousands of pixels). It is difficult for computer vision algorithms to run on raw image data and to generalize well from training data to unseen data. Therefore, it is desirable to extract features to make computer vision tasks easier to solve and finding “good” feature representations is vital for this kind of high-level computer vision task. Features are defined here as attributes that can be extracted from the input data, and these feature representations could then facilitate subsequent high-level computer vision tasks.

Both color and texture features are extracted from an input image. Color feature is obtained by averaging RGB and CIE Lab color space within a superpixel. Texture feature is acquired from applying 3-level complex wavelet transform (CWT) to the input image, then averaging the magnitude of wavelet coefficients within a superpixel. Totally, we have 15-dimensional feature vector from each superpixel; 6 from color and 9 from texture.

State-of-the-art feature extractors usually consist of a filter bank, a non-linear transformation, and some kind of feature pooling method [15], which collect nearby values through an average, maximum, or histogramming computation. The filter bank could be oriented edge detection filters such as Scale Invariant Feature Transform (SIFT) [16]. Hand-crafted feature methods, such as SIFT, are still the most common ones. Typically, these hand-crafted features use low-level concepts (e.g. edges) rather than higher-level concepts such as parallel lines or basic geometric elements [17].

It has been shown that dictionary learning can be used in computer vision to learn features for modeling the local appearance of natural images, leading to state-of-the-art performance [17]. Thus, learning hierarchical feature representations automatically is a more flexible and more general strategy. In this project, we adapt Deconvolutional Networks [17] to form the hierarchical sparse convolutional feature representations in a purely unsupervised fashion. However, the drawback of Deconvolutional Networks architecture is its fix geometry. As the features are aggregated according to a predefined pattern, the higher level features represent data with poor spatial accuracy [29]. Therefore, we only use a 2-layer architecture in this segmentation project.

6 Evaluation and experiments

While evaluation methodology for supervised image segmentation result is obvious, there is no clear standard to evaluate unsupervised image segmentation. We adopt the methodology mentioned in [26] which regards the border between each pair of adjacent resulting segmentation regions as the object boundary which can be evaluated using boundary-based precision-recall (PR) framework of [27]. In the framework, each pixel in the resulting boundary image can take a real value ranging from 0 to 1; 1 when the pixel is believed to be a boundary and 0 when is not. However, the framework is still vulnerable to boundary misalignment occurred naturally between the boundaries produced by segmentation algorithms and ones produced by several human subjects.

We report our results from experiment on segmentation and matching of boundary images from Berkeley Segmentation Data Set and Benchmarks 500 (BSDS500), reported in [26]. The dataset contains 300 training and 200 testing images, each of which has ground-truth boundary images done manually by five different human subjects on average. Since this is unsupervised framework, we only run our algorithm on the 200 testing images directly without training the model beforehand, and thus the 300 training images are discarded.

We used two kinds of feature vectors in this research. In the original paper, authors have used a simple feature vectors which extracted from color and texture information. We also used feature learning to use a feature vector for each pixel and then each superpixels. It shows promising result and has better performance in terms of precision and accuracy.

In the first experiments, both color and texture features are extracted from an input image. Color feature is obtained by averaging RGB and CIE Lab color space within a superpixel. Texture feature is acquired from applying 3-level complex wavelet transform (CWT) to the input image, then averaging the magnitude of wavelet coefficients within a superpixel. Totally, we have 15-dimensional feature vector from each superpixel; 6 from color and 9 from texture. Note that we have not used significant number of features in our experiment compare to others in the literature. That is because our focus is on the performance of the ITSBN alone, not on the image feature.

We extract a 45-dimensional learned feature vectors using 2-layer Deconvolutional Networks [17]. Features have been extracted from a 9*9 kernel around each pixel. As we use supetpixels in the leaves of the tree, we construct a feature vector for each superpixels by averaging the feature vectors of pixels belongs to. So for each superpixels we have a 45-dimension feature vector.

At the end, the values of precision, recall, F-measure and computational run-time are averaged across all the test images in the dataset as shown in Table 1. The values summarizes the performance of both methods applied to the dataset. At the same parameter setting, the results show that ITSBN slightly outperforms supGMM. The greater precision value indicates less noisy image segmentation results in ITSBN than supGMM. This makes a lot of sense because ITSBN has tree-structured prior which is equivalent to adding the label smoothness regularization to the maximization of the log-likelihood. However, supGMM seems to slightly does a better job on recalling the boundary pixel.

Table 1. The across-dataset average of precision, recall and F-measure of supGMM, ITSBN and FL

Method	**Precision**	**Recall**	**F-measure**	**Run-time**
supGMM	0.2597	0.5711	0.3470	30sec/image
ITSBN	0.2636	0.5697	0.3501	2min/image
FL+ITSBN	0.2154	0.4127	0.2831	2min/image+

7 Conclusion

In this project, we have described and implemented an efficient unsupervised image segmentation algorithm using probabilistic tree-structure Bayesian network. We implemented Irregular Tree Structure Bayesian Networks (ITSBNs). The tree structure of ITSBN can be tailored to fit natural boundaries in each input image, thus eliminates blocky segmentation results. Based on the factor graph representation, we derived a sum-product algorithm for the inference step. In order to avoid adapting the tree structure at each iteration, we presented a fast and simple method to build an irregular tree from a set of superpixels in different scales extracted from each input image. This flexibility indeed is very much desired as a great number of such algorithms have been made publicly available, each with different properties. We also used feature learning method to extract features from each superpixels. By integrating non-parametric density estimation technique with the traditional precision-recall framework, the proposed method is more robust to boundary inconsistency due to human subjects. We experimentally show the improvement of ITSBN over the baseline method which motivates us to further investigate the model of similar type. Although, feature learning method did not extract effective features for segmentation, we can improve it by combining these learned feature and simple features.

Acknowledgments

We thank Li Cheng, Kittipat Kampa and Matthew Zeiler for insightful discussions.

References

[1] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[2] T. Cour, F. Benezit, and J. Shi. Spectral segmentation with multiscale graph decomposition. In *CVPR*, 2005.

[3] A. Levinshtein, A. Stere, K. Kutulakos, D. Fleet, S. Dickinson, and K. Siddiqi. Turbopixels: Fast superpixels using geometric flows. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009.

[4] Pedro Felzenszwalb and Daniel Huttenlocher. Efficient graph-based image segmentation. In *CVPR*, 2004.

[5] Alastair Moore, Simon Prince, Jonathan Warrell, Umar Mohammed, and Graham Jones. Superpixel Lattices. In *CVPR*, 2008

[6] O. Veksler, Y. Boykov, and P. Mehrani. Superpixels and supervoxels in an energy optimization framework. In *ECCV*, 2010

[7] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, May 2002.

[8] A. Vedaldi and S. Soatto. Quick shift and kernel methods for mode seeking. In *ECCV*, 2008.

[9] Luc Vincent and Pierre Soille. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. In *IEEE Transactions on Pattern Analalysis and Machine Intelligence*, 13(6):583–598, 1991.

[10] J. Pearl. *Probabilistic reasoning in Intelligent Systems*. Morgan Kaufman Publishers, 1998

[11] C. A. Bouman and M. Shapiro. A Multiscale Random Field Model for Bayesian Image Segmentation. In *IEEE Transaction on Image Processing*, 3(2), 1994.

[12] W. W. Irving, P. W. Fieguth and A. S. Willsky, An Orverlapping Tree Approach On Multiscale Stochastic Modeling and Estimation. In *IEEE Transaction on Image Processing*, 6(11), 1997.

[13] G. Mori, Guiding Model Search Using Segmentation, In *ICCV*, 2005.

[14] N. Adams, C.K.I. Williams, Dynamic Trees for Image modeling, In *Image and Vision Computing*, 2002.

[15] Kavukcuoglu, K, Ranzato, M.A, LeCun, Y. What is the best multi-stage architecture for object recognition?. In *CVPR*, 2009.

[16] David G Lowe, Distinctive image features from scale-invariant keypoints, In *Internatonal Journal of Computer Visopm*, 60(1), 2004

[17] M. Zeiler, D. Krishnan, G. Taylor, R. Fergus. Deconvolutional Networks. In *CVPR*, 2010.

[18] Chellappa, R. and Chatterie, S. Classification of Textures using Guassian Markov Random Fields. In *IEEE Trans. Accoust., Speech and Signal Processing*, volume 33, pages 959–963, 1985.

[19] Chellappa, R. and Jain, A. *Markov Random Fields - Theory and Applications*. Academic Press Ltd, London, UK, 1993.

[20] Geman, S. and Geman, D.. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 6, no. 6, pages 721–741, 1984.

[21] Luettgen, M. R. and Willsky, A. S. Likelihood Calculation for a Class of Multiscale Stochastic Models, with Application to Texture Discrimination. In *IEEE Transactions on Image Processing*, 4(2), 194–207, 1995.

[22] Dayan, P., Hinton, G. E., Neal, R. M., and Zemel, R. S. The Helmholtz Machine. In *Neural Computation*, 7(5), 1995.

[23] von der Malsburg, C. Pattern Recognition by Labelled Graph Matching. In *Neural Networks*, volume 1, pages 141–148, 1988.

[24] Montanvert, A., Meer, P., and Rosenfeld, A. Hierarchical Image Analysis Using Irregular Tessellations. In *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(4), 307–316, 1991.

[25] Utans, J. and Gindi, G. Improving Convergence in Hierarchical Matching Networks for Object Recognition. In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, In *NIPS*, 1993.

[26] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik, Contour detection and hierarchical image segmentation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, pp. 898–916, 2011.

[27] D.R. Martin, C.C. Fowlkes, and J. Malik, Learning to detect natural image boundaries using local brightness, color, and texture cues. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, pp. 530 –549, 2004.

[28] R. Jenssen, D. Erdogmus, K. Hild, J. Principe, and T. Eltoft, Optimizing the Cauchy-Schwarz PDF distance for information theoretic, non-parametric clustering. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*. Springer, pp. 34–45, 2005.

[29] Leon Bottou. From Machine learning to Machine Reasoning, *arXiv preprint*, 2011