Local Label Descriptor for Example Based Semantic Image Labeling

Yiqing Yang[1], Zhouyuan Li[1], Li Zhang[1], Christopher Murphy[2], Jim Ver Hoeve[1], and Hongrui Jiang[1]

[1] University of Wisconsin-Madison
[2] University of California, Davis

Abstract. In this paper we introduce the concept of *local label descriptor*, which is a concatenation of label histograms for each cell in a patch. Local label descriptors alleviate the label patch misalignment issue in combining structured label predictions for semantic image labeling. Given an input image, we solve for a label map whose local label descriptors can be approximated as a sparse convex combination of exemplar label descriptors in the training data, where the sparsity is regularized by the similarity measure between the local feature descriptor of the input image and that of the exemplars in the training data set. Low-level image over-segmentation can be incorporated into our formulation to improve efficiency. Our formulation and algorithm compare favorably with the baseline method on the CamVid and Barcelona datasets.

1 Introduction

Local image feature descriptors, such as SIFT and HOG, are widely used in computer vision, for example, in object detection, key point matching, and scene classification, to name just a few. When used in semantic image segmentation, the local feature descriptors are often first fed into local classifiers that produce class label scores, and then the scores are combined in a Conditional Random Field (CRF) to generate the final segmentation. Previously, local classifiers are either two-class (car or non-car) or multi-class (human, building, or sky). In both cases, although the local classifiers use local context information in making decisions, no context information is present in their label score output that is passed onto the higher level CRF.

In this paper, we propose a novel energy formulation to model context for spatially consistent semantic image segmentation. Our energy function is based on sparse coding of *local label descriptors*. Local label descriptors are analogous to local feature descriptors, such as HOG. Recall that when we construct a local feature descriptor, the common idea is to treat an image patch as a set of overlapping cells and concatenate the histograms of image features in each cell to form the descriptor. Applying this idea to a label patch, we define a local label descriptor to be a concatenation of label histograms for each cell in the label patch. Local label descriptors offer two computational conveniences in pixel labeling.

A. Fitzgibbon et al. (Eds.): ECCV 2012, Part VII, LNCS 7578, pp. 361–375, 2012.

– A local label descriptor models the probability of local label patterns. Continuous mathematical operations (*e.g.*, linearly combination) can be defined on such descriptors but cannot be done on hard label patches.
– If two local feature descriptors are similar, we should expect *their histograms of the labels in corresponding cells to be similar*, rather than the label values, because feature descriptors are designed to be robust to small geometric misalignment and hence the label patches corresponding to similar feature descriptors may not be perfectly aligned, especially for label patterns with detailed label transition (*e.g.*, a lamp post with a sky background). This problem is illustrated in Figure 1.[1]

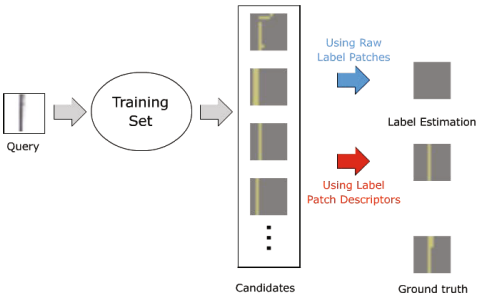

Fig. 1. Illustration of the misalignment problem. A patch with lamp post is queried in the training set. Several candidates with similar feature descriptors are retrieved; although all candidates contain the lamp post in the sky, but they are not well aligned. Each individual pixel gets most votes for sky background, and the lamp post structure is lost in the final output label. We can resolve this issue by using the label patch descriptors which is robust to small geometric transformation.

Building upon local label descriptors, we cast the semantic image labeling as a convex optimization problem, as illustrated in Figure 2. Given an input image, we solve for a label map whose local label descriptors can be approximated as a sparse convex combination of exemplar label descriptors in the training data, where the sparsity is regularized by the similarity measured between the local feature descriptor of the input image and that of the exemplars in the training data set. Since the joint space of feature descriptor and label descriptor is huge, we propose an iterative approximate method to optimize the proposed objective function. We also demonstrate that low-level image over-segmentation can be incorporated into our formulation to further improve the solution. In summary, this paper makes the following contributions:

– We introduce the concept of local label descriptor and show its effectiveness in semantic image labeling.
– Based on the local label descriptor, we formulate the semantic image labeling as a convex optimization problem and propose a continuous optimization

[1] This problem is essentially an aliasing problem in sampling the training patches. If we could afford to store all possible training patches at the highest sampling density with respect to various geometric transformation, this problem might be alleviated. However, it may not be practical to do so; very often in practice, we face a budget of limited computational resources. Therefore, it is beneficial to have a general solution to this problem.

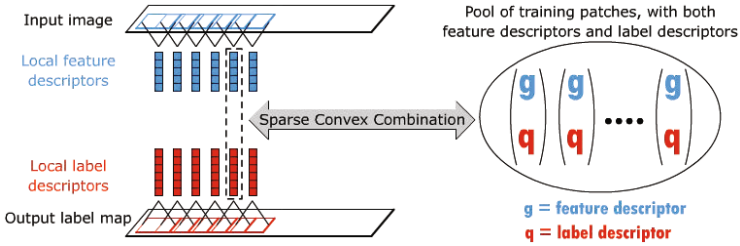

Fig. 2. A diagram illustrating our formulation of semantic image labeling using local label descriptors. Each local label descriptor is a concatenation of label histograms for each cell in a patch. Because computing the histogram over a cell can be viewed as a box filtering whose support has the same size as the cell, our formulation can be viewed as a label descriptor deconvolution process: given the predicted local label descriptors, we seek to restore a per-pixel label map.

procedure that is less prone to local minimum than the previous discrete greedy search [1]. Our formulation can incorporate low-level image segmentation that is often used in the literature.

– Under our new problem formulation, we show that it is possible to use the intermediate label estimation to update label prediction exemplars and improve the accuracy of final label estimation.

We evaluate the idea of local label descriptor and our algorithm on CamVid [2] and Barcelona [3] datasets and show that they improve the performance of the baseline algorithm [1]. As the structured label prediction is a rather unexploited topic in computer vision, we wish our idea and formulation may provoke further studies on the topic.

2 Related Work

One proven approach to image labeling is by optimizing a Conditional Random Field (CRF). Traditional CRF models [4,5] consist of unary terms and pairwise terms. The unary terms measure the likelihood of associating a pixel with a particular class label while the pairwise terms regularize neighboring pixels taking different labels. Although the pairwise terms have limited power to express long range contextual information, several variants of CRF have been proposed to address this issue. For example, Kohli *et al.* [6] propose robust high order potentials for enforcing label consistency. Galleguillos *et al.* [7] and Ladický *et al.* [8] propose to model co-occurrence using CRF. Gonfaus *et al.* [9] propose harmony potentials to allow for different labels to appear in one region at different scales. Several other works also in this endeavor include [10,11,12,13].

Another approach to image labeling is based on the idea of scene alignment, *a.k.a.* non-parametric scene parsing. This idea was first proposed in the seminal work by Russell *et al.* [14], in which an input image is matched in a set of training images with labels. The labels of the retrieved images are transferred to the input image. In [15], Liu *et al.* used SIFT flow to transfer per-pixel labels. SIFT flow is most effective when the spatial layout of objects is similar between

the input image and the retrieved images. To relax this requirement, Tighe and Lazebnik [3] introduced a scalable approach that operates on super-pixels, which is more efficient than [15] that operates on pixels.

The third approach is using Recursive Neural Networks (RNN) [16]. This method requires more sophisticated training but the testing speed is very fast. This method reports comparable results with [3]. While it has its biological inspiration, the intuition behind this method is not as well understood than the first two approaches.

Our method is most related to the work by Kontschieder *et al.* [1], in which exemplar local label patches are retrieved using a random forest and a greedy discrete search is used to produce a label map that has the most consensus to the retrieved label patches. In our work, instead of using raw label patches for structure prediction, we propose to use local label descriptors, which is more robust to the label patch misalignment issue in label voting. Furthermore, instead of directly solving for the hard label map, we propose to solve a label probability map, which is formulated as a convex optimization problem. We show that a continuous convex optimization is less likely to be stuck in local minimum and thereby improves the accuracy of label estimation. Finally, we demonstrate that low level over-segmentation can be incorporated in our formulation to reduce the number of unknown variables and make the optimization more efficient.

Our work also takes much inspiration from the work by Huang *et al.* [17], which proposes to use a global label descriptor to select scene-specific CRF models to improve labeling accuracy. Our label descriptor is designed for local image patches and we show that intermediate estimates of local label descriptors can be used to update label structure prediction and thus improve the accuracy of final label estimates.

3 Problem Formulation

We state the semantic image labeling problem as follows. Assume a set of training images, each with a label map. The label of a pixel indicates the class to which the pixel belongs. Given a test image, the task is to compute its label map.

3.1 Notation and Representation

We use i as the index of a pixel in an image $\mathcal{I}$ and let K be the number of classes and $\mathcal{K} = \{1, \cdots, K\}$ be the set of classes. We represent an image as a set of overlapping cells; each cell is an array of $C \times C$ pixels. Let N and $\mathcal{N}$ be the number and set of all cells, respectively, and n be the index of a cell in $\mathcal{N}$. Each patch is a consecutive array of $B \times B$ cells. Similarly, let M and $\mathcal{M}$ be the number and set of all patches, respectively, and m be the index of a patch in $\mathcal{M}$. To avoid introducing too many symbols, we use the notation $n \in m$ to indicate that cell n is used in patch m. (The cell size C and patch size B, as well as cell stride and patch stride, are parameters to set up the cells and patches, which will be provided in Section 6). For each training image, given the cell and

patch structure, we extract a local image feature descriptor for each patch. HOG is used in our implementation, which is a concatenation of gradient orientation histograms computed over each cell in the patch. Each training image has a label map; we use the cell and patch structure to extract a local label descriptor for each patch.[2] Specifically, we first compute a histogram of labels for each cell in the patch and then concatenate the histograms to form the label descriptor for the patch. In the end, the training data set is a set of exemplar pairs of local feature descriptor and local label descriptor. Let the number of pairs be J, and $(\mathbf{g}_j, \mathbf{q}_j)$ be the j'th pair, where $\mathbf{g}_j$ and $\mathbf{q}_j$ are feature and label descriptors, respectively.

3.2 The Objective Function

Given a testing image, we also set up the cell and patch structure. We seek a label map whose local descriptors can be represented as a sparse combination of exemplar label descriptors in the training set. Instead of solving for the label map directly, we formulate the problem based on the the probability that one pixel belongs to a particular class.

Specifically, let $p_{i,k}$ be the probability of pixel i belonging to class k; all $p_{i,k}$'s are non-negative and sum up to 1 over k for each i. Using $\mathbf{p}_i = [p_{i,1}; \cdots ; p_{i,K}]$ to represent the label probability vector for pixel i, we can evaluate the label histogram over cell n as $\mathbf{p}_n = \frac{1}{C^2} \sum_{i \in n} \mathbf{p}_i$. Then the label descriptor for patch m is simply a concatenation of $\mathbf{p}_n$ for all cell n in patch m. This label descriptor can be expressed as a matrix vector product $\mathbf{A}_m \mathbf{p}$, where $\mathbf{p} = [p_{i,k}]_{i \in \mathcal{I}, k \in \mathcal{K}}$ is the vector of label probability values for all pixels and $\mathbf{A}_m$ is a constant coefficient matrix whose elements depend on the location of patch m in the image.

To design our objective function, we seek to express $\mathbf{A}_m \mathbf{p}$ as a sparse convex combination of exemplar label descriptors in the training set. For patch m, let $\{a_{m,j}\}_{j \in \mathcal{J}}$ be its combination coefficients. We hope $a_{m,j}$ is nonzero only if the feature descriptor exemplar $\mathbf{g}_j$ in the training set is similar to the feature descriptor of patch m, $\mathbf{f}_m$. If we fix $\mathbf{A}_m \mathbf{p}$, such $\{a_{m,j}\}_{j \in \mathcal{J}}$ can be computed by minimizing the following regularized objective function:

$$\begin{aligned} \phi_m(\{a_{m,j}\}_{j \in \mathcal{J}}, \mathbf{p}) &= \|\mathbf{A}_m \mathbf{p} - \sum_{j \in \mathcal{J}} \mathbf{q}_j a_{m,j}\|^2 + \eta \sum_{j \in \mathcal{J}} w_{m,j} a_{m,j} \\ \texttt{subject to} \quad a_{m,j} &\geq 0 \quad \texttt{and} \quad \sum_{j \in \mathcal{J}} a_{m,j} = 1 \\ p_{i,k} &\geq 0 \quad \texttt{and} \quad \sum_{k \in \mathcal{K}} p_{i,k} = 1 \end{aligned} \tag{1}$$

where the weight $w_{m,j}$ encourages non-zero coefficients to be those whose corresponding feature descriptor exemplars $\mathbf{g}_j$ are similar to the feature descriptor $\mathbf{f}_m$ of patch m. We set $w_{m,j} = \|\mathbf{f}_m - \mathbf{g}_j\|$.

[2] The cell size and cell stride for label descriptors can be different from those used for feature descriptors. Without loss of generality, we assume them to be the same when describing our formulation.

We add Eq. (1) for all patches together to form our global objective function Φ with respect to $\mathbf{p}$ and all combination coefficients $\{a_{m,j}\}_{m\in\mathcal{M},j\in\mathcal{J}}$

$$\Phi(\{a_{m,j}\}_{m\in\mathcal{M},j\in\mathcal{J}},\mathbf{p}) = \sum_m \phi_m(\{a_{m,j}\}_{j\in\mathcal{J}},\mathbf{p}) + \lambda \sum_{(i,i')} v_{i,i'}\|\mathbf{p}_i - \mathbf{p}_{i'}\|^2 \quad (2)$$

where $\|\mathbf{p}_i - \mathbf{p}_{i'}\|^2$ is a regularization that encourage label smoothness. The regularization is modulated by the weight $v_{i,i'}$, which measures the color similarity between the adjacent pixel pair (i, i').

3.3 Segmentation-Based Representation

In Eq. (2), each pixel's label probability vector is unknown, and the whole image has a large number of unknowns to solve for. We can incorporate low-level segmentation in our formulation to make the label inference more efficient. Given an over-segmentation of an image with S segments, we assume all the pixels in one segment share the same label probability vector. In this case, the segmentation serves as a change of variables that reduces the number unknowns in the optimization.

Specifically, let $\mathbf{z}_s = [z_{s,1}; \cdots ; z_{s,K}]$ be the label probability vector for segment s and $\mathbf{z} = [\mathbf{z}_1; \cdots ; \mathbf{z}_S]$ be the concatenation of label probability vectors of all segments. Then we have $\mathbf{p} = \mathbf{S}\mathbf{z}$, where $\mathbf{S}$ is a matrix whose element at i'th row and s'th column is 1 if pixel i belongs to segment s, and 0 otherwise. Substitute $\mathbf{p} = \mathbf{S}\mathbf{z}$ into Eq. (2) gives the following objective function with much fewer of unknowns.

$$\Phi_{\mathtt{seg}}(\{a_{m,j}\}_{m\in\mathcal{M},j\in\mathcal{J}},\mathbf{z}) = \sum_m \phi_m(\{a_{m,j}\}_{j\in\mathcal{J}},\mathbf{S}\mathbf{z}) + \lambda \sum_{(s,s')} u_{s,s'}\|\mathbf{z}_s - \mathbf{z}_{s'}\|^2 \quad (3)$$

where $u_{s,s'}$ measures similarity between the average colors of adjacent segment pair (s, s').

4 Optimization

We first note that Eq. (3) is a convex function. Any downhill algorithm will converge to a global optimum solution. However, the challenge in practice is that the dimension of coefficients $\{a_{m,j}\}_{m\in\mathcal{M},j\in\mathcal{J}}$ is very high, which is the number of patches times the number of exemplars in the training set. Therefore, it is impractical to optimize Eq. (3) with respect to all $\{a_{m,j}\}$ simultaneously.

Instead, we propose an iterative approach to optimize Eq. (3). Our basic idea is that, at each iteration, we select only a small number of exemplars for each patch m and only optimize the corresponding coefficients while enforcing other coefficients to be zero. We call the selected exemplars for patch m its *candidate set*, denoted with $\mathcal{J}_m$, ($\mathcal{J}_m \subset \mathcal{J}$). Note that, in practice, the solution of the coefficients $\{a_m\}_{j\in\mathcal{J}_m}$ for patch m is sparse; that is, the number of non-zeros is much smaller than the size of the candidate set $\mathcal{J}_m$. As a result, in the next iteration, we can keep the exemplars that correspond to the non-zeros coefficients and replace the ones with zero coefficients with new exemplars. Therefore, the candidate set $\mathcal{J}_m$ for each patch m is evolving as iterations continue.

Using the candidate sets $\{\mathcal{J}_m\}_{m\in\mathcal{M}}$ for all patches, the objective function in Eq. (3) becomes

$$\Phi(\{a_{m,j}\}_{m\in\mathcal{M},j\in\mathcal{J}_m},\mathbf{z}) = \sum_{m\in\mathcal{M}} \phi_m(\{a_{m,j}\}_{j\in\mathcal{J}_m},\mathbf{Sz}) + \lambda \sum_{(s,s')} u_{s,s'}\|\mathbf{z}_s - \mathbf{z}_{s'}\|^2 \quad (4)$$

We next discuss how to minimize Eq. (4) and then present our strategy to update the candidate sets.

4.1 Minimizing Eq. (4)

The objective function in Eq. (4) is quadratic; its variables are non-negative subject to the normalization constraints:

$$\forall s \in \mathcal{S}, \sum_{k\in\mathcal{K}} z_{s,k} = 1 \quad \texttt{and} \quad \forall m \in \mathcal{M}, \sum_{j\in\mathcal{J}_m} a_{m,j} = 1 \quad (5)$$

We could solve this constrained optimization problem using standard quadratic programming with constraints. However, given the special nature of the constraints, we apply a multiplicative weight [18] type of algorithm. Specifically, we use a change of variables to eliminate the constraints and update each variable by multiplying it with a factor to minimize the objective function. Without loss of generality, we first simplify the constraints in Eq. (5) to be a single normalization constraint; multiple normalization constraints can be easily handled with a simple modification. Since the objective function in Eq. (4) is quadratic, we can write the problem as

$$\begin{aligned} &\min_{\mathbf{x}} \left\{\Phi(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{H}\mathbf{x} + \mathbf{b}^{\mathsf{T}}\mathbf{x}\right\} \\ &\texttt{subject to} \quad \mathbf{x} \geq 0, \mathbf{1}^{\mathsf{T}}\mathbf{x} = 1 \end{aligned} \quad (6)$$

To satisfy the non-negative and normalization constraints on $\mathbf{x}$, we parameterize $\mathbf{x}$ using $\mathbf{y}$ of the same dimension as $\mathbf{x} = \frac{\exp(-\mathbf{y})}{\mathbf{1}^{\mathsf{T}}\exp(-\mathbf{y})}$, where $\exp(-\mathbf{y})$ is evaluated element-wise. Starting from an initial estimate $\mathbf{x}^{(0)}$, we iteratively update $\mathbf{x}^{(t+1)} \propto \mathbf{x}^{(t)} \odot \exp(-\Delta\mathbf{y}^{(t)})$, where $\odot$ denotes element-wise product; $\Delta\mathbf{y}^{(t)}$ is obtained by minimizing $\arg\min_{\Delta\mathbf{y}} = \Phi(\mathbf{x}^{(t)} + \frac{\partial\mathbf{x}}{\partial\mathbf{y}}\Delta\mathbf{y})$ without constraints, where $\frac{\partial\mathbf{x}}{\partial\mathbf{y}}$ is the Jacobian matrix at $\mathbf{y}^{(t)}$. It is straight forward to verify that the Jacobian matrix is

$$\frac{\partial\mathbf{x}}{\partial\mathbf{y}} = \mathbf{x}\mathbf{x}^{\mathsf{T}} - \texttt{diag}(\mathbf{x}) \quad (7)$$

Note that the computational cost of $\frac{\partial\mathbf{x}}{\partial\mathbf{y}}\Delta\mathbf{y} = \mathbf{x}(\mathbf{x}^{\mathsf{T}}\Delta\mathbf{y}) - \texttt{diag}(\mathbf{x})\Delta\mathbf{y}$ is linear (rather than quadratic) with respect to the dimension of $\mathbf{x}$.

If the normalization constraint in Eq. (6) is defined for several non-overlapping groups of elements in $\mathbf{x}$, the algorithm remains the same, except that the parameterization is applied to each group separately and the Jacobian matrix $\frac{\partial\mathbf{x}}{\partial\mathbf{y}}$ has a block-wise diagonal structure, with each diagonal block still having the special form as in Eq. (7).

4.2 Evolving Candidate Sets

Once we update the segment label probability vectors $\mathbf{z}$ and coefficients for current candidate sets $\{a_{m,j}\}_{m\in\mathcal{M},j\in\mathcal{J}_m}$, we update the candidate sets $\{\mathcal{J}_m\}_{m\in\mathcal{M}}$

for all patches in preparation for the next iteration. Specifically, we seek to replace the exemplars in $\mathcal{J}_m$ that correspond to zero coefficients $a_{m,j}$ with new exemplars. Several strategies exist. For example, for each patch m, we could swap in a random set of exemplars in $\mathcal{J}$ and continue the optimization. However, this strategy leads to slow convergence.

Our strategy is to effectively use the estimated label to guide the candidate set update. Note that, because of the sparse regularization in Eq. (3), the solution of $\{a_{m,j}\}_{j\in\mathcal{J}_m}$ is sparse for each patch m, which in turn makes the solution of segment label probability $\mathbf{z}_s$ sparse. If $\mathbf{z}_s$ has only one (significantly) nonzero element, it means segment s has a confident class label estimate. Otherwise, there is uncertainty in labeling this segment. For example, in practice, we find it is easy for a pixel on "sidewalk" to have non-zero label probabilities for both "road" and "sidewalk", because the local appearance of both road and sidewalk are similar in many cases. We hope to find exemplars to help disambiguate the estimate of $\mathbf{z}_s$ in the future iteration. Our intuition, which is based on empirical observation, is that nonzero components in $\mathbf{z}_s$ indicate the potential labels.

Recall that for each patch, its label descriptor is a concatenation of label histograms from all its cells. Since the cells in the patch is a $B \times B$ 2D array, the label descriptor can be viewed as a volume data of size $B \times B \times K$, where K is the number of classes. One slice of this volume with constant class index k can be viewed as the local response map of this patch for class k. Let $\mathbf{q}$ be a local label descriptor, we denote this slicing operation as $\texttt{Slice}(\mathbf{q}, k)$, which produces a local response map for class k.

Given the current label descriptor estimation of the patch m, for each of its non-zero local response maps, we look for label descriptor exemplars with similar local response map of the same class k in the training data. We also require the corresponding feature descriptor exemplars to be similar to the feature descriptor of the patch. To balance these two requirements, we use the following heuristic distance metric.

$$\left\| \frac{\texttt{Slice}(\mathbf{A}_m\mathbf{p}, k)}{\|\texttt{Slice}(\mathbf{A}_m\mathbf{p}, k)\|_\infty} - \texttt{Slice}(\mathbf{q}_j, k) \right\| + \eta \|\mathbf{f}_m - \mathbf{g}_j\| \tag{8}$$

where $\mathbf{p} = \mathbf{S}\mathbf{z}$ is the pixel label probability map, $\mathbf{A}_m\mathbf{p}$ is the current estimation of label descriptor for patch m, $\|\cdot\|_\infty$ is the infinity norm that takes the maximum value in the matrix. The response map of current label descriptor estimation is normalized because the exemplar label descriptors are always very sparse, but current label descriptor estimation may not be. Normalizing the response map of the latter makes the two response maps more comparable.

4.3 Using Random Forests for Candidate Set Search

Given Eq. (8) as distance metric, we use a KNN algorithm to update the candidate set. Several approaches exist for this purpose, such as the various exact KNN methods evaluated in [19] for image patch query, the approximated NN methods such as LSH [20], and more recently, the PatchMatch [21]. Searching for KNN is not a technical contribution in this paper. We use the random forest

as a data structure for approximate nearest neighbor search. The construction of the random forests with structured labels follows [1], with slight modifications made to adapt from hard labels to label descriptors; the details are provided in our supplemental material.

Given a random forest, at each iteration, to update the candidate set $\mathcal{J}_m$ for patch m, we randomly pick one tree from the forest, and pass the feature descriptor of patch m from the root to the leaf. The exemplar feature descriptors may not be very similar to the query feature descriptor. Then we backtrack by moving up from the leaf to the root and traverse the sibling subtrees that are not visited during the pass from the root to the leaf node. The backtracking stops if a pre-determined number of exemplars are visited; 12000 is used in our experiment. From these 12000 exemplars, we select the top candidates, measured by Eq. (8), to update the candidate set for patch m.

5 Relationship to [1]

Kontschieder *et al.* [1] proposed the first method that uses random forests to exploit structured labels. Our work is inspired by this seminal work, which can be viewed as a special case of our formulation. In this section, we elaborate the comparison and in the next section, we present experiment results to demonstrate the advantage of our formulation. Specifically, [1] is a special case of our formulation when:

- No segmentation is used (or each pixel is a segment)
- No label histogram descriptor is used (or the cell size over which the label histogram is computed is one pixel)
- Elements of $p_{i,k}$ and $a_{m,j}$ are constrained to be binary ($p_{i,k}, a_{m,j} \in \{0, 1\}$)
- $\eta = 0$, *i.e.*, no sparse regularization that favors label patch predictions whose feature descriptors are more similar to the query feature descriptor
- $\lambda = 0$, *i.e.*, no smoothness regularization between adjacent pixels with similar colors
- No candidate set evolution (fixed to the initial retrieval result)
- Discrete greedy search (rather than relaxed continuous convex optimization)

Specifically, when $\eta = \lambda = 0$ and elements of $p_{i,k}$ and $a_{m,j}$ are constrained to be binary and subject to the constraints that for each i, $\sum_k p_{i,k} = 1$, and for each m, $\sum_j a_{m,j} = 1$, our energy function is equivalent to

$$\Phi(\{a_{m,j}\}_{m\in\mathcal{M},j\in\mathcal{J}_m}, \mathbf{p}) = \sum_m \sum_{j\in\mathcal{J}_m} a_{m,j} \|\mathbf{A}_m\mathbf{p} - \mathbf{q}_j\|^2 \tag{9}$$

To minimize this energy with respect to the binary variables, we note that fixing $\mathbf{p}$, for each m, $a_{m,j} = 1$ if $\mathbf{q}_j$ is the closest to the current label estimation $\mathbf{A}_m\mathbf{p}$; and $a_{m,j} = 0$, otherwise. Fixing $\{a_{m,j}\}$, for each pixel i, its label probability $p_{i,k}$ will be 1 for label k if the label k receives most votes at pixel i from all the selected label patches that cover pixel i. This is exactly the label inference algorithm presented in [1]. Our optimization can be viewed as a convex relaxation of the discrete greedy search. In summary, our formulation has three advantages over the previous work.

- Our convex optimization is less likely to be stuck in a local minimum. The relaxed probability vectors contain labeling uncertainties, which can be exploited in the optimization.
- By combining the local label descriptors and segmentation based representation, our formulation is more robust to label patch misalignment, which prevents fine label structures from being correctly labeled.
- The initial query of local feature descriptors is not guaranteed to obtain correct label predictions if the local feature is ambiguous. Our formulation allows the candidate set of label predictions to be updated during the optimization, which may lead to higher chance of finding correct label predictions after context information is propagated.

6 Experiments

We tested our method on two datasets, CamVid [2] and Barcelona [3]. In this section we report a subset of the performance comparisons between our method and the baseline method [1]. **More evaluations are available in the supplemental material.** Our method extends [1] in the following aspects:

(1) **Segmentation-based** *vs.* Pixel-based
(2) **Convex Continuous Optimization** *vs.* Discrete Greedy Search
(3) **Label Histogram Descriptor** *vs.* Raw Label Patch
(4) **Evolving Candidate Set** *vs.* Freeze Initial Candidate Set

We tested both combinations of (1)+(2)+(3) and (1)+(2)+(3)+(4). Note that if we do not apply any of these options, the method will be equivalent to the baseline method. If discrete greedy search is evaluated, we initialize the solution using the Simple Fusion as in [1]. If continuous optimization is evaluated, we initialize all unknown variables using uniform probability distribution. After continuous optimization, we generate the label map by first selecting the class label with the dominant label probability and then refine the label map with the discrete greedy search.

In all of our experiments, the training images and the testing images are resized to be within a box of 320×240. Patches are sampled on a grid with a stride of 6 on both training images and testing images. We extract extra training patches from a pyramid with a factor of 0.8 to increase tolerance to scale variation. Patches from the second scale and the third scale are sampled with stride 5 and 4, respectively. The cell size $C = 8$ for both feature and label descriptors in all scales; each patch has 24×24 pixels. To extract HOG features, we set up 5×5 cells in a patch with a cell stride of 4. To extract label descriptors, we set up 13×13 cells centered within the patch with a cell stride of 1.

We train 6 random trees to form one forest in our experiment, and each tree is trained on the whole training set. When constructing the tree, we do not stop splitting a node until the maximal difference over all the components of the descriptor has converged within a predefined threshold.

Given a testing image, we over-segment the image using MeanShift[3] with spatial and range bandwidth 2 and minimum area 12, so that one segment is very likely to have only one label. To evaluate the results, we use three measurements as in [3], including the global pixel-wise recall, per-class pixel-wise recall, and the average intersection vs. union score used in PASCAL VOC challenges.

We implemented these methods in C++ with OpenCV, and run them on a workstation with an Intel E5506 (2.13GHz) CPU. The computational cost for one iteration depends on the number of segments and the number of query patches. We achieve 0.5 second per iteration under our experiment setup.

6.1 Experiments on CamVid Dataset

The Cambridge-driving Labeled Video Database (CamVid) is a collection of videos with object class semantic labels. It consists of over 10 minutes of high quality 30Hz footage from four streams. One of the four streams (with prefix 0001TP) was taken during dusk, the other three were taken during daylight. In total, there are 600 such images. We only used 11 commonly used classes from the 32 provided in the dataset. In our experiments on CamVid, we use 367 training images as in [1], from which we sampled 1804172 patches for random forest construction. The testing set for CamVid consists of 172 images from daylight video frames. We set $u_{s,s'}$ in Eq. (4) as follows. If the L_2 distance between the average colors of two adjacent segments is less than a threshold, $u_{s,s'} = 1$; 0, otherwise. We set $\lambda = 10$ and $\eta = 1$.

Table 1. Evaluation of our methods on CamVid

Method	Global	Avg(Class)	Avg(Pascal)
Our Baseline	64.42	30.18	22.25
(1)+(2)+(3)	73.46	36.45	29.39
(1)+(2)+(3)+(4)	73.67	36.34	29.55

	road	building	sky	tree	sidewalk	bicyclist	fence	pedestrian	car	column
Our Baseline	98.16	70.72	55.07	56.24	9.90	0.00	0.07	0.02	11.52	0.05
(1)+(2)+(3)	97.57	79.31	88.24	66.29	11.09	0.00	0.10	1.03	17.06	3.76
(1)+(2)+(3)+(4)	98.03	80.73	88.86	61.52	12.49	0.07	0.05	1.09	16.40	4.13

In Table 1, we compare the performance of our methods with the baseline method. Our methods score higher on overall accuracy using different criteria. Besides, there are noticeable boosts on accuracy for most classes, for both large classes (*e.g.* sky, trees) and small classes (*e.g.* columns, cars). Visual comparisons of several example results are shown in Figure 3.

Note that our implementation of the baseline method does not reproduce the high accuracy reported in [1]. We believe that this is mainly due to the details of tree constructions and image feature extraction, some of which are not clearly

[3] http://coewww.rutgers.edu/riul/research/code/EDISON/index.html

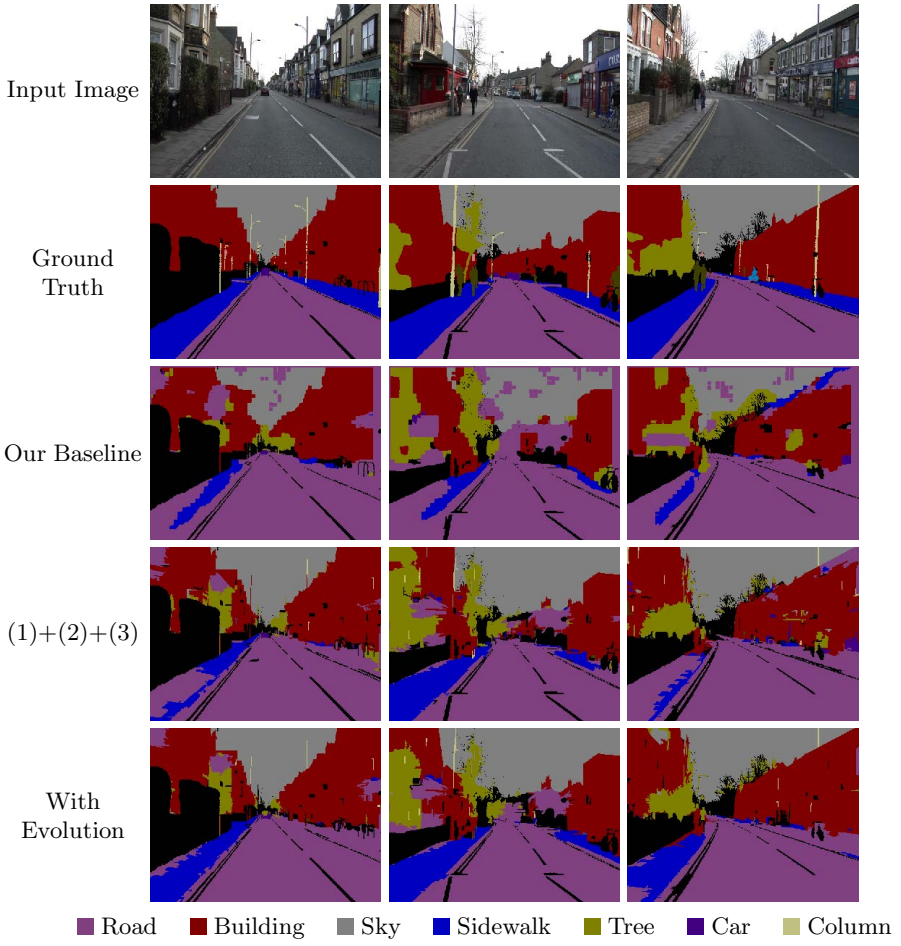

Fig. 3. Sample visual results of 3 images from experiments on CamVid dataset, in which we compared the baseline method and our methods, with or without candidate set evolution. The results show that low level segmentation helps to preserve class boundaries. Local label descriptors helps to recover thin structures such as lamp posts. Also, our continuous convex optimization is less prone to local minimum so that it produces more consistent results. The last row indicates that our candidate set evolution helps to resolve ambiguity between road and sidewalk that exists in the initial feature query. **Best viewed in color.**

documented in [1]. For efficiency reasons, we only enumerate 20 split hypothesis for each node instead of 500 as in [1]. In our current implementation, we used only HOG descriptors, but no color information as used in [1]. We sample testing patches less densely (our test patch stride = 6 while theirs = 1).

However, we believe our comparison is fair because all the comparisons use the same tree construction procedure, the same feature extraction, and the same

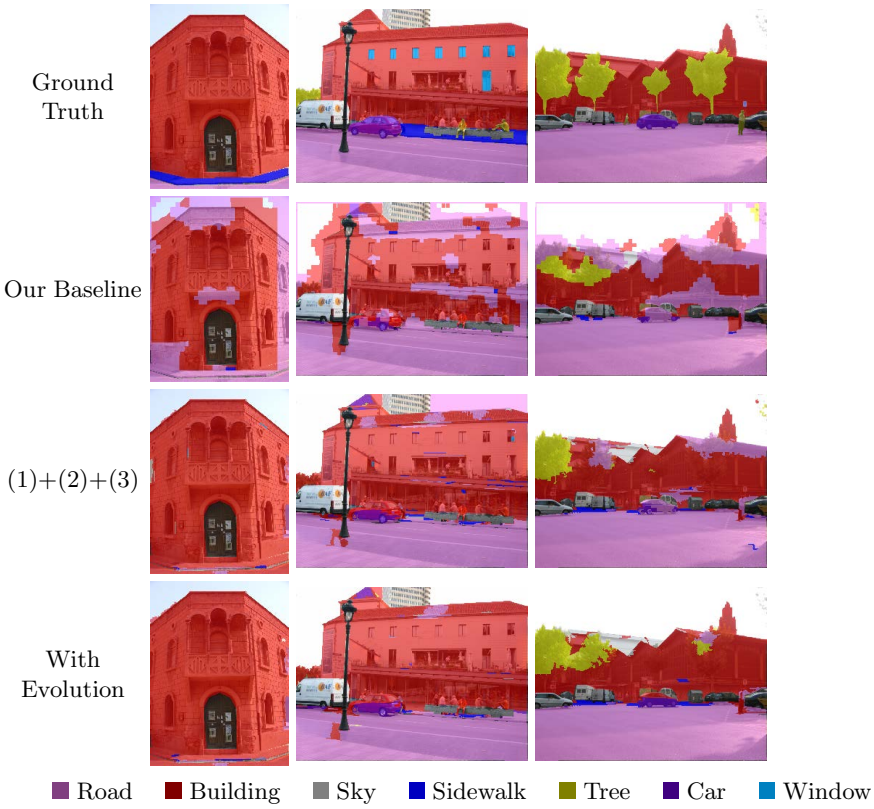

Fig. 4. Results on Barcelona dataset for baseline method and our methods, with and without evolution, for 3 testing images, from left to right. Compared to the baseline method, our method reveals more fine structures such as cars, and achieves better consistency in large objects, i.e. buildings, road, and sky. **Best viewed in color.**

cell and patch setup. We emphasize our primary goal is to show the improvement when label descriptors, convex relaxation, and candidate set evolution are used.

6.2 Experiments on Barcelona Dataset

We evaluated our methods on the Barcelona dataset, which is also used in [3]. This dataset originates from a subset of LabelMe, and is not as well labeled as CamVid (e.g. a car may be labeled as building). Unlike CamVid in which all the images come from video clips taken behind the windshield of a car, images in Barcelona dataset vary in perspective and locations. The testing set consists of a series of street view images from Barcelona. This dataset is originally labeled with 170 different semantic classes, from which we used 11 classes with top occurrence in the testing set, i.e. SKY, BUILDING, TREE, CAR, ROAD, PERSON, WINDOW, SIDEWALK, SIGN, WALL and MOTORBIKE. To form the training set for the forest, we selected 346 images in which the above 11

Table 2. Evaluation on Barcelona Dataset

Method	Global	Avg(Class)	Avg(Pascal)
Our Baseline	57.29	23.89	16.37
(1)+(2)+(3)	63.73	26.52	19.59
(1)+(2)+(3)+(4)	63.73	26.22	19.51

semantic classes account for more than 94% of the labeled pixels. In Table 2, we can see that our methods again show improvement over the baseline. Visual comparison of several example results are provided in Figure 4.

7 Conclusions

In this paper we introduce the concept of *local label descriptor*, which is a concatenation of label histograms for each cell in a patch. Local label descriptors alleviate the label patch misalignment issue in combining structured label predictions for semantic image labeling. Given an input image, we cast the semantic image labeling as a convex optimization problem, in which we solve for a label map whose local label descriptors can be approximated as a sparse convex combination of exemplar label descriptors in the training data, while the sparsity is regularized by the similarity measure between the local feature descriptor of the input image and that of the exemplars in the training data set. Our new formulation produces encouraging results on the Camvid and Barcelona datasets. In the future, we would like to (1) engineer better feature representations and tree construction procedures to have stronger baseline methods, (2) combine local descriptors with global label descriptors in [17], and (3) design better strategies for updating candidate sets.

Acknowledgement. This work is supported in part by NSF EFRI-BSBA-0937847, IIS-0845916, IIS-0916441, a Sloan Research Fellowship, a Packard Fellowship for Science and Engineering, and a gift donation from Eastman Kodak Company.

References

1. Kontschieder, P., Bulo, S., Bischof, H., Pelillo, M.: Structured class-labels in random forests for semantic image labelling. In: ICCV (2011)
2. Brostow, G.J., Shotton, J., Fauqueur, J., Cipolla, R.: Segmentation and Recognition Using Structure from Motion Point Clouds. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part I. LNCS, vol. 5302, pp. 44–57. Springer, Heidelberg (2008)
3. Tighe, J., Lazebnik, S.: SuperParsing: Scalable Nonparametric Image Parsing with Superpixels. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part V. LNCS, vol. 6315, pp. 352–365. Springer, Heidelberg (2010)

4. He, X., Zemel, R., Carreira-Perpinan, M.: Multiscale conditional random fields for image labeling. In: CVPR (2004)
5. Shotton, J., Winn, J.M., Rother, C., Criminisi, A.: *textonBoost*: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3951, pp. 1–15. Springer, Heidelberg (2006)
6. Kohli, P., Ladicky, L., Torr, P.: Robust higher order potentials for enforcing label consistency. In: CVPR (2008)
7. Galleguillos, C., Rabinovich, A., Belongie, S.: Object categorization using co-occurrence, location and appearance. In: CVPR (2008)
8. Ladicky, L., Russell, C., Kohli, P., Torr, P.H.S.: Graph Cut Based Inference with Co-occurrence Statistics. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part V. LNCS, vol. 6315, pp. 239–253. Springer, Heidelberg (2010)
9. Gonfaus, J., Boix, X., van de Weijer, J., Bagdanov, A., Serrat, J., Gonzandlez, J.: Harmony potentials for joint classification and segmentation. In: CVPR (2010)
10. Torralba, A., Murphy, K., Freeman, W., Rubin, M.: Context-based vision system for place and object recognition. In: ICCV (2003)
11. Rabinovich, A., Vedaldi, A., Galleguillos, C., Wiewiora, E., Belongie, S.: Objects in context. In: ICCV (2007)
12. Toyoda, T., Hasegawa, O.: Random field model for integration of local information and global information. TPAMI 30 (2008)
13. Ladický, Ľ., Sturgess, P., Alahari, K., Russell, C., Torr, P.H.S.: What, Where and How Many? Combining Object Detectors and CRFs. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part IV. LNCS, vol. 6314, pp. 424–437. Springer, Heidelberg (2010)
14. Russell, B.C., Torralba, A., Liu, C., Fergus, R., Freeman, W.T.: Object recognition by scene alignment. In: NIPS (2007)
15. Liu, C., Yuen, J., Torralba, A.: Nonparametric scene parsing via label transfer. TPAMI 33 (2011)
16. Socher, R., Lin, C.C.Y., Ng, A.Y., Manning, C.D.: Parsing natural scenes and natural language with recursive neural networks. In: ICML (2011)
17. Huang, Q., Han, M., Wu, B., Ioffe, S.: A hierarchical conditional random field model for labeling and segmenting images of street scenes. In: CVPR (2011)
18. Arora, S., Hazan, E., Kale, S.: The multiplicative weights update method: a meta algorithm and applications. Technical report, Princeton University (2005)
19. Kumar, N., Zhang, L., Nayar, S.K.: What Is a Good Nearest Neighbors Algorithm for Finding Similar Patches in Images? In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part II. LNCS, vol. 5303, pp. 364–378. Springer, Heidelberg (2008)
20. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Commun. ACM
21. Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.B.: PatchMatch: A randomized correspondence algorithm for structural image editing. ACM Transactions on Graphics, Proc. SIGGRAPH (2009)