Discriminative Indexing for Probabilistic Image Patch Priors

Yan Wang[1,⋆], Sunghyun Cho[2,⋆⋆], Jue Wang[2], and Shih-Fu Chang[1]

[1] Dept. of Electrical Engineering, Columbia University, USA
{yanwang,sfchang}@ee.columbia.edu
[2] Adobe Research, USA
sodomau@postech.ac.kr, juewang@adobe.com

Abstract. Newly emerged probabilistic image patch priors, such as Expected Patch Log-Likelihood (EPLL), have shown excellent performance on image restoration tasks, especially deconvolution, due to its rich expressiveness. However, its applicability is limited by the heavy computation involved in the associated optimization process. Inspired by the recent advances on using regression trees to index priors defined on a Conditional Random Field, we propose a novel discriminative indexing approach on patch-based priors to expedite the optimization process. Specifically, we propose an efficient tree indexing structure for EPLL, and overcome its training tractability challenges in high-dimensional spaces by utilizing special structures of the prior. Experimental results show that our approach accelerates state-of-the-art EPLL-based deconvolution methods by up to 40 times, with very little quality compromise.

1 Introduction

Image priors have been widely used in many ill-posed image restoration problems, such as deconvolution and denoising, to help resolve the ambiguity. One classic family of image priors is defined on image gradients, which assume that the magnitude of image gradients follows certain distributions such as exponential distributions [1], hyper Laplacian distributions [2], or a mixture of Gaussians [3]. These priors are computationally efficient using simple gradient filters and the Half-Quadratic Splitting optimization framework [2]. However, due to the extremely small spatial support of gradient filters, gradient priors cannot faithfully capture image structures.

To address this issue, image priors with larger spatial support have been proposed. One popular direction is to formulate the image restoration problem within a Conditional Random Field (CRF) framework, and associate nonadjacent pixels by connecting them in the field. Field of Experts (FoE) [4] as a typical example, constructs a CRF on all the pixels and define priors on the cliques of the CRF, with pixels within each local patch fully connected. While

⋆ This work was done when Yan Wang worked as an intern at Adobe Research.
⋆⋆ Sunghyun Cho is now with Samsung Electronics.

providing much larger spatial support than the gradient priors, the complicated field structure also suffers from the optimization tractability problem. To this end, approximate inference is often adopted, still resulting in slow speed.

While other challenges such as pixel saturation [5] and outliers [6] also exist in the image restoration field, the critical problem of optimization framework still remains open, and an emerging trend is to define probabilistic priors on image patches (i.e *probabilistic patch-based prior*), without explicit connections among pixels despite the natural pixel sharing between adjacent patches. An exemplar is Expected Patch Log-Likelihood (EPLL) [7], which shows state-of-the-art performance on image deblurring and competitive results on denoising and inpainting. These methods use Half-Quadratic Splitting for optimization which is more efficient than the inference of the CRFs. Unfortunately, they still require an excessive amount of computation which severely limits their practical usage. For example, the non-blind deconvolution method of Zoran and Weiss takes tens of minutes for an one megapixel image [7] on a decent PC. It becomes even worse for blind deconvolution, where the non-blind deconvolution component needs to be applied repeatedly and typically requires hours to finish [8].

Prior Indexing. The speed issue of the non-gradient priors is a well-known problem and various approaches have been proposed to address it. For the random-field-based priors, Jancsary *et al.* [9] restrict the potential functions of the CRF to be Gaussian functions for faster inference. To compensate for the performance drop from limiting forms of potential functions, regression trees are trained to discriminatively determine the mean and covariance of the potential functions, resulting in a Regression Tree Field formulation [10] that provides state-of-the-art performance for denoising and inpainting. This method can be interpreted as using random forests to pre-index a flexible prior defined on cliques in the random field. A similar idea of using pre-trained tree structures to efficiently construct a Regression Tree Field is also used in deblurring recently [11].

For the newly emerged probabilistic patch-based prior direction, however, little exploration has been done in expediting the associated optimization process, albeit such expedition can potentially benefit a series of practical applications and possibly reveal more insights about the patch-based priors. Inspired by the pre-indexing view of the Regression Tree Fields, we propose to pre-index the probabilistic patch-based priors to speed up their optimization. And we adopt EPLL as an example to demonstrate the novel prior indexing approach.

Challenges. However, indexing the patch-based priors is fairly challenging and the existing approaches are not readily extended to its unique settings. First, the image patches lie in a relatively high dimensional space. This makes straightforward lookup tables, as used in the hyper Laplacian prior [2], not able to work properly because of the huge memory consumption. Content-based hashing is known to be compact and fast, especially for high-dimensional data, but its accuracy is insufficient for image restoration tasks. Second, from the motivation of acceleration, we have a tight budget in the tree depth and the natural image patches have special structures different from the common distributions. Therefore as we will show shortly, existing tree indexing structures as used in [9][11] also suffer from the computational cost problem in our settings.

To address the challenges, we propose an efficient and compact tree indexing structure, whose training algorithm is specifically tailored for the patch distributions of natural images for efficient computation. Specifically, we observe that the EPLL prior can be well approximated with one single Gaussian in each optimization step, although such Gaussians may be different in each step. Therefore we propose to train a tree structure to efficiently determine the mean and covariance of the Gaussian, with a training algorithm similar to decision tree but using a more efficient candidate generation scheme.We take image deblurring as the primary application because of the state-of-the-art performance EPLL shows on it. Complexity analysis and experimental results show our indexing approach leads to significant acceleration, while preserving the power of patch-based priors. Qualitative experiments also demonstrate the potential of proposed indexing approach in deblurring real-life photos and image denoising.

Our main technical contributions include:

1. A novel framework of indexing patch-based natural image priors using decision trees (Section 3).
2. An efficient way of constructing the indexing tree by exploring the special structure of the parametric patch prior components (Section 4).

2 Observations and Our General Framework

2.1 Background and Notations

Before introducing our approach in more detail, we first provide a formal description of the problem. Image degradation is typically modeled as

$$\mathbf{y} = A\mathbf{x} + \mathbf{n}, \tag{1}$$

where $\mathbf{y}$, $\mathbf{x}$ are $\mathbf{n}$ are vectors representing an observed blurry image, its latent image to be recovered, and noise. For denoising, $A = I$, an identity matrix. For deconvolution, A is a convolution matrix.

The restored image $\hat{\mathbf{x}}$ can be estimated using Maximum A Posteriori (MAP) estimation, with a Gaussian likelihood function and Gaussian noise:

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} \left\{ \frac{\lambda}{2} \|\mathbf{y} - A\mathbf{x}\|^2 - \log p(\mathbf{x}) \right\}, \tag{2}$$

where λ is a parameter to control the restoration strength.

GMM based Patch Prior proposed by Zoran and Weiss [7] is defined as:

$$p(\mathbf{x}) \propto \prod_i p(\mathbf{x}_i) = \prod_i \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_i | \mu_k, \Sigma_k), \tag{3}$$

where i is a pixel index, and $\mathbf{x}_i$ is a patch centered at the i-th pixel. A Gaussian Mixture Model (GMM) $\{\mu_k, \Sigma_k, \pi_k\}_{k=1}^{K}$ is learned from a large collection of natural image patches, with k as the index of the Gaussian components, μ_k, Σ_k and π_k as the mean, covariance and weights of the Gaussians respectively.

Directly optimizing Equation (2) is difficult due to the coupling of the two terms. For efficient optimization, auxiliary variables $\{\mathbf{z}_i\}$ can be introduced as done in [7], reformulating Equation (2) with a popular half-quadratic scheme as:

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} \Big\{ \frac{\lambda}{2} \|\mathbf{y} - A\mathbf{x}\|^2 + \frac{\beta}{2} \sum_i \|\mathbf{z}_i - \mathbf{x}_i\|^2 - \sum_i \log p(\mathbf{z}_i) \Big\}, \tag{4}$$

The optimization starts from a small value of β, and develops by fixing $\mathbf{x}$ to solve for $\mathbf{z}$ (the z-step), and fixing $\mathbf{z}$ to solve for $\mathbf{x}$ (the x-step) alternatingly, with increasing β values. When β becomes large enough, the optimal $\hat{x}$ and $\hat{z}$ will be nearly the same with negligible difference.

Bottleneck. While the x-step can be computed quickly as the first and second terms in Equation (4) are quadratic, the z-step is a much slower optimization process. With fixed $\mathbf{x}$, the z-step tries to solve the following problems for all i:

$$\hat{\mathbf{z}}_i = \underset{\mathbf{z}_i}{\operatorname{argmin}} \left\{ \frac{\beta}{2} \|\mathbf{z}_i - \mathbf{x}_i\|^2 - \log \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{z}_i | \mu_k, \Sigma_k) \right\}, \tag{5}$$

which is a complex and expensive non-linear optimization problem involving a lot of matrix multiplications. To alleviate the optimization difficulty, Zoran and Weiss [7] only use the Gaussian component with the largest conditional likelihood $p(k|P_i\mathbf{x})$ instead of all the components to do the optimization. More specifically, with the chosen Gaussian $\hat{k}_i$, they solve the following simplified problem:

$$\hat{\mathbf{z}}_i = \underset{\mathbf{z}_i}{\operatorname{argmin}} \left\{ \frac{\beta}{2} \|\mathbf{z}_i - \mathbf{x}_i\|^2 - \log \mathcal{N}(\mathbf{z}_i | \mu_{\hat{k}_i}, \Sigma_{\hat{k}_i}) \right\}. \tag{6}$$

However, this approximation still needs a huge amount of computation. Specifically, to find $\hat{k}_i$ for the i-th patch, we need to compute

$$\begin{aligned} \arg\max_k p(k|\mathbf{x}_i) &\propto p(\mathbf{x}_i|k)p(k) \\ &= \int_{\mathbf{z}_i} p(\mathbf{x}_i|\mathbf{z}_i)p(\mathbf{z}_i|k)p(k) \\ &= \int_{\mathbf{z}_i} \mathcal{N}(\mathbf{x}_i|\mathbf{z}_i, \beta^{-1}I)\mathcal{N}(\mathbf{z}_i|\mu_k, \Sigma_k)\pi_k \\ &= \pi_k \mathcal{N}(\mathbf{x}_i|\mu_k, \beta^{-1}I + \Sigma_k) \end{aligned} \tag{7}$$

for *all the K Gaussian components*, resulting in $2K$ expensive matrix multiplication operations *for every patch.*

That is, the bottleneck of EPLL lies in using the naive linear scan to solve the optimization problem in Equation (7).

2.2 Observations and Our Approach

Given that Equation (7) is a discrete optimization problem for which efficient gradient-based methods cannot be applied, we propose to use a discriminative tree structure to output an approximate $\hat{k}_i$ directly based on a series of simple operations. This can also be interpreted from an information retrieval perspective, i.e. instead of an exhaustive scan on all the candidates, we use a series of quick tests to determine the rough area in the feature space that the patch lies in, and directly adopt the corresponding $\hat{k}_i$ as the approximate solution. Another interpretation from a machine learning perspective is, it is equivalent to treating Equation (7) as a classification problem and using a discriminative classifier to directly predict the most likely class.

One immediate concern one may have is that this is only an approximated solution, which may affect the quality of the restored image. However, our experiments show that with properly constructed indexing trees, we can achieve a high approximation accuracy in real applications. Furthermore, as we will show shortly, for the patches that are harder to be classified properly, on which error is more likely to be introduced, the value of final $\hat{k}_i$ actually has less effects on the quality of the restored patch $\hat{\mathbf{z}}_i$. Therefore, such indexing on the patch-based priors can provide significant acceleration with very little quality compromise.

Now the question is: is it possible to build an efficient index for dominant Gaussian identification? Fortunately the patch-based prior has its special structures which allow us to further improve its efficiency. If we take a closer look at the GMM learned from natural image patches, most Gaussian components have very elongated shapes, *i.e.* Σ_k has only a few large eigenvectors, and they do not overlap each other much except for small parts [12]. This leads us to believe that it may be possible to use a hierarchy of simple classifiers, *e.g.* linear classifiers, to break the high-dimensional space to different subspaces that belong to different Gaussians. One subsequent concern is that since all the mixture components share the same center which is the origin as observed in [7], such overlap may confuse the linear classifiers. We found this is not a big deal because what the optimizer in Equation (6) does is to push the patch a bit to the center along the path determined by the dominant Gaussian, and it makes little difference when the patch is already close to the shared center even if a wrong Gaussian is identified and used for it.

Our Approach. We thus propose to build a decision tree based on linear classifiers to index the dominant Gaussian components in EPLL. However, traditional decision tree algorithms cannot be directly applied here because its random generation scheme of classifier candidates is extremely inefficient in a high-dimensional space. To make the training process more stable and efficient, we utilize the structure of the GMM and overcome the challenges of candidate classifier generation with a Gibbs sampling approach. A filter-based fast inference of Markov Random Field [13] is also employed to improve indexing accuracy.

3 Index-Assisted Patch Prior Optimization

In this section we assume the decision tree has already been built, and describe how to quickly find the most dominant component $\hat{k}_i$, which gives the largest $p(k|\mathbf{x}_i)$ for a given noisy patch $\mathbf{x}_i$. Specifically, from a given noisy patch $\mathbf{x}_i$, the search process goes from the root of the tree to one of the leaves. Each non-leaf node in the tree has a linear classifier $\mathbf{sgn}(\mathbf{w}^T\mathbf{x}+b)$, determining where $\mathbf{x}_i$ goes in the next level as follows:

$$\text{Next}(\mathbf{x}_i|\mathbf{w},b) = \begin{cases} \text{Left child} & \mathbf{w}^T\mathbf{x}_i + b \geq 0, \\ \text{Right child} & \mathbf{w}^T\mathbf{x}_i + b < 0. \end{cases} \tag{8}$$

While traversing the tree from its root to a leaf node, the space of patches is recursively bisected by the linear classifiers, ending with a polyhedron $L_i = \{\mathbf{x}|W_i\mathbf{x} + B_i \leq 0\}$, with W_i and B_i determined by the traversal path of $\mathbf{x}_i$. We store the expected probability of each Gaussian component dominating a random point within this polyhedron $\phi_{ik} = \mathbb{E}_{\mathbf{x}\in L_i}[\text{Prob}(\hat{\mathcal{N}}(\mathbf{x}) = \mathcal{N}_k)]$ in the leaf node, and then use it to approximate the probability of $\mathbf{x}_i$ having $\mathcal{N}_k$ as the dominant Gaussian.

Note that this tree testing process is very efficient. First, each linear classifier only requires a dot product operation. Second, only a few levels of tree nodes (e.g. 12 levels) are enough for reasonable accuracy in practice. This makes it even faster than the hashing-based approaches which typically require more than 20 bits for reasonable accuracy.

Refinement Using MRFs. To find the dominant Gaussian $\hat{k}_i$ for a given patch $\mathbf{x}_i$, instead of the winner-take-all selection of k with the largest ϕ_{ik}, which will introduce possibly large errors, we use a discrete Markov Random Field (MRF) to infer the final $\hat{k}_i$s, with the enforcement on the spatial consistency in terms of the dominant Gaussians. Specifically, the potential function of the MRF is defined as:

$$\Psi(\{\hat{k}_i\}) = \lambda_1 \sum_i \Psi_1(\hat{k}_i) + \lambda_2 \sum_{\text{Neighbors } i,j} \Psi_2(\hat{k}_i, \hat{k}_j) \tag{9}$$

where $\Psi_1(\hat{k}_i) = -\phi_{i\hat{k}_i}$, and $\Psi_2(\hat{k}_i, \hat{k}_j)$ is defined as:

$$\Psi_2(\hat{k}_i, \hat{k}_j) = \begin{cases} 0 & \text{if } \hat{k}_i = \hat{k}_j \\ |I_i - I_j|^2 & \text{otherwise} \end{cases} . \tag{10}$$

I_i and I_j are the average intensities of the patches $\mathbf{x}_i$ and $\mathbf{x}_j$, respectively. Equation (9) is minimized to find the refined dominant Gaussians $\{\hat{k}_i\}$ for all the patches, where we adopt an approximation approach, cost-volume filtering [13]. Similarly to Loopy Belief Propagation, the cost-volume filters update the marginal distribution stored in each node. However, instead of message collection and passing, such updates are performed with the guided filter [14], which is accelerated by integral images and extremely fast. More specifically, K "images" with intensities as ϕ_{ik} are first collected, and the guided filter is applied on every "image", with the smoothed input from the x-step as the guidance.

Wiener Filtering. Once the dominant mixture component $\hat{k}_i$ for each patch $\mathbf{x}_i$ is found, it is fed to the optimizer for Equation (6), which has a close form solution as the Wiener filter:

$$\hat{\mathbf{z}}_i = (\Sigma_{\hat{k}_i} + \sigma^2 I)^{-1}(\Sigma_{\hat{k}_i}\mathbf{x}_i + \sigma^2 I\mu_{\hat{k}_i}). \tag{11}$$

Time Complexity. Given an index tree with depth D for a K-component GMM defined on $n \times n$ patches, since on each level we only need to apply one dot product, the tree traversal for each patch requires $O(n^2D)$ operations. The cost-volume filtering needs $O(K)$ time for each patch. Therefore the overall time complexity is $O(mn^2D + mK)$ for an image with m patches, with a very small coefficient for $O(K)$, which is from the guided filter. In contrast, the original EPLL needs $O(mn^4K)$ time.

4 Prior Index Construction

Given an observed patch $\mathbf{x}$, we expect the trained index tree to output the k which approximately maximizes $p(k|\mathbf{x})$ (Equation (7)). As $p(k|\mathbf{x})$ also depends on β, which is the pre-defined parameter for the alternating optimization, we build different index trees with respect to different values of β , with the algorithm introduced in this section.

Although the testing phase of our index tree is similar to a decision tree, the training algorithm of decision trees cannot be directly applied here. In the decision tree training algorithm, given a set of training examples, many *classifier candidates* are *randomly* generated, each of which will divide the training examples into two partitions. And then the best classifier with the largest *information gain* computed from the partitions will be selected and stored in the node. This can be viewed as a naive optimizer randomly searching for the classifier with the largest information gain. When incorporated with linear classifiers, this works fine on low-dimension data. However, with the increase of the dimensionality, the feasible space to search is expanding much faster than the small space where the good solutions lie. This leads to the failure of the naive random search optimizer when the dimensionality of $\mathbf{x}$ is not trivially small, *i.e.* a huge number of trials are required before it reaches the optimal or even near-optimal solutions.

To demonstrate such inefficiency of the random search scheme, we collect two million 8×8 training patches with ground truth labels of the dominant Gaussian. The traditional decision tree training algorithm is applied on the dataset, with 1000 candidates randomly generated for every node. It takes 48 hours to obtain a 12-level tree on a Core i7 3.0GHz desktop computer with a MATLAB implementation, and we plot the average entropy of each level as the green curve in Figure 1. From the figure, we can see that even after 12 levels, the average entropy is still close to 0.6, indicating the distributions in the leaf nodes are still not far away from uniform and contains not much information. Given we have a high expectation on the testing speed thus a tight budget on the tree depth, decision tree training algorithm actually does not fit our problem settings.

To mitigate this challenge, we exploit the special structure of the GMM learned from natural image patches, and formulate the candidate classifier generation as an optimization problem coupled with random sampling. The recursive greedy training framework of the decision tree is still used in our approach due to its simplicity and robustness. In the following paragraphs, we will discuss each step of our training process in more detail.

Training Data Generation. To train an index tree for a given β, we collect a set of noisy patches $\{\mathbf{x}\}$ from the output of the x-steps of EPLL [7] as the X for training because that is the input our index will face in real applications. The ground truth labels Y are then determined with Equation (7). While there is no theoretical clue about how many training examples are "enough", we will revisit this step in Section 5 for the practical concern of the size of the training dataset.

Candidate Classifier Generation. Given a set of noisy patches X and the ground truth labels Y, the problem we are facing is to find a linear classifier $\mathbf{sgn}(\hat{\mathbf{w}}^T\mathbf{x} + \hat{b})$ so that the information gain is maximized:

$$\hat{\mathbf{w}}, \hat{b} = \underset{\mathbf{w},b}{\operatorname{argmax}} E(Y) - \frac{|Y_+|}{|Y|}E(Y_+) - \frac{|Y_-|}{|Y|}E(Y_-), \tag{12}$$

in which $E(\cdot)$ is the entropy function, and Y_+ and Y_- are the positive and negative partitions divided by the classifier:

$$\begin{aligned} Y_+ &= \{y_i | \mathbf{w}^T\mathbf{x}_i + b \geq 0, \ \forall i\}, \quad \text{and} \\ Y_- &= \{y_i | \mathbf{w}^T\mathbf{x}_i + b < 0, \ \forall i\}. \end{aligned} \tag{13}$$

It has been shown that naive random search does not work for high dimensional $\mathbf{x}$. Given this problem is non-differentiable with the discrete training examples, we do not use classical continuous optimization methods such as gradient descent or BFGS. Instead, we adopt a Gibbs sampling approach, while some heuristics are introduced to restrict the space from which the candidates are generated.

There are two important observations of the learned GMM. First, for most components, only a few strongest eigenvectors of the covariance matrix take the most energy of the Gaussian. This indicates it is possible to dramatically reduce the computation complexity by only doing sampling based on these a few strong *principal directions*, which are the eigenvectors of the Gaussians' covariance matrices. Second, all of the Gaussian components share the same center which is the origin, as observed in [7]. This can be explained with the inherent symmetry of the natural image patches.

These two properties inspire us a simple heuristic to generate a classifier candidate for two principal directions from two Gaussians. Take Figure 2 for an example, if two 2-D Gaussians are given with the two principal directions $\mathbf{e}_1$ and $\mathbf{e}_2$ marked as red, a reasonable guess of the decision (hyper)plane would be

$$\mathbf{w} = \lambda_1\mathbf{e}_1 - \lambda_2\mathbf{e}_2, \tag{14}$$

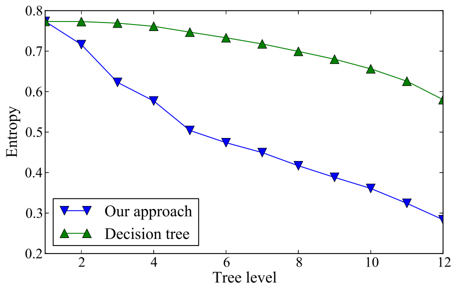


Fig. 1. Comparison of how the entropy decreases in different levels of the index tree, with different training schemes. The traditional decision tree training algorithm is plotted in green, with the proposed approach in blue.

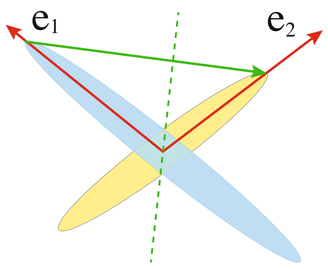


Fig. 2. A toy example of the proposed heuristic to generate a candidate classifier from two given principal directions $\mathbf{e}_1, \mathbf{e}_2$. The dashed green line shows the generated classifier when $b = 0$, with the green arrow as its normal vector.

in which λ_1, λ_2 are the corresponding eigenvalues of $\mathbf{e}_1, \mathbf{e}_2$, as the green arrow shows. Note $-\mathbf{e}_1$ and $-\mathbf{e}_2$ are also the principal directions. Therefore this scheme will actually generate four $\mathbf{w}$-s.[1]

With this candidate generation scheme, the problem turns to how to sample the principal directions such that we can partition the training data "effectively". With the expectation of minimizing the tree depth with a target accuracy, we add a balance factor to the objective function in Equation (12). More specifically, we expect the positive and negative examples predicted by the classifier $\mathbf{sgn}(\hat{\mathbf{w}}^T\mathbf{x} + \hat{b})$ is roughly the same in number. Given this is hard to optimize, we further relax it to expect the average projection values to be as small as possible. Then the objective function becomes,

$$E(Y) - \frac{|Y_+|}{|Y|}E(Y_+) - \frac{|Y_-|}{|Y|}E(Y_-) - \gamma \left| \sum_{\mathbf{x}} \left(\mathbf{w}^T\mathbf{x} + b \right) \right|, \qquad (15)$$

$$\text{s.t. } \|\mathbf{w}\|^2 = 1 \text{ generated from Equation (14).}$$

Here $\gamma = 0.5$ is a parameter controlling the strength of the balance factor.

Note both the terms in Equation (15), the information gain and the balance factor, would only change when some example $\mathbf{x}_i$ changes its predicted label. That is, it will change faster if the $\mathbf{w}$ swipes along some high-density area with more training examples, while slower in the low-density areas. Therefore it is reasonable to sample more $\mathbf{w}$-s from the regions with low GMM probabilistic densities, which are the analogy to stationary points in the continuous case. More specifically, we put more priority in sampling the decision boundaries between

[1] One classifier may not be able to distinguish the two Gaussians shown in Figure 2. But a simple two-level decision trump from the four candidate classifiers would have enough discrimination power.

Algorithm 1. Index construction for patch priors.

```
Input: the patch prior $\{\pi_k, \mu_k, \Sigma_k\}_{k=1}^{K}$, training examples $X$, the ground truth
       labels $Y$, and the max tree depth $D$
Output: a decision tree $T$ based on linear classifiers
1  if $D = 0$ then
2      return a leaf node with label distribution of $Y$.
3  end
4  foreach $1 \leq I \leq I_{max}$ do
5      Sample two Gaussians $k_1, k_2$ without replacement with probability
       $p(k) = \pi_k$.
6      Given each Gaussian $k$ from $k_1, k_2$ , sample one eigenvector from the
       eigenvectors of the covariance matrix $\{\mathbf{e}_{ki}\}$ with probability $p(i|k) = \lambda_{ki}$,
       where $\lambda_{ki}$ are the corresponding eigenvalues.
7      Use Equation (14) to generate $\mathbf{w}$-s given the two eigenvectors $\mathbf{e}_1, \mathbf{e}_2$.
8      Sample $b$ from $\mathcal{N}(0, 1)$.
9  end
10 Collect all the candidate $\mathbf{w}$ and $b$, store the one maximizing Equation (15) in
   the tree node $T$.
11 Train the left and right child of $T$ with $(D - 1)$ tree depth and $Y_+, Y_-$ as
   training data, which are defined in Equation (13).
12 return T.
```

two principal directions with large eigenvalues. That is, given the weights of the Gaussians $\{\pi_k\}$, we first sample two Gaussians with probability $p(k) = \pi_k$, and then sample one principal direction from the eigenvectors of the covariance matrix of each Gaussian $\{\mathbf{e}_{ki}\}$ with corresponding eigenvalues as the probability $p(i|k) = \lambda_{ki}$. After that, Equation (14) is applied on the principal directions to obtain the final $\mathbf{w}$-s, which forms a Gibbs sampling process. Since all the Gaussians share the same center as the origin, we use $\mathcal{N}(0, 1)$ to sample the b-s.

A complete algorithm is illustrated in Algorithm 1. We apply the proposed approach to the same data in the experiment shown in Figure 1, and obtain much better training efficiency, with average entropy below 0.3 in the 12th level, which is plotted as the blue curve in Figure 1. This proves the effectiveness of our training scheme, and in the next section, we will do more justification on our approach, followed by the evaluations on actual applications.

5 Experiments

We conduct a series of experiments to quantitatively verify (1) how well the discriminative prior indexing performs for dominant Gaussian identification; and (2) how well the proposed approach performs on real applications in terms of quality and speed. In this section we first quantitatively evaluate the proposed method in non-blind image deblurring, and then justify its components, especially on the performance of domainant Gaussian identification. Other applications including deblurring real-life photos and denoising are also demonstrated.

5.1 Evaluation on Non-blind Image Deblurring

Dataset and Evaluation Protocol. We use the standard benchmark [15], which contains 48 blurry photos and 12 motion kernels collected from real life for the evaluation. Different deblurring approaches are applied on the input images, and average PSNRs among all the kernels on each image are reported as quantitative measurements. We compare our deblurring approach with tree-based indexing with several state-of-the-art algorithms, including Discriminative non-blind deblurring [11] (referred as Schmidt), ℓ_0 based deblurring [16] (referred as Xu), and Cho's fast deblurring [17] (referred as Cho).

Implementation Details. We collect two million patches from 100 training images from the Berkeley Segmentation Dataset [18], convolve them with one blur kernel different from all the testing kernels, and add Gaussian noise to obtain the training data. Then an index tree with 12 levels is trained for each β in Half-Quadratic Splitting is trained using Algorithm 1 for our deblurring approach. As observed in [12], increasing the component number of the GMM hardly improves EPLL's performance after it reaches 10. Subsequently we adopt a 10-component GMM as the prior for both our approach and the EPLL baseline.

We implement our algorithm in MATLAB, with the core components such as the index tree testing written in C++. We further integrate Fast Fourier Transform to accelerate the x-step [1], resulting in a comprehensive fast non-blind deblurring algorithm, whose running time is adopted as the time of our approach. All the running time is measured on a desktop computer with a Core i7 3.0GHz CPU.

Results and Discussions. The average PSNRs of all approaches are shown in Table 1. Ours_C shows our PSNRs based on the kernels estimated from Cho's approach [17], and Ours_X is based on Xu's kernel [16]. We can see our non-blind deblurring component improves the performance of both Cho's and Xu's approaches in most cases. Although our PSNR is slightly worse than Schmidt's approach [11], the running time per each RGB image is 2 minites in average, which is about 20 times faster than [11][2] and 40 times faster than EPLL. Also note that this is achieved when the blur kernel for index construction is dramatically different from the blur kernels used in the test images. It suggests that our index construction is not sensitive to the blur kernel used for training.

5.2 Evaluation on Prior Indexing and Parameter Tuning

We also evaluate the performance of our prior indexing in terms of component identification accuracy. With the same training data and training algorithm in Section 5.1, we vary the depth of the decision trees to explore how it affects the indexing performance. The classification accuracy of the dominant Gaussian is calculated with ground truth from brute-force search, and is averaged on all the stages and all the test images as the evaluation protocol.

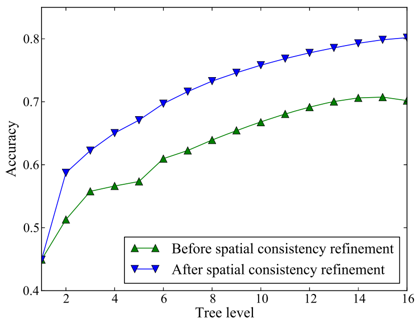
[2] The authors of [11] didn't report the running time on [15], but on smaller images. We project their running time to [15] based on the (linear) time complexity on resolution.

Table 1. Average PSNRs of each testing image on non-blind deblurring. Ours$_C$ and Ours$_X$ indicate our non-blind deblurring approach based on kernels estimated from Cho and Xu.

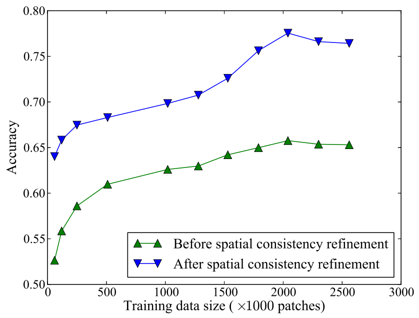
Img	1	2	3	4
Cho [17]	30.61	26.03	31.32	27.98
Xu [16]	31.64	26.64	31.45	28.42
Schmidt [11]	32.05	26.99	32.13	28.90
Ours$_C$	30.75	26.12	32.28	28.00
Ours$_X$	31.69	26.68	32.31	28.65

Table 2. Quantitative evaluation results on image denoising. The PSNR in dB is shown for each baseline and noise level (σ) setting. The average running time (in seconds) is shown in the rightmost column.

σ	0.1	0.25	0.5	1.0	Time
BM3D[19]	30.33	26.92	23.91	17.85	4.4
BM3D$_S$ [20]	30.46	26.62	23.22	19.73	782
K-SVD$_G$ [21]	29.39	25.57	22.68	19.31	60.1
K-SVD$_I$ [21]	29.76	25.68	22.70	19.38	177.7
EPLL [7]	29.57	26.13	23.44	20.62	61.7
Our approach	29.47	26.08	23.49	20.62	4.5



(a) Component identification accuracy along with different tree depth.



(b) Component identification accuracy along with training data size.

Fig. 3. Quantitative evaluation on the dominant Gaussian identification

The results with different tree depth settings are plotted in Figure 3(a), where we also show the classification accuracy before and after the spatial consistency refinement step. Firstly, it shows that the identification accuracy reaches 80% with 16 levels of tree nodes. Given we have 10 components in the GMM, this proves that the tree index does a reasonable job in approximating the brute-force search with merely a few dot product operations. Considering the trade-off between quality and efficiency, we use 12 level trees in all the other experiments. In addition, it also suggests that the cost-volume based MRF inference improves the identification accuracy by 10% consistently over the raw identification results. This verifies our observation on the spatial coherence of the distributions of dominant Gaussians.

Training Data Collection. With the same training and testing image sets, we also explore how many training patches are required for achieving reasonable quality of dominant Gaussian identification. Figure 3(b) plots the identification accuracy against different training data sizes. It suggests that with a 12-level tree, the accuracy saturates after the training dataset reaches two million patches.

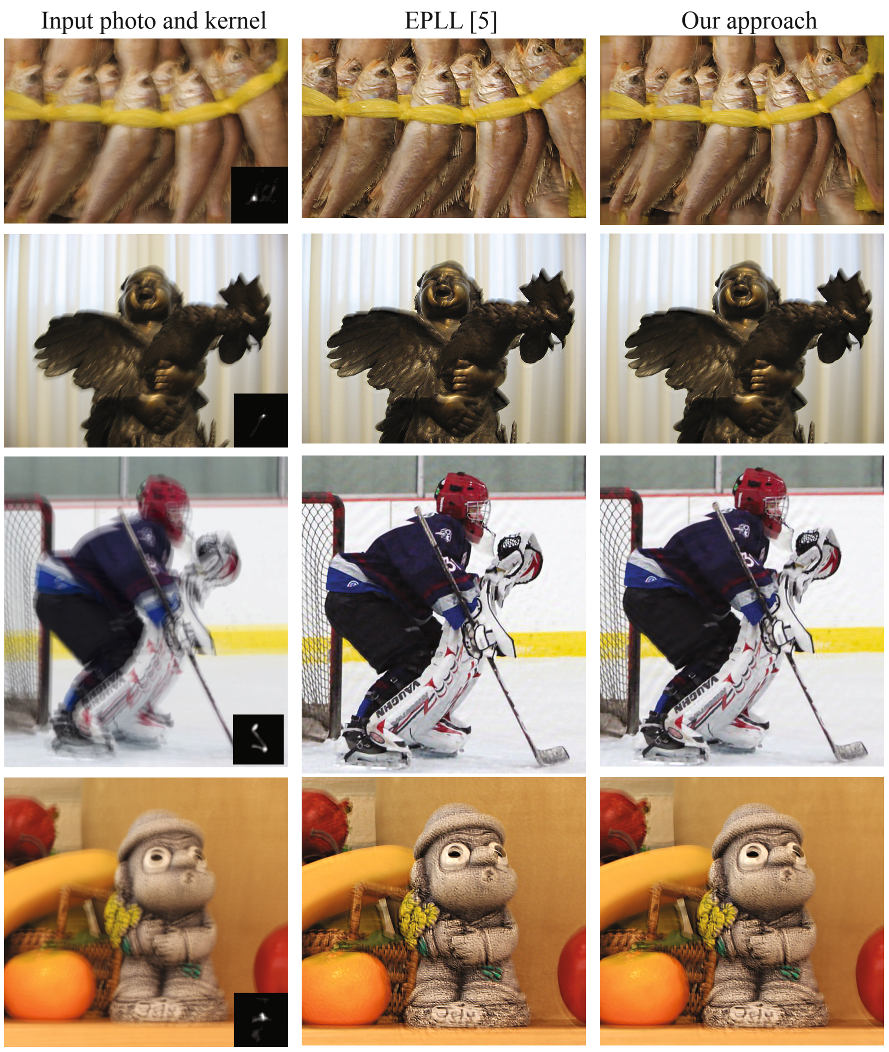


Fig. 4. Qualitative evaluation on deblurring high-resolution photos from real life. The input with the motion kernel estimated with [17], the deblurring results of EPLL and the proposed approach are shown from left to right.

We thus use this setting for all the experiments, including the non-blind image deblurring, deblurring high-reslution photos and image deblurring.

5.3 Deblurring High-Resolution Photos from Real Life

To demonstrate the capability to handle real-life data of our deblurring approach, we collect some blurred photos taken from real life, run [17]'s approach to estimate a blur kernel, and then apply the proposed algorithm on the R, G, B channels seperately. While all the collected photos have resolution larger than 800×800, EPLL [7] needs more than half an hour to deblur each image, and therefore is not practical for deblurring applications in real life. On the other hand, our algorithm generally outputs the result within 3 minutes. Figure 4 shows a comparison between the results from our approach and EPLL. From the figure, we can see that the proposed approach is able to achieve deblurring results with nearly unnoticable difference from the original patch-based approaches.

5.4 Evaluation on Image Denoising

To demonstrate the potential of the proposed approach in other low-level vision applications, we also report the performance on denoising. We use the standard benchmark in denoising, eight 512×512 gray-scale standard test images *Babara, Boat, Cameraman, Hill, House, Lena, Man* and *Peppers* for this evaluation. Gaussian noise with standard variance as 0.1, 0.25, 0.5 and 1 is added to the original images respectively as the noisy inputs. Average PSNR as well as the running time for all the images are measured for different noise levels. We compare our approach with the state-of-the-art denoising algorithms BM3D[19], BM3D-SAPCA[20] (referred as BM3D$_S$), K-SVD[21] with global dictionary (referred as K-SVD$_G$) and learned dictionary from the noisy image (referred as K-SVD$_I$), and EPLL[7], with the authors' implementations and recommended parameters. The quantitative results are reported in Table 2. The results show that the performance of EPLL is slightly worse than BM3D and BM3D-SAPCA, which is reasonable given that the latter two are specially designed for denoising. Our approach achieves very similar performance to EPLL, with < 0.1dB PSNR drop on average, but is much faster than EPLL and other denoising methods except BM3D. We further confirmed that there are no noticeable differences between our and EPLL's results.

6 Conclusion

We have presented an indexing method to improve the efficiency of applying patch-based image priors to image restoration tasks. We show that directly applying the traditional decision tree training algorithm is not optimal in our case due to the high dimensionality of the patch data. We therefore propose a training algorithm with a novel classifier candidate generation scheme utilizing the structure of the patch prior. Experimental results show that our approach achieves up to 40 times acceleration, and at the same time comparable high quality results with the original EPLL approach. The performance is also competitive with other state-of-the-art deconvolution algorithms.

There are also several interesting directions for future exploration, such as how to analytically construct the index solely from the prior model and how to apply the index to other vision problems.

References

1. Yang, J., Zhang, Y., Yin, W.: An efficient TVL1 algorithm for deblurring multichannel images corrupted by impulsive noise. Journal on Scientific Computing 31(4) (2009)
2. Krishnan, D., Fergus, R.: Fast image deconvolution using hyper-laplacian priors. In: NIPS (2009)
3. Fergus, R., Singh, B., Hertzmann, A., Roweis, S., Freeman, W.T.: Removing camera shake from a single photograph. In: ToG (SIGGRAPH) (2006)
4. Roth, S., Blacky, M.: Fields of experts. IJCV 82(2), 205–229 (2009)
5. Whyte, O., Sivic, J., Zisserman, A.: Deblurring shaken and partially saturated images. In: Proceedings of the IEEE Workshop on Color and Photometry in Computer Vision, with ICCV 2011 (2011)
6. Cho, S., Wang, J., Lee, S.: Handling outliers in non-blind image deconvolution. In: ICCV (November 2011)
7. Zoran, D., Weiss, Y.: From learning models of natural image patches to whole image restoration. In: ICCV (2011)
8. Sun, L., Cho, S., Wang, J., Hays, J.: Edge-based blur kernel estimation using patch priors. In: ICCP (2013)
9. Jancsary, J., Nowozin, S., Rother, C.: Loss-specific training of non-parametric image restoration models: A new state of the art. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part VII. LNCS, vol. 7578, pp. 112–125. Springer, Heidelberg (2012)
10. Jancsary, J., Nowozin, S., Sharp, T., Rother, C.: Regression Tree Fields - an Efficient, Non-Parametric Approach to Image Labeling Problems. In: CVPR (2012)
11. Schmidt, U., Rother, C., Nowozin, S., Jancsary, J., Roth, S.: Discriminative non-blind deblurring. In: CVPR (2013)
12. Zoran, D., Weiss, Y.: Natural images, gaussian mixtures and dead leaves. In: NIPS (2012)
13. Rhemann, C., Hosni, A., Bleyer, M., Rother, C., Gelautz, M.: Fast cost-volume filtering for visual correspondence and beyond. In: CVPR (2011)
14. He, K., Sun, J., Tang, X.: Guided image filtering. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part I. LNCS, vol. 6311, pp. 1–14. Springer, Heidelberg (2010)
15. Köhler, R., Hirsch, M., Mohler, B., Schölkopf, B., Harmeling, S.: Recording and playback of camera shake: benchmarking blind deconvolution with a real-world database. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part VII. LNCS, vol. 7578, pp. 27–40. Springer, Heidelberg (2012)
16. Xu, L., Zheng, S., Jia, J.: Unnatural l0 sparse representation for natural image deblurring. In: CVPR (June 2013)
17. Cho, S., Lee, S.: Fast motion deblurring. ToG (SIGGRAPH ASIA) 28(5) (2009)
18. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: ICCV (2001)
19. Dabov, K., Foi, A., Katkovnik, V., Egiazarian, K.: Image denoising by sparse 3D transform-domain collaborative filtering. TIP (8) (August 2007)
20. Dabov, K., Foi, A., Katkovnik, V., Egiazarian, K.: BM3D image denoising with shape-adaptive principal component analysis. In: SPARS (2009)
21. Elad, M., Aharon, M.: Image denoising via sparse and redundant representations over learned dictionaries. TIP 15(12), 3736–3745 (2006)