Graph-Shifts: Natural Image Labeling by Dynamic Hierarchical Computing

Jason J. Corso
Computer Science and Engineering
SUNY at Buffalo
jcorso@cse.buffalo.edu

Alan Yuille
Statistics
UCLA
yuille@stat.ucla.edu

Zhuowen Tu
Neurology
UCLA
zhuowen.tu@loni.ucla.edu

Abstract

In this paper, we present a new approach for image labeling based on the recently introduced graph-shifts algorithm. Graph-shifts is an energy minimization algorithm that does labeling by dynamically manipulating, or shifting, the parent-child relationships in a hierarchical decomposition of the image. Each shift optimally reduces the energy by indirectly causing a change to the labeling; graph-shifts is able to rapidly compute and select this optimal shift at every iteration. There are no constraints on the terms of the (pairwise) energy function. The algorithm was originally presented in the context of medical image labeling using conditional random field models. In this paper, we consider the algorithm in the context of both low- and high-level natural image labeling. We show that for examples in both classes of problems, graph-shifts does labeling both accurately and rapidly. For low-level vision, we explore image restoration, and for high-level vision, we make use of a hybrid discriminative-generative model to segment and label images into semantically meaningful regions (e.g., trees, building, etc.). For both problems, we obtain comparable or superior results to the state-of-the-art computed in just a few seconds per image.

1. Introduction

Image labeling remains a core task in both low- and high-level vision problems. The energy minimization formulation is a standard image labeling methodology, typically assuming only non-zero unary and binary potentials. Even under these assumptions, the configuration space is combinatorial in the labels and the energy landscape has many local minima. The recent comparative survey [18] (of low-level problems) demonstrates the progress the community has made on these problems but also the need for more powerful and rapid methods.

To that end, we investigate the recently proposed graph-shifts [5] energy minimization algorithm in the context of both low- and high-level labeling problems. The algorithm does energy minimization by iteratively manipulating a hierarchical decomposition of the image defined as a multi-level graph. The hierarchy is adaptive in the sense that the graph and neighborhood structure is data-dependent in contrast to conventional pyramidal schemes [1] in which the hierarchical neighborhood structure is fixed. A big advantage of graph-shifts is that each iteration can exploit this adaptive hierarchical structure and cause a large change in the labeling, thereby giving fast convergence while avoiding local minima in the energy function.

We originally proposed graph-shifts in the medical imaging literature to segment and label 8 sub-cortical brain structures [5]. By comparison, the number of labels in natural image problems can be much higher (e.g., 256), and the variance of the region classes is, by nature, much greater. In this paper, we explore the capability of graph-shifts to handle such higher-dimensional, more complicated energy spaces for labeling problems in low- and high-level vision problems. In the remainder of the introduction, we give motivation and context for the two labeling problems. Then, in section 2, we describe the graph-shifts algorithm. In sections 3 and 4 we present low- and high-level labeling problems and give experimental results and comparisons. We conclude in section 5.

1.1. Low-Level Labeling

In low-level labeling problems, the typical energy to use is that of a pairwise Markov random field (MRF). Many new algorithms have been proposed for solving the associated energy minimization problems. For example, graph cuts [3] is one such algorithm that guarantees achieving a *strong* local minimum for some two-class energy functions. However, the restrictions on the energy function and the extension to m-class problems limit the theoretical guarantees. Max product belief propagation [9] computes local maxima of the posterior, but it is not guaranteed to converge for the loopy graphs present in low-level vision. Efficient belief propagation [7] can lead to running times in the order of seconds. However, despite its high-regard and widespread use, it performed poorly in the recent benchmark study [18].

Figure 1. Some example images and manual labels in each category from the Lotushill dataset [23]. A label legend is provided underneath.

1.2. High-Level Labeling

Natural scene understanding, segmenting and labeling image regions with semantically meaningful labels (e.g., trees, cars, etc.), has increasingly attracted attention since it is a key aspect in image understanding and image search. In general, region labeling is a very hard problem due to the large variation of natural images ranging from indoor to outdoor, small to large scale, and rural to city scenes. Moreover, some type of objects, e.g. buildings, may have very different designs and appear very differently under different viewing directions and scales.

Existing approaches [4, 11, 14] in scene understanding often rely on texture clustering in which content (object) segmentation is not explicitly tackled. Recent examples that use rectilinear patches include Lu et al. [15], who use a mixture texture model, and Fei-fei et al. [6] who adopt an unsupervised hierarchical model from category to patches. Some common drawbacks of these approaches: (1) spatial relationships of image patches and regions are not characterized, which is one of the key aspects in scene understanding; (2) texture properties of image patches are not fully explored; (3) no image segmentation is performed and their approaches only target scene categorization. Barnard et al. [2] did indeed attempt a segmentation before classification. But, segmentation and labeling are chicken-and-egg; they use a generic low-level segmentation algorithm, which is unlikely to give good segments without incorporating high-level knowledge like shape and context. Shotton et al. [17] noticed similar drawbacks of these methods and propose a boosted model of joint appearance, shape and context on a conditional random field [13].

We propose a supervised approach for combined image segmentation and region labeling. We use a hybrid discriminative-generative modeling scheme. The discriminative term is modeled by an extension of the probabilistic boosting tree algorithm [20] that does multi-class representation in a tree structure. It can better handle the variability in natural image patches. The generative terms captures the local context of the image regions. We consider a subset of the recently published Lotushill dataset [23] that has 10 generic objects types (regions) targeted in the scene and 400 images. Examples from the dataset are given in figure 1. We also include results on the MSRC 21-class dataset from Shotton et al. [17] and demonstrate superior labeling accuracy.

2. Hierarchical Computing with Graph-Shifts

In this section, we discuss the graph-shifts algorithm in the context of labeling natural images in both low- and high-level vision problems. We first present the class of energy functions to be considered, and define those energies in the hierarchy. Second, we discuss the mechanics of the shifting procedure that reduces the energy defined on the hierarchy. Third, we present the complete algorithm.

2.1. The Energy Model

For an input image $\mathbf{I}$ defined on the pixel lattice D, the task is to assign each pixel $\mu \in D$ to one of a fixed set of K models $m_\mu \in \{1, ..., K\}$. We want the labeling to minimize an energy function criterion:

$$E[\{m_\omega : \omega \in D\}] = \sum_{\nu \in D} E_1(\mathbf{I}(S[\nu]), m_\nu) \; + \; \frac{1}{2} \sum_{\substack{\nu \in D, \mu \in D: \\ N(\nu,\mu)=1}} E_2(\mathbf{I}(\nu), \mathbf{I}(\mu), m_\nu, m_\mu) \; . \tag{1}$$

The notation $S[\nu]$ means the local subimage centered at pixel ν, and $N(\nu, \mu) = 1$ indicates ν and μ are neighbors on the lattice. This energy falls in the broad class of Markov random fields and, depending on the assumptions could be considered a conditional random field [13] or a hybrid discriminative-generative model as described in section 4. In low-level vision problems, these labels are physically meaningful, e.g., they could be direct pixel intensities in image restoration. But, in high-level problems, these labels are semantically meaningful, e.g., label 1 means sky, 2 water, etc. The actual definition of the unary and binary terms, $E_1(\cdot)$ and $E_2(\cdot)$ resp., are specific to the problems and defined when necessary in sections 3 and 4.

2.2. Initializing The Hierarchical Graph

We define a graph G to be a set of nodes $\mu \in \mathcal{U}$ and a set of edges (the overloaded notation μ is meaningful—pixels are nodes). The graph is hierarchical and composed of multiple layers. The nodes at the lowest layer are the elements of the pixel lattice D and the edges are defined to link neighbors on the lattice. The higher layers are computed recursively by adaptively coarsening the image. Our implementation uses the coarsening method defined in [5]. The basic idea is that edges in the graph are randomly turned on or off

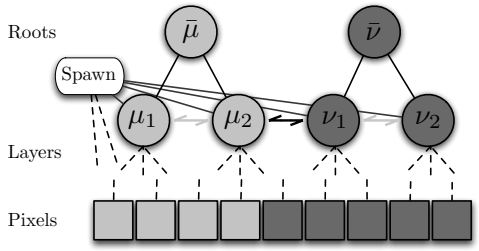

Figure 2. Example of graph structure with the spawn node.

based on the local affinity. The *on* edges induce a connected components clustering, and each component defines a new node in the next coarse layer in the hierarchy. Thus, nodes at coarser layers in the hierarchy represent (roughly) homogeneous regions in the images. The procedure is adaptive and the resulting hierarchy is data-dependent.

Two nodes at a coarse layer are joined by an edge if any of their children are joined by an edge. The predicate $N(\mu,\nu)=1$ indicate that nodes μ,ν on the same layer are neighbors, with $N(\mu,\nu)=0$ otherwise. The nodes are constrained to have a single parent, $A(\mu)$, (except for the nodes at the top layer) and every node has at least one child, $C(\mu)$ (except for nodes at the bottom layer). At the top of the hierarchy (figure 2), we define a special *root* layer of nodes that are each tied to a specific model. We write m_k to denote the model variable associated with a root node. Each node is assigned a label that is constrained to be the label of its parent, $m_\mu = m_{A(\mu)}$; call this the *parent-label constraint*. Since all non-root nodes can trace their ancestry back to a single root node, an instance of the graph G is equivalent to a labeled segmentation $\{m_\mu : \mu \in D\}$ of the image. Finally, a *spawn node* is added, which is the neighbor of all nodes except the root layer nodes (it is further discussed in section 2.4).

2.3. Recursive Energy Definition

We now recursively assign an energy to all nodes, and neighboring node pairs in the hierarchy. This enables us to rapidly compute the energy of a node at any layer in the hierarchy. The unary term for assigning a model m_μ to a node μ is defined recursively by:

$$E_1(\mu, m_\mu) = \begin{cases} E_1\left(\mathbf{I}(S[\mu]), m_\mu\right) & \text{if } C(\mu)=\emptyset \ , \\ \sum_{\nu\in C(\mu)} E_1(\nu, m_\mu) & \text{otherwise} \ . \end{cases} \quad (2)$$

The pairwise energy term E_2 between nodes μ_1 and μ_2, with models m_{μ_1} and m_{μ_2} is defined recursively by:

$$E_2(\mu_1,\mu_2,m_{\mu_1},m_{\mu_2}) = \quad (3)$$
$$\begin{cases} E_2(\mathbf{I}(\mu_1),\mathbf{I}(\mu_2),m_{\mu_1},m_{\mu_2}) & \text{if } C(\mu_1)=\emptyset \\ & \text{and } C(\mu_2)=\emptyset \\ \sum_{\nu_1\in C(\mu_1),\nu_2\in C(\mu_2):N(\nu_1,\nu_2)=1} E_2(\nu_1,\nu_2,m_{\mu_1},m_{\mu_2}) & \text{otherwise} \ . \end{cases} ,$$

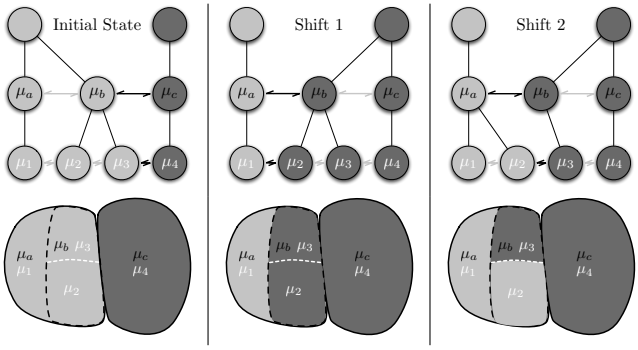

Figure 3. Intuitive split/merge shift example with two classes, light and dark gray. The top shows a graph-hierarchy, and bottom shows a possible corresponding image. White annotations depict the finer level and black depict the coarser level.

2.4. Graph-Shifts Mechanics

The graph-shifts algorithm minimizes the energy by dynamically transforming the graph hierarchy. Two transformations, or *shifts*, on the graph are defined: (1) split/merge and (2) spawn. A shift is represented as a dynamic annotation to the edges in the graph layers, i.e., a shift is a local transform between two neighboring nodes. Each shift locally restructures the graph and causes a relabeling of a pixel subset and, hence, a change in the total energy.

The split/merge shift corresponds to a node μ changing its parent to the parent $A(\nu)$ of a neighboring node ν. Figure 3 shows an intuitive example split/merge shift on a toy graph. The split/merge shift is constrained to change the labeling based on the current labels in the local neighborhood of each node. The second type of shift, however, permits a node to switch to any other label. During graph creation, a *spawn node* is defined to which all nodes in the graph are connected (see figure 2). This spawn node can take the label of any model in the set, and when evaluating the shift for a node, all possible models are evaluated. A node μ taking a spawn shift causes a new sub-graph to be dynamically created. The new sub-graph is a chain of nodes from μ to the top of the hierarchy (see figure 4).

With these two shifts, the graph-shifts algorithm is complete in the sense that any labeling can be reached from any initial labeling. The split/merge shift provides both rapid cluster relabeling and local boundary refinements. The spawn shift is a *birth process* giving a jump in energy space; the corresponding *death process* is captured by the split/merge shift (when a node μ shifts to its neighbor and leaves its current parent childless).

2.5. Graph-Shifts Algorithm

Each change of a node model will result in a change of energy (because of the parent-label constraint). We need to efficiently compute the possible shifts (change of energy)

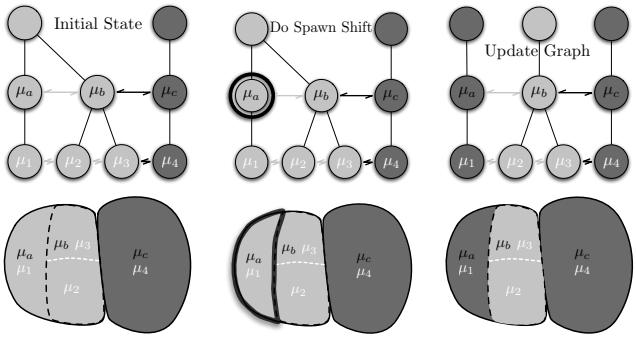

Figure 4. An example of the spawn shift being selected (middle panel, double-circle) and the hierarchy updated with the new root-level node (right panel). The top shows a graph-hierarchy, and bottom shows a possible corresponding image.

for all nodes. Fortunately, the change, or *shift-gradient* can be computed efficiently using the recursive formulae given above in equations (2) and (3). For node μ shifting from model m_μ to model $\hat{m}_\mu$, the shift-gradient is

$$\Delta E(m_\mu \rightarrow \hat{m}_\mu) = E_1(\mu, \hat{m}_\mu) - E_1(\mu, m_\mu) \; + \sum_{\eta:N(\mu,\eta)=1} [E_2(\mu, \eta, \hat{m}_\mu, m_\eta) - E_2(\mu, \eta, m_\mu, m_\eta)] \quad . \quad (4)$$

We maintain an exhaustive list of both the split/merge and the spawn shifts; given the additional spawn node, both types of shifts are simply annotations on edges in the graph. The cost of computing the shift gradient is equivalent for both shift types, but performing a spawn shift, while still logarithmic in order, has a higher computational cost in creating the new sub-graph. When computing a potential graph-shift, we first evaluate the shift-gradient for all of the node's neighbors. Next, we evaluate the shift-gradient to the spawn node, only considering those models for which there was no neighbor. We keep only those shifts that have negative gradients in the list (the single best potential shift is stored per node). The size of this list is generally small, very few neighbors in the graph have different models and a spawn shift more often increases the energy than decreases (due to the additional binary energy cost). Empirically, this is about 2% of all possible shifts in the graph.

Graph-shifts proceeds by selecting the steepest shift-gradient in the list and makes the corresponding shift in the hierarchy (pseudo-code is given in figure 5 and some samples from the process in figure 6). This changes the labels in the part of the hierarchy where the shift occurs, but leaves the remainder of the hierarchy unchanged. If the shift is a spawn, then a new sub-graph is dynamically generated. The algorithm recomputes the shift-gradients in the changed part of the hierarchy and updates the weight list. We repeat the process until convergence, when the shift-gradient list is empty (i.e. all shift-gradients in the graph

GRAPH-SHIFTS MINIMIZATION

Input: Image $\mathbf{I}$ on lattice D.
Output: Label Image $\mathbf{L}$ on lattice D.

```
0  Initialize graph hierarchy.
1  Compute exhaustive set of potential shifts S.
2  while S is not empty
3     s ← the shift in S that best reduces the energy.
4     Apply shift s to the graph.
5     if (s is a spawn)
6        Build new subgraph and create model.
7     Recompute affected shifts and update S.
8  Compute label image L from final hierarchy.
```

Figure 5. Graph-shifts pseudo-code.

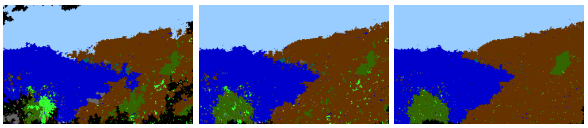

Figure 6. An example of the graph-shifts algorithm doing energy minimization. The row shows initialization, after 5, and 500 shifts. Images, labels and final result given in figure 10.

are positive or zero). The algorithm tends to initially prefer shifts at coarse levels of the hierarchy since those typically alter the labels of many nodes on the lattice and cause large changes in energy. As the algorithm proceeds, it tends to select shifts at finer levels, but this trend is not monotonic.

3. Low-Level Vision: Restoration

Image restoration is the problem of removing the noise and other artifacts of an acquired image to restore it back to its original, or ideal state. The label set comprises the 256 gray-levels, $m_\mu \in \{1, 2, \ldots, 255\}$. We work with the well-known penguin image; its ideal image is given on the right. When running graph-shifts, we only use the split/merge shift; since there are so many labels, it is assumed there is no need for the extra ability to spawn for lack of a good neighbor.

In all of the results, we use a truncated quadratic on the unary energy term, which is conditioned independently at each pixel in the conventional MAP-MRF sense:

$$E_1(\mathbf{I}(\mu), m_\mu) = \min(\alpha_1 ||\mathbf{I}(\mu) - m_\mu)||^2, \alpha_2) \quad . \quad (5)$$

We use a truncated linear binary energy. It is defined on two labels and is fixed by two parameters, β_1, β_2:

$$E_2(m_{\mu_1}, m_{\mu_2}) = \min(\beta_1 ||m_{\mu_1} - m_{\mu_2}||, \beta_2) \quad . \quad (6)$$

where we've dropped the image terms from the binary potential $\mathbf{I}(\cdot)$ since they play no role. These are among many plausible energies to use for restoration, but we choose them because they seem reasonable and facilitate direct comparison with existing works via the MRF benchmark [18]

	Time (ms)		SSD Error ($\times 10^3$)	
Variance →	10	20	10	20
Input Images			1888	7319
EBP [7]	4724	4399	4358	10934
BP [19]	634500	635151	1705	7211
ICM [18]	2380	2380	1674	7154
α-Expansion [3]	11030	11420	1669	7150
TRW-S [12]	466970	467310	1665	7149
Graph-Shifts	7489	6047	1584	5557

Table 1. Quantitative comparison of restoration speed and SSD error. Graph-shifts (a Java code) scores among the faster algorithms (all C++), and has the lowest overall SSD error.

and efficient belief propagation (EBP) code [7][1] We use the following parameters on the potentials $\alpha_1 = \beta_1 = 1$, $\alpha_2 = 100$ and $\beta_2 = 20$, and the unary and binary energies are equally weighted. We set our performance criterion to be the sum of squared differences error between the original (ideal) image and the restored image that is outputted by each algorithm. Note, however, that the energy minimization algorithms are not directly minimizing the sum-of-squared differences error function. The MRF benchmark used [18] the final energy to compare the algorithms, but did mention that the energy may not be the best metric (i.e., we may have the wrong model). Since the problem is restoration, we feel the SSD error comparison is a more natural measure and use it.

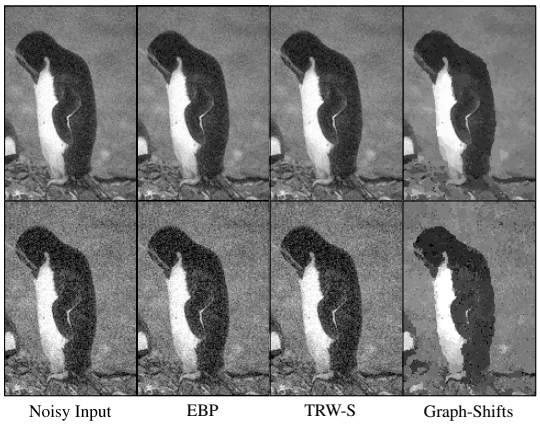

Figure 7. Visual comparison of restoration results between graph-shifts, EBP [7] and TRW-S [12]. Top row was perturbed by noise with a variance of 10 and bottom row was 20.

Figure 7 and table 1 present a comparison of the image restoration graph-shifts with the state-of-the-art methods. When computing the SSD error, we crop 5 rows and columns since the EBP implementation does not do labeling near the image borders. The changes for EBP and TRW-S are difficult to see visually; they mostly occur around the body of the penguin. We cannot explain the inability of the methods, other than graph-shifts, to obtain a visually satisfying restoration (considering all the methods were run on the same exact MRF energy function). However, the quantitative results (table 1) make a better comparative explanation. TRW-S is the best among the algorithms from the MRF benchmark [18], which agrees with one of their findings. However, the success of the ICM method in both speed and SSD error is a surprising result since it consistently scored last in [18].

[1]We are grateful to [3, 7, 12, 18, 19] for releasing their software; this has enabled us to do a direct comparison of the existing methods against our graph-shifts implementation.

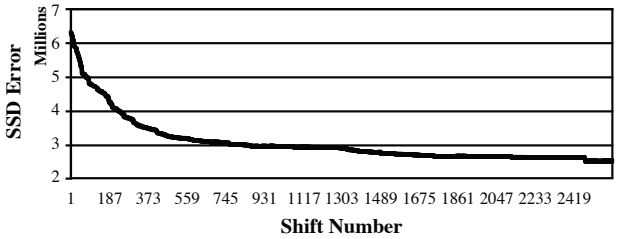

Figure 8. The graph shows the SSD error during graph-shifts.

Graph-shifts beats all the others in SSD error. We believe its success stems from the adaptive hierarchical nature of the algorithm; immediately after initialization the hierarchy represents groups of *similar* pixels that can get relabeled together. There is no need to wait for long-range interactions such as in message passing approaches (BP, TRW-S), which may be dominated by the relatively highly weighted data term in this energy. To further explain what's happening during the graph-shifts process, we show a graph of the SSD error during the minimization procedure in figure 8. It is clear that the early shifts, which typically although not monotonically occur at higher levels in the hierarchy, greatly reduce the SSD error.

In terms of efficiency, the standard message passing algorithms are slowest, and all others run in the order of seconds. Directly comparing speed is not completely fair because our implementation of graph-shifts is written in Java, but all other methods are in C++ (one expects at least a factor of two speedup).

4. High-Level Vision: Region Labeling

The high-level vision problem we consider is region labeling—coupled segmentation and classification. We tackle this problem using a hybrid discriminative-generative model minimized with the graph-shifts algorithm. We begin with a description of the hybrid model, then discuss the PBT.M2 multiclass discriminative modeling approach, and next present results on a 10- and a 21-class dataset. In both cases, graph-shifts is able to improve the pixel accuracy over the classifier alone, even by 20%. Our results on the 21-class dataset are superior to the state of the art on the same problem by 4.4%.

4.1. Modeling

The first term E_1 is a discriminative appearance model. It gives local evidence that the pixel μ takes model m_μ:

$$E_1(\mu, m_\mu) = -\log P\big(m_\mu|\mathbf{I}(S[\mu])\big) \qquad (7)$$

where $P(m_\mu|\mathbf{I}(S[\mu]))$ is the probability for the label m_μ at pixel $\mu \in D$ conditioned on a local sub-image of μ. The discriminative model is learned by a new multiclass boosting algorithm, PBT.M2 (discussed next in 4.2).

The second term E_2 is a *pairwise term* which both enforces a context model and encourages smooth boundaries:

$$E_2(\mathbf{I}(\nu), \mathbf{I}(\mu), m_\nu, m_\mu) = \\ -\kappa \log P(m_\nu, m_\mu) + \big(1 - \delta(m_\nu, m_\mu)\big) \ . \qquad (8)$$

This joint density on the models is also learned from training data. We compute a normalized histogram of label-pairs along region boundaries in all training images. E_2 represents a generative model about the prior knowledge in the scene structure. Hence, our total energy model is hybrid incorporating both discriminative and generative terms.

4.2. Multi-Class Discriminative Learning

To compute E_1, our task is to learn and compute the discriminative model $P(m_\mu|\mathbf{I}(S[\mu]))$. To demonstrate the versatility of graph-shifts, we work with two different models: i) the PBT.M2 model, discussed below, and ii) the auto-context model [21], which basically does discriminative modeling based on both appearance and spatial context. Each input sample is a sub-image and the output is the probability of the center pixel μ being on a region with model $m_\mu \in \{1, \ldots, K\}$. Complex appearance patterns of $\mathbf{I}(S[\mu])$ make this a difficult task. [20] adopt a probabilistic boosting tree (PBT) approach to learn and compute a multi-class classifier. However, the PBT performs 2-way splits only. This restriction can be inefficient when, e.g., with only 4 classes one still needs to compute 3 strong classifiers; a direct multiclass split is better.

Let the training set be $T = \{(\mathbf{I}_a, m_a), a = 1 \ldots n\}$ where $\mathbf{I}_a$ is a 11×11 image sample (patch), $m_a \in \{0 \ldots K\}$ denotes its class label, and n is the total sample number. Let $p(T, j)$ be the proportion of samples in T that belong to the jth class. The entropy of set T can be defined as $\text{info}(T) = -\sum_{j=0}^{T} p(T, j) \log_2(p(T, j))$. Lower entropy indicates a set with sparse classes.

We recursively construct a decision tree in which each tree node is either an AdaBoost [8] (2-class) strong classifier or AdaBoost.MH [10] (m-class) strong classifier. For a strong classifier $H \in \{H_2, H_m\}$ (either 2-class or m-class), it splits T into t sub-groups $(T_1, ..., T_t)$. We choose the H which obtains the biggest information gain

$$G(T, H) = -\sum_{i=1}^{t} \frac{|T_i|}{|T|} \text{info}(T_i) - \text{cost}(H), \qquad (9)$$

PBT.M2 ALGORITHM

Input: Labeled training examples $T = \{(\mathbf{I}_a, m_a), a = 1..n\}$ with each $m_a \in \{0..K\}$.
Output: Discriminative Model $P(m|\mathbf{I})$.
0 $H_2 \leftarrow$ trained 2-class AdaBoost [8] classifier.
1 $H_m \leftarrow$ trained m-class AdaBoost.MH [10] classifier.
2 Choose strong classifier, $H \in \{H_2, H_m\}$, that maximizes information gain $G(T, H)$.
3 Stop if error is smaller than a threshold.
4 Split the training set T using H and recurse.

Figure 9. PBT.M2 algorithm.

where the first term is similar to that in the well-known C4.5 algorithm [16] and $\text{cost}(H)$ computes the total computational cost for strong classifier H. Therefore, the choice of H balances between how well to separate the current set and the computational cost. Fig. (9) outlines PBT.M2.

PBT.M2 learns and computes an overall multi-class discriminative probability by

$$P(m|\mathbf{I}) = \sum_{l_1,..,l_n} \tilde{P}(y|l_n, ..., l_1), ..., Q(l_2|l_1, \mathbf{I})q(l_1|\mathbf{I}) \ ,$$

where l_i represents the ith layer in the tree, and $Q(l_i)$ computes the discriminative probability by each boosting node.

4.3. Experimental Results

We experiment with two multi-class datasets: a 10-class, 400-image subset of the LHI dataset [23], and a 21-class, 590-image dataset called MSRC from [17]. Examples from the LHI dataset are given in figure 1. In both cases, we randomly split the data into training and testing sets; LHI uses 170 training images and MSRC uses 327. We use the PBT.M2 model on LHI, and use the auto-context model [21] on MSRC.

Based on the labeled training images, we first train the PBT.M2 classifier to select and fuse a set of features out of a pool of around 10^5 features (color, intensity, position, and histograms on Gabor responses). This is our discriminative model. Second, we learn the pair-wise contextual relationships of the objects, which is our generative model part. For a test image, we first compute its classification maps (multi-class discriminative probabilities) using the learned PBT.M2 classifier; i.e., for each pixel, the probability that it belongs to each class. Then, a final labeling is obtained by minimizing the hybrid energy term 1 using graph-shifts. Computing the classification maps for all the pixels takes about 17 seconds on a modern PC and energy minimization by graph-shifts takes on average 3 seconds.

Figure 10 illustrates some LHI testing images by our algorithm. The first row shows the original images; the second row displays the manual labels; the third row are the initial classification maps (maximum discriminative probability); the fourth row shows the results without the region

Algorithm	Dataset	Accuracy
Adaboost Cascades [22]	LHI	50.0%
Classifier Only	LHI	55.9%
Graph-Shifts	LHI	**75.5%**
Shotton et al. [17]	MSRC	72.2%
Classifier Only	MSRC	74.0%
Graph-Shifts	MSRC	**76.6%**

Table 2. Total pixel accuracy comparisons for both datasets.

context term ($\kappa = 0$ in equation 8); the last row are the results by our overall algorithm. We show some example test results on the MSRC dataset in figure 11. Regions are labeled and color-coded. We see that even in the higher 21-class case, the graph-shifts minimizer can yield good results. Lower quality results are given to the right.

Table 2 gives a quantitative measure of the region labeling results. On the LHI dataset, the graph-shifts process improves the overall pixel accuracy by almost 20%. On the MSRC dataset, it improves the results by about 3%, the lower improvement in this case is because the auto-context model learns some E_2-like information directly into E_1 already; so, graph-shifts has less work to do. On the MSRC dataset, with graph-shifts, our overall accuracy is 76.6%, which is 4.4% higher than the published results in [17]. Confusion matrices for LHI (table 4) and MSRC (table 3) help to explain more details about the proposed method. Empty cells indicate $< 0.05\%$ confusion. In the LHI confusion, the animal/human (creature) and bridge score very low because of their comparative intra-class variability–to–amount of training data ratio. Most other confusion is intuitively explained: sky is, when misclassified, confused with water, and buildings are confused as mountains, etc.

5. Conclusion

In summary, this paper explored the graph-shift algorithm for labeling problems on natural images. Graph-shifts does energy minimization by dynamically manipulating an adaptive hierarchical decomposition of the image. For low-level vision, we explored the problem of image restoration. Using the exact same MRF model, we compared graph-shifts to state-of-the-art methods, including BP, TRW-S, and α-expansion, and we found that graph-shifts achieves the lowest sum-of-squared differences error at almost the fastest computing rate. In high-level vision, we tackled the region labeling (coupled segmentation and classification) problem on two multi-class datasets. We use a hybrid discriminative-generative model learned with an extension of the PBT algorithm called PBT.M2 that optimally selects between a 2-class and multi-class classifier at each node in the tree. Graph-shifts is able to improve upon the classification results, by 20% in one case. Our final labeling results are superior to the current published state-of-the-art results and obtained in seconds.

Acknowledgements

ZT and JJC (partially) are funded by NIH Grant U54 RR021813 entitled Center for Computational Biology. AY is funded by NSF 0413214.

References

[1] P. Anandan. A Computational Framework and an Algorithm for the Measurement of Visual Motion. *Intl. J. of Computer Vision*, 2(3):283–310, 1989. 1

[2] K. Barnard, P. Duygulu, R. Guru, P. Gabbur, and D. Forsyth. The Effects of Segmentation and Feature Choice in a Translational Model of Object Recognition. In *Proc. of CVPR*, 2003. 2

[3] Y. Boykov, O. Veksler, and R. Zabih. Fast Approximate Energy Minimization via Graph Cuts. *IEEE Trans. on PAMI*, 23(11):1222–1239, 2001. 1, 5

[4] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Blobworld: Image Segmentation using Expectation-Maximization and Its Application to Image Querying. *IEEE Trans. on PAMI*, 24(8):1026–1038, 2002. 2

[5] J. J. Corso, Z. Tu, A. Yuille, and A. W. Toga. Segmentation of Sub-Cortical Structures by the Graph-Shifts Algorithm. In N. Karssemeijer and B. Lelieveldt, editors, *Proc. of IPMI*, pages 183–197, 2007. 1, 2

[6] L. Fei-Fei and P. Perona. A Bayesian Hierarchical Model for Learning Natural Scene Categories. In *Proc. of CVPR*, 2005. 2

[7] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient Belief Propagation for Early Vision. *Intl. J. of Computer Vision*, 70(1), 2006. 1, 5

[8] Y. Freund and R. E. Schapire. A Decision-Theoretic Generalization of On-line Learning and an Application to Boosting. *J. of Comp. and Sys. Sci.*, 55(1):119–139, 1997. 6

[9] B. J. Frey and D. MacKay. A Revolution: Belief Propagation in Graphs with Cycles. In *Proc. of NIPS*, 1997. 1

[10] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. Technical report, Dept. of Statistics, Stanford University, 1998. 6

[11] S. Gordon, H. Greenspan, and J. Goldberger. Applying the Information Bottleneck Principle to Unsupervised Clustering of Discrete and Continuous Image Representations. In *Proc. of ICCV*, 2003. 2

[12] V. Kolmogorov. Convergent Tree-reweighted Message Passing for Energy Minimization. *IEEE Trans. on PAMI*, 28(10):1568–1583, 2006. 5

[13] S. Kumar and M. Hebert. Discriminative Random Fields: A Discriminative Framework for Contextual Interaction in Classification. In *Proc. of ICCV*, 2003. 2

[14] J. Li and J. Z. Wang. Automatic linguistic indexing of pictures by a statistical modeling approach. *IEEE Trans. on PAMI*, 25(9):1075–1088, 2003. 2

[15] L. Lu, K. Toyama, and G. D. Hager. A Two Level Approach for Scene Recognition. In *Proc. of CVPR*, volume 1, pages 688–695, 2005. 2

[16] J. R. Quinlan. Improved Use of Continuous Attributes in C4.5. *J. of Art. Intel. Res.*, 4:77–90, 1996. 6

[17] J. Shotton, J. Winn, C. Rother, and A. Criminisi. TextonBoost: Joint Appearance, Shape and Context Modeling for Multi-Class Object Recognition and Segmentation. In *Proc. of ECCV*, 2006. 2, 6, 7, 8

[18] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A Comparative Study of Energy Minimization Methods for MRFs. In *Proc. of ECCV*, volume 2, pages 16–29, 2006. 1, 4, 5

[19] M. Tappen and W. T. Freeman. Comparison of Graph Cuts with Belief Propagation for Stereo, using Identical MRF Parameters. In *Proc. of ICCV*, pages 900–907, 2003. 5

[20] Z. Tu. Probabilistic Boosting-Tree: Learning Discriminative Models for Classification, Recognition, and Clustering. In *Proc. of ICCV*, 2005. 2, 6

[21] Z. Tu. Auto-context and its application for high-level vision tasks. In *Proc. of CVPR*, 2008. (to appear). 6

[22] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. of CVPR*, 2001. 7

[23] Z. Yao, X. Yang, and S. C. Zhu. Introduction to a Large Scale General Purpose Ground Truth Dataset: Methodology, Annotation Tool, and Benchmarks. In *Proc. of EMMCVPR*, 2007. 2, 6

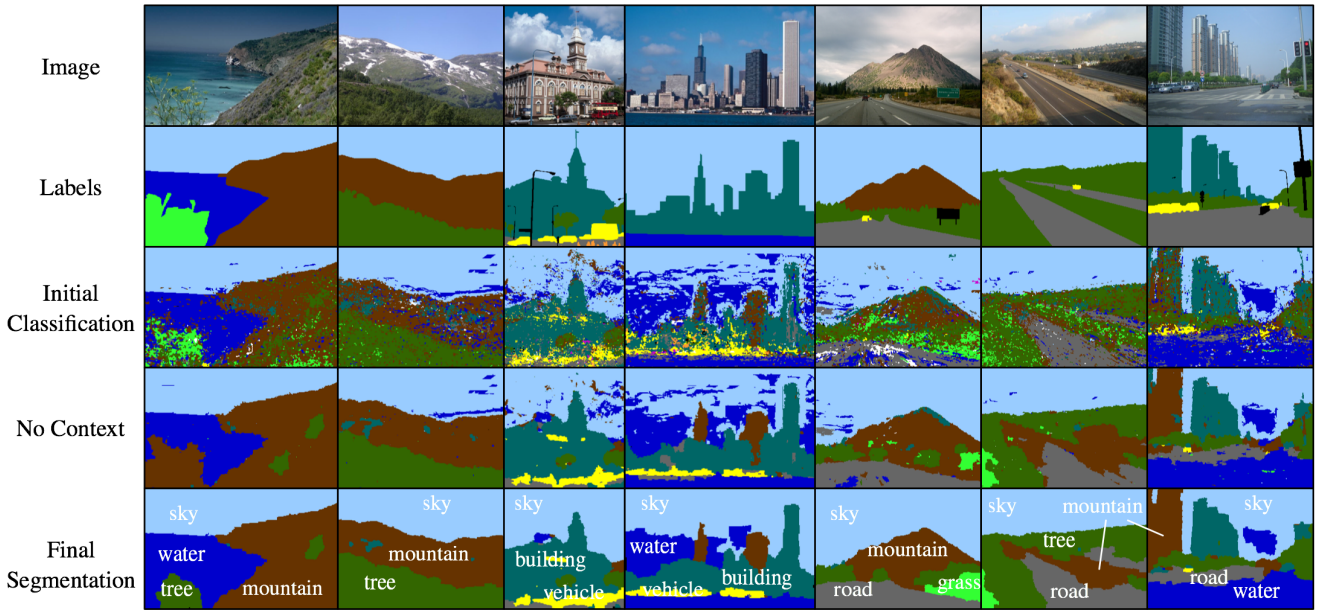

Figure 10. Example region labeling results on the LHI dataset (label legend in fig. 1). The two columns on the right make mislabelings.

building	**72.8**	0.7	6.1			1.5	1.2	0.9	0.2	3.4	1.0	0.1	0.3	0.4	5.0	1.3	3.8	0.8		0.6	1.0
grass	0.3	**93.1**	4.4	0.2	0.1	0.1	0.1	0.6			0.1			0.1		0.9	0.1				
tree	4.5	2.7	**84.7**	0.1		0.5	0.4	1.4		0.1	1.8	0.9				2.7	0.1				
cow	1.9	13.8	4.2	**66.1**	0.9	0.1		1.2				4.7		2.9				4.1		0.2	
sheep	1.0	15.5	0.3	8.1	**68.7**		0.3							4.1			0.3				
sky	1.4		1.2			**93.3**	0.3	2.6									0.1		1.1		
airplane	7.6	0.5	0.2			0.8	**89.8**	0.3									0.7				4.1
water	5.2	0.5	0.8			3.1	0.4	**69.6**		0.4				0.6		0.5	11.8		6.8	0.1	1.5
face	5.3	1.0	0.6			0.2		0.1	**84.6**	0.1		0.8			2.6	0.3				4.5	
car	18.3		1.2			1.2		5.2		**66.8**	0.6		1.3				5.2			0.2	1.5
bicycle	5.1		1.1							2.3	**86.4**						5.1				1.7
flower	2.8	2.9	1.5	0.1	0.1	1.4		0.1		7.0		**58.0**		7.1	14.0		0.7		2.3	2.0	
sign	20.2	0.1	2.0			0.5					0.5		**63.7**		11.2	0.2	1.7				0.1
bird	23.9	7.0	6.8			10.3	0.8	3.5		4.0	5.3			**25.0**		1.7	8.3		2.6	0.8	1.2
book	1.9		1.3			0.1		0.4	0.1	1.9		1.6			**90.9**		0.8			0.9	0.3
chair	23.5	2.8	11.6	0.3				0.3		3.2	3.3	2.4			3.8	**43.7**	4.8			0.5	
road	6.1	1.6	0.3		0.1	0.2	0.1	14.2		0.4	0.6			0.1		0.1	**73.8**		1.2	0.3	0.2
cat	4.4		2.2	1.4				9.5	5.6	0.3		0.7		0.8		1.2	18.7	**40.8**	12.0	2.4	0.6
dog	9.5	6.5	3.2	7.6	4.6			0.8	15.3							1.2	11.4	0.7	**30.4**	8.9	
body	14.0	4.0	1.4	0.8		0.1		1.0	4.3	0.8	0.1	9.3	0.8	1.0	7.1	2.2	1.0		0.2	**51.7**	1.7
boat	19.4		3.4			0.7		11.4		44.3				15.2	4.9		0.2			0.5	**73.8**

Table 3. Confusion matrix for the MSRC dataset (empty cells have values < 0.05). Total pixel accuracy is 76.6%, before graph-shifts it is 74.0%, and the published result by Shotton et al. [17] 72.2%.

sky	**87.8**	8.0			0.3	1.8		0.8		
water	2.7	**65.2**	4.6	2.1	5.9	15.6		3.0		0.3
road	0.1	15.7	**69.5**	0.4	0.8	4.8		3.3		0.9
grass	0.5	0.9	0.6	**68.4**	7.7	18.9		1.5		0.1
tree	1.1	1.9	0.5	4.3	**65.8**	16.9		7.3		0.2
mountain	1.9	3.4	0.6	3.9	9.8	**73.7**		5.2		0.1
creature		4.9	5.9	3.0	4.0	29.4	**8.3**	39.6		2.7
building	1.8	1.1	1.9	1.1	3.7	9.6		**78.6**		0.5
bridge	0.2	2.1	6.3	6.5	4.2	13.0		56.7	**7.9**	2.2
vehicle		5.2	10.8	0.1	6.5	20.3	0.1	21.5		**34.3**

Table 4. Confusion matrix for LHI dataset. Total pixel accuracy is 73.5%, pixel accuracy with just the PBT.M2 model is 55.9%

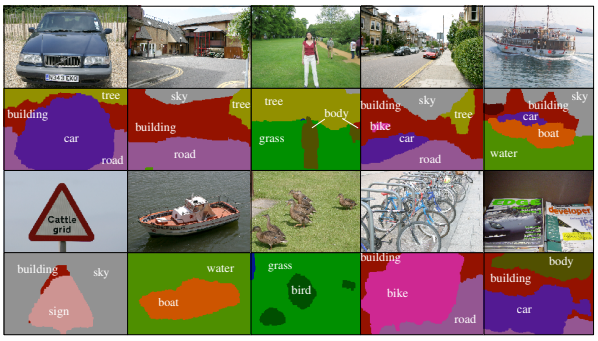

Figure 11. Example results on the MSRC dataset The last column shows a poor labeling.