From Learning Models of Natural Image Patches to Whole Image Restoration

Daniel Zoran
Interdisciplinary Center for Neural Computation
Hebrew University of Jerusalem
daniez@cs.huji.ac.il

Yair Weiss
School of Computer Science and Engineering
Hebrew University of Jerusalem
http://www.cs.huji.ac.il/~yweiss

Abstract

Learning good image priors is of utmost importance for the study of vision, computer vision and image processing applications. Learning priors and optimizing over whole images can lead to tremendous computational challenges. In contrast, when we work with small image patches, it is possible to learn priors and perform patch restoration very efficiently. This raises three questions - do priors that give high likelihood to the data also lead to good performance in restoration? Can we use such patch based priors to restore a full image? Can we learn better patch priors? In this work we answer these questions.

We compare the likelihood of several patch models and show that priors that give high likelihood to data perform better in patch restoration. Motivated by this result, we propose a generic framework which allows for whole image restoration using any *patch based prior for which a MAP (or approximate MAP) estimate can be calculated. We show how to derive an appropriate cost function, how to optimize it and how to use it to restore whole images. Finally, we present a* generic*, surprisingly simple Gaussian Mixture prior, learned from a set of natural images. When used with the proposed framework, this Gaussian Mixture Model outperforms all other generic prior methods for image denoising, deblurring and inpainting.*

1. Introduction

Image priors have become a popular tool for image restoration tasks. Good priors have been applied to different tasks such as image denoising [1, 2, 3, 4, 5, 6], image inpainting [6] and more [7], yielding excellent results. However, learning good priors from natural images is a daunting task - the high dimensionality of images makes learning, inference and optimization with such priors prohibitively hard. As a result, in many works [4, 5, 8] priors are learned over small image *patches*. This has the advantage of making computational tasks such as learning, inference and likelihood estimation much easier than working with whole images directly. In this paper we ask three questions: **(1)** Do patch priors that give high likelihoods yield better *patch* restoration performance? **(2)** Do patch priors that give high likelihoods yield better *image* restoration performance? **(3)** Can we learn better patch priors?

2. From Patch Likelihoods to Patch Restoration

For many patch priors a closed form of log likelihood, Bayesian Least Squares (BLS) and Maximum A-Posteriori (MAP) estimates can be easily calculated. Given that, we start with a simple question: Do priors that give high likelihood for natural image patches also produce good results in a restoration task such as denoising? Note that answering this question for priors of whole images is tremendously difficult - for many popular MRF priors, neither the log likelihood nor the MAP estimate can be calculated exactly [9].

In order to provide an answer for this question we compare several popular priors, trained over 50,000 8×8 patches randomly sampled from the training set of [10] with their DC removed. We compare the log likelihood each model gives on a set of *unseen* natural image patches (sampled from the test set of [10]) and the performance of each model in patch denoising using MAP estimates. The models we use here are: Independent pixels with learned marginals (Ind. Pixel), Multivariate Gaussian over pixels with learned covariance (MVG), Independent PCA with learned (non-Gaussian) marginals and ICA with learned marginals. For a detailed description of these models see the Supplementary Material.

The results for each of the models can be seen in Figure 1. As can be seen, the higher the likelihood a model gives for a set of patches, the better it is in denoising them when they are corrupted.

3. From Patch Likelihoods to Whole Image Restoration

Motivated by the results in Section 2, we now wish to answer the second question of this paper - do patch priors that give high likelihoods perform better in *whole image*

2011 IEEE International Conference on Computer Vision
978-1-4577-1102-2/11/$26.00

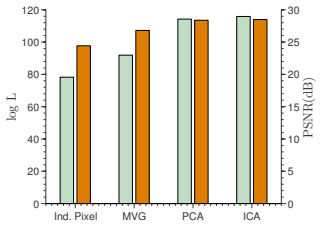


Figure 1: The likelihood of several off-the-shelf patch priors, learned from natural images, along with their *patch* denoising performance. As can be seen, patch priors that give higher likelihood to the data give better *patch* denoising performance (PSNR in dB). In this paper we show how to obtain similar performance in *whole image* restoration.

restoration? To answer this question we first need to consider the problem of how to use patch priors for whole image restoration.

To illustrate the advantages and difficulties of working with patch priors, consider Figure 2. Suppose we learn a simple patch prior from a given image (Figure 2a). To learn this prior we take all overlapping patches from the image, remove their DC component and build a histogram of all patches in the image, counting the times they appear in it. Under this prior, for example, the most likely patch would be flat (because the majority of patches in the original image are flat patches), the second most likely patch would be the tip of a diagonal edge and so on (see Figure 2b for a subset of this histogram). This prior is both easy to learn and easy to do denoising with by finding the MAP estimate given a corrupted patch. Now, suppose we are given a new, noisy *image* we wish to denoise (Figure 2c) - how should we do this using our *patch* prior?

The first, and simplest solution to this problem is to decompose the noisy image into a set of *non-overlapping* patches, denoise each patch independently by finding the MAP estimate from our prior and restore the image by placing each of the cleaned patches into its original position. This simple solution creates notorious artifacts at patch borders (see Figure 2d for an example) - if we now take a *random* patch from our newly constructed image (red patch in Figure 2d), it will be extremely unlikely under our prior (as most of the patches in the reconstructed image do not even exist in our prior, so their likelihood is 0). A more sophisticated solution may be to decompose the image into all *overlapping* patches, denoise each one independently and then average each pixel as it appears in the different patches to obtain the reconstructed image. This yields better results (see Figure 2f) but still has its problems - while we average the pixels together we create new patches in the reconstructed image which are not likely under our prior (red patch in Figure 2f). We can also take the central pixel from each of the overlapping patches but this suffers from the same problems (Figure 2e).

Going back to the motivation from Section 2, the intuition for our method is simple - suppose we take a random patch from our *reconstructed* image, we wish this patch to be *likely* under our prior. If we take another random patch from the reconstructed image, we want it also to be likely under our prior. In other words, we wish to find a reconstructed image in which *every* patch is *likely* under our prior while keeping the reconstructed image still close to the corrupted image — maximizing the *Expected Patch Log Likelihood* (EPLL) of the reconstructed image, subject to constraining it to be close to the corrupted image. Figure 2g shows the result of EPLL for the same noisy image — even though EPLL is using the *exact same* prior as in the previous methods, it produces superior results.

3.1. Framework and Optimization

3.1.1 Expected Patch Log Likelihood - EPLL

The basic idea behind our method is to try to maximize the Expected Patch Log Likelihood (EPLL) while still being close to the corrupted image in a way which is dependent on the corruption model. Given an image $\mathbf{x}$ (in vectorized form) we define the EPLL under prior p as:

$$EPLL_p(\mathbf{x}) = \sum_i \log p(\mathbf{P}_i\mathbf{x}) \tag{1}$$

Where $\mathbf{P}_i$ is a matrix which extracts the i-th patch from the image (in vectorized form) out of all *overlapping* patches, while $\log p(\mathbf{P}_i\mathbf{x})$ is the likelihood of the i-th patch under the prior p. Assuming a patch location in the image is chosen uniformly at random, EPLL is the expected log likelihood of a patch in the image (up to a multiplication by $1/N$).

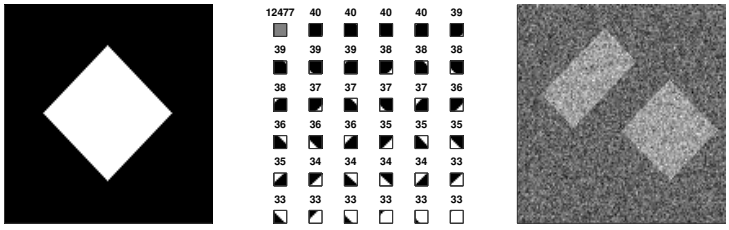
Now, assume we are given a corrupted image $\mathbf{y}$, and a model of image corruption of the form $\|\mathbf{A}\mathbf{x}-\mathbf{y}\|^2$ - We note that the corruption model we present here is quite general, as denoising, image inpainting and deblurring [7], among others, are special cases of it. We will discuss this in more detail in Section 3.1.3. The cost we propose to minimize in order to find the reconstructed image using the patch prior p is:

$$f_p(\mathbf{x}|\mathbf{y}) = \frac{\lambda}{2}\|\mathbf{A}\mathbf{x}-\mathbf{y}\|^2 - EPLL_p(\mathbf{x}) \tag{2}$$

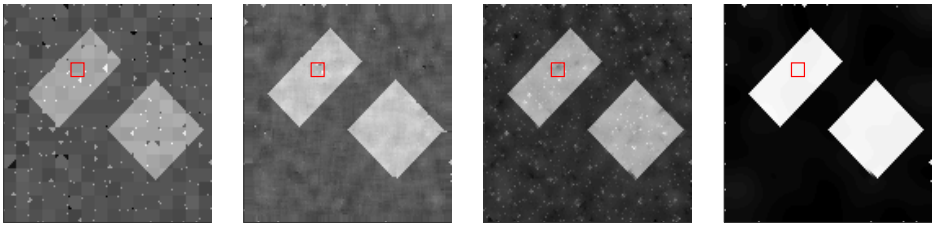
Equation 2 has the familiar form of a likelihood term and a prior term, but note that $EPLL_p(\mathbf{x})$ is *not* the log probability of a full image. Since it sums over the log probabilities of all overlapping patches, it "double counts" the log probability. Rather, it is the expected log likelihood of a randomly chosen patch in the image.

3.1.2 Optimization

Direct optimization of the cost function in Equation 2 may be very hard, depending on the prior used. We present here an



(a) Training Image (b) Prior Learned (c) Noisy Image



(d) Non Overlapping (e) Center Pixel (f) Averaged Overlapping (g) Our Method

Figure 2: The intuition behind our method. 2a A training image. 2b The prior learned from the image, only the 36 most frequent patches are shown with their corresponding count above the patch - flat patches are the most likely ones, followed by edges with 1 pixel etc. 2c A noisy image we wish to restore. 2d Restoring using non-overlapping patches - note the severe artifacts at patch borders and around the image. 2e Taking the center pixels from each patch. 2f Better results are obtained by restoring all overlapping patches, averaging the results - artifacts are still visible, and a lot of the patches in the resulting image are unlikely under the prior. 2g Result using the proposed method - note that there are very few artifacts, and most patches are very likely under our prior.

alternative optimization method called "Half Quadratic Splitting" which has been proposed recently in several relevant contexts [11, 7]. This method allows for efficient optimization of the cost. In "Half Quadratic Splitting" we introduce a set of patches $\left\{\mathbf{z^i}\right\}_1^{\mathbf{N}}$, one for each overlapping patch $\mathbf{P}_i\mathbf{x}$ in the image, yielding the following cost function:

$$c_{p,\beta}(\mathbf{x}, \left\{\mathbf{z^i}\right\}|\mathbf{y}) = \frac{\lambda}{2}\|\mathbf{A}\mathbf{x}-\mathbf{y}\|^2 + \sum_i \frac{\beta}{2}\left(\|\mathbf{P}_i\mathbf{x}-\mathbf{z}^i\|^2\right) - \log p(\mathbf{z}^i) \quad (3)$$

Note that as $\beta \to \infty$ we restrict the patches $\mathbf{P}_i\mathbf{x}$ to be equal to the auxiliary variables $\left\{\mathbf{z^i}\right\}$ and the solutions of Equation 3 and Equation 2 converge. For a *fixed* value of β, optimizing Equation 3 can be done in an iterative manner, first solving for $\mathbf{x}$ while keeping $\left\{\mathbf{z^i}\right\}$ constant, then solving for $\left\{\mathbf{z^i}\right\}$ given the newly found $\mathbf{x}$ and keeping it constant.

Optimizing Equation 3 for a fixed β value requires two steps:

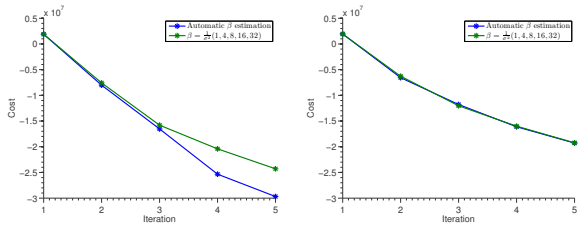
- Solving for $\mathbf{x}$ given $\left\{\mathbf{z^i}\right\}$ — This can be solved in closed form. Taking the derivative of 3 w.r.t to the vector $\mathbf{x}$, setting to 0 and solving the resulting equation yields

$$\hat{\mathbf{x}} = \left(\lambda\mathbf{A}^T\mathbf{A} + \beta\sum_j \mathbf{P}_j^T\mathbf{P}_j\right)^{-1} \left(\lambda\mathbf{A}^T\mathbf{y} + \beta\sum_j \mathbf{P}_j^T\mathbf{z}^j\right) \quad (4)$$

  Where the sum over j is for all overlapping patches in the image and all the corresponding auxiliary variables $\left\{\mathbf{z^i}\right\}$.

- Solving for $\left\{\mathbf{z^i}\right\}$ given $\mathbf{x}$ — The exact solution to this depends on the prior p in use - but for *any* prior it means solving a MAP problem of estimating the most likely patch under the prior, given the corrupted measurement $\mathbf{P}_i\mathbf{x}$ and parameter β.

We repeat the process for several iterations, typically 4 or 5 — at each iteration, solve for $\mathbf{Z}$ given $\mathbf{x}$ and solve for $\mathbf{x}$ given the new $\mathbf{Z}$, both given the current value of β. Then, increase β and continue to the next iteration. These two steps improve the cost $c_{p,\beta}$ from Equation 3, and for large β we also improve the original cost function f_p from Equation 2. We note that it is not necessary to find the *optimum* of each of the above steps, any approximate method (such as an approximate MAP estimation procedure) which still



(a) MAP estimation (b) BLS estimation

Figure 3: Optimization of the cost function from Equation 2, using the proposed optimization method. The results shown are for a simple denoising experiment, once with automatic estimation of β values, and once with a fixed schedule. In Sub-Figure 3a the MAP estimate for the patches was used. In Sub-Figure 3b the BLS estimate of the patches was used - it can be seen that even though we don't use the MAP in this experiment, the cost still decreases.

improves the cost of each sub-problem will still optimize the original cost function (albeit at different rates, depending on the exact setting).

The choice of β values is an open question. We use two approaches - the first is optimizing the values on a set of training images (by hand, or by brute force). The second option, which is relevant in denoising, is to try to estimate β from the current image estimate at every step — this is done by estimating the amount of noise σ present in the image $\hat{\mathbf{x}}$, and setting $\beta = \frac{1}{\sigma^2}$. We use the noise estimation method of [12].

Figure 3 shows a small experiment in which we verify that the original cost function in Equation 2 indeed decreases as a function of iterations as β grows. At the first experiment, β was estimated from the current image estimate and in the second, we used a fixed β schedule. The prior used was the ICA prior for which the likelihood is easily calculated. Even though the half quadratic splitting is only guaranteed to monotonically decrease the cost for infinite β values, we show experimentally that the cost decreases for different schedules of β where the schedule effects mostly the convergence speed. In addition, even when using BLS (instead of MAP), the cost still decreases - this shows that we don't need the find the optimum at each step, just to improve the cost of each sub-problem.

In summary, we note three attractive properties of our general algorithm. First, it can use *any* patch based prior and second, its run time is only 4–5 times the run time of restoring with simple patch averaging (depending on the number of iterations). Finally, perhaps the most important one is that this framework does not require learning a model $P(\mathbf{x})$ where $\mathbf{x}$ is a natural image, rather, learning needs only to concentrate on modeling the probability of image patches.

3.1.3 Denoising, Deblurring and Inpainting

In denoising, we have additive white Gaussian noise corrupting the image, so we set the matrix $\mathbf{A}$ from Equation 4 to be the identity matrix, and set λ to be related to the standard deviation of the noise ($\approx \frac{1}{\sigma^2}$). This means that the solution for $\mathbf{x}$ at each optimization step is just a weighted average between the noisy image $\mathbf{y}$ and the average of pixels as they appear in the auxiliary overlapping patches. The solution for $\mathbf{Z}$ is just a MAP estimate with prior p and noise level $\sqrt{\frac{1}{\beta}}$. If we initialize $\mathbf{x}$ with the noisy image $\mathbf{y}$, setting $\lambda = 0$ and $\beta = \frac{1}{\sigma^2}$ results in simple patch averaging when iterating a single step. The big difference, however, is that in our method, because we iterate the solution and $\lambda \neq 0$, at each iteration we use the current estimated image, averaging it with the noisy one and obtaining a *new* set of $\mathbf{Z}$ patches, solving for them and then obtaining a new estimate for $\mathbf{x}$, repeating the process, while increasing β. For image deblurring (non-blind) $\mathbf{A}$ is a convolution matrix with a known kernel.

Image inpainting is similar, $\mathbf{A}$ is a diagonal matrix with zeros for all the missing pixels. Basically, this can be thought of as "denoising" with a per pixel noise level - infinite noise for missing pixels and zero noise for all other pixels. See Supplementary Material for some examples of inpainting.

3.2. Related Methods

Several existing methods are closely related, but are fundamentally different from the proposed framework. The first related method is the Fields of Experts (FoE) framework by Roth and Black [6]. In FoE, a Markov Random Field (MRF) whose filters are trained by approximately maximizing the likelihood of the training *images* is learned. Due to the intractability of the partition function, learning with this model is extremely hard and is performed using contrastive divergence. Inference in FoE is actually a special case of our proposed method - while the learning is vastly different, in FoE the inference procedure is equivalent to optimizing Equation 2 with an independent prior (such as ICA), whose filters were learned before hand. A common approximation to learning MRFs is to approximate the log probability of an image as a sum of local marginal or conditional probabilities as in the method of composite likelihood [13] or directed models of images [14]. In contrast, we do not attempt to approximate the global log probability and argue that modeling the local patch marginals is sufficient for image restoration. This points to one of the advantages of our method - learning a patch prior is much easier than learning a MRF. As a result, we can learn a much richer patch prior easily and incorporate it into our framework - as we show later.

Another closely related method is KSVD [3] - in KSVD, one learns a patch based dictionary which attempts to maximize the sparsity of resulting coefficients. This dictionary

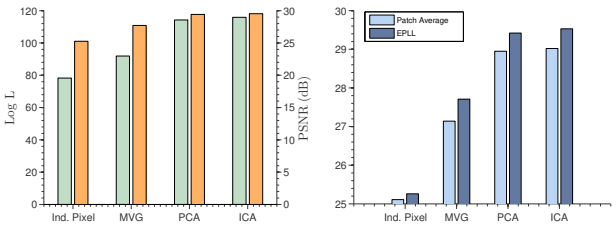
can be learned either from a set of natural image patches (generic, or global as it is sometimes called) or the noisy image itself (image based). Using this dictionary, all overlapping patches of the image are denoised independently and then averaged to obtain a new reconstructed image. This process is repeated for several iterations using this new estimated image. Learning the dictionary in KSVD is different than learning a patch prior because it may be performed as part of the optimization process (unless the dictionary is learned beforehand from natural images), but the optimization in KSVD can be seen as a special case of our method - when the prior is a sparse prior, our cost function and KSVD's are the same. We note again, however, that our framework allows for much richer priors which can be learned beforehand over patches - as we will see later on, this boasts some tremendous benefits.

A whole family of patch based methods [2, 5, 8] use the noisy image itself in order to denoise. These "non-local" methods look for similarities within the noisy image itself and operate on these similar patches together. BM3D [5] groups together similar patches into "blocks", transforms them into wavelet coefficients (in all 3 dimensions), thresholds and transforms backs, using all the estimates together. Mairal et al. [8] which is currently state-of-the-art take a similar approach to this, but instead of transforming the patches via a wavelet transform, sparse coding using a learned dictionary is used, where each block is constrained to use the same dictionary elements. One thing which is common to all of the above non-local methods, and indeed almost all patch based methods, is that they all average the clean patches together to form the final estimate of the image. As we have seen in Section 3, this may not be the optimal thing to do.

3.3. Patch Likelihoods and the EPLL Framework

We have seen that the EPLL cost function (Equation 2) depends on the likelihood of patches. Going back to the priors from Section 2 we now ask - do better priors (in the likelihood sense) also lead to better *whole* image denoising with the proposed EPLL framework? Figure 4 shows the average PSNR obtained with 5 different images from the Berkeley training set, corrupted with Gaussian noise at $\sigma = 25$ and denoised using each of the priors in section 2. We compare the result obtained using simple patch averaging (PA) and our proposed EPLL framework. It can be seen that indeed - better likelihood on patches leads to better denoising both on independent patches (Figure 1) and whole images (Figure 4). Additionally, it can be seen that EPLL improves denoising results significantly when compared to simple patch averaging (Figure 4).

These results motivate the question: Can we find a better prior for image patches?



(a) (b)

Figure 4: **(a)** Whole image denoising with the proposed framework with all the priors discussed in Section 2. It can be seen that better priors (in the likelihood sense) lead to better denoising performance on *whole* images, left bar is log L, right bar is PSNR. **(b)** Note how the EPLL framework improves performance significantly when compared to simple patch averaging (PA)

4. Can We Learn Better Patch Priors?

In addition to the priors discussed in Section 2 we introduce a new, simple yet surprisingly rich prior.

4.1. Learning and Inference with a Gaussian Mixture Prior

We learn a finite Gaussian mixture model over the pixels of natural image patches. Many popular image priors can be seen as special cases of a GMM (e.g. [9, 1, 14]) but they typically constrain the means and covariance matrices during learning. In contrast, we do not constrain the model in any way — we learn the means, full covariance matrices and mixing weights, over all pixels. Learning is easily performed using the Expectation Maximization algorithm (EM). With this model, calculating the log likelihood of a given patch is trivial:

$$\log p(\mathbf{x}) = \log \left(\sum_{k=1}^{K} \pi_k N(\mathbf{x}|\mu_k, \Sigma_k) \right) \tag{5}$$

Where π_k are the mixing weights for each of the mixture component and μ_k and Σ_k are the corresponding mean and covariance matrix.

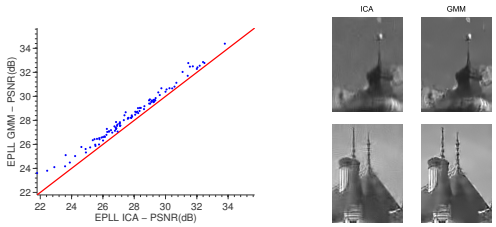
Given a noisy patch $\mathbf{y}$, the BLS estimate can be calculated in closed form (as the posterior is just another Gaussian mixture) [1]. The MAP estimate, however, can not be calculated in closed form. To tackle this we use the following approximate MAP estimation procedure:

1. Given noisy patch $\mathbf{y}$ we calculate the conditional mixing weights $\pi'_k = P(k|\mathbf{y})$.
2. We choose the component which has the highest conditional mixing weight $k_{max} = \max_k \pi'_k$.
3. The MAP estimate $\hat{\mathbf{x}}$ is then a Wiener filter solution for the k_{max}-th component:

$$\hat{\mathbf{x}} = \left(\mathbf{\Sigma}_{k_{max}} + \sigma^2 \mathbf{I}\right)^{-1} \left(\mathbf{\Sigma}_{k_{max}} \mathbf{y} + \sigma^2 \mathbf{I} \mu_{k_{max}}\right)$$

		Patch Restoration		Image Restoration	
Model	Log L	BLS	MAP	PA	EPLL
Ind. Pixel	78.26	25.54	24.43	25.11	25.26
MVG	91.89	26.81	26.81	27.14	27.71
PCA	114.24	28.01	28.38	28.95	29.42
ICA	115.86	28.11	28.49	29.02	29.53
GMM	**164.52**	**30.26**	**30.29**	**29.59**	**29.85**

Table 1: GMM model performance in log likelihood (Log L), patch denoising (BLS and MAP) and image denoising (Patch Average (PA) and EPLL, the proposed framework) - note that the performance is better than all priors in all measures. The patches, noisy patches, images and noisy images are the same as in Figure 1 and Figure 4. All values are in PSNR (dB) apart from the log likelihood.



(a) PSNR Comparison (b) Detail Shots

Figure 5: Comparison of the performance of the ICA prior to the high likelihood GMM prior using EPLL and noise level $\sigma = 25$. 5a depicts a scatter plot of PSNR values obtained when denoising 68 images from[10]. Note the superior performance of the GMM prior when compared to ICA on all images. 5b depicts a detail shot from two of the images - note the high visual quality of the GMM prior result. The details are best seen when zoomed in on a computer screen.

This is actually one iteration of the "hard version" of the EM algorithm for finding the modes of a Gaussian mixture [15].

4.2. Comparison

We learn the proposed GMM model from a set of 2×10^6 patches, sampled from [10] with their DC removed. The model is with learned 200 mixture components with zero means and full covariance matrices. We also trained GMMs with unconstrained means and found that all the means were very close to zero. As mentioned above, learning was performed using EM. Training with the above training set takes around 30h with unoptimized MATLAB code[1]. Denoising a patch with this model is performed using the approximate MAP procedure described in 4.1.

Having learned this GMM prior, we can now compare its performance both in likelihood and denoising with the priors we have discussed thus far in Section 1 on the same dataset of *unseen* patches. Table 1 shows the results obtained with the GMM prior - as can be seen, this prior is superior in likelihood, patch denoising and whole image denoising to all other priors we discussed thus far.

In Figure 5a we show a scatter plot of PSNR values obtained with ICA and the GMM model using EPLL at noise level $\sigma = 25$ on 68 images from the Berkeley test set. Note that the high likelihood GMM model is superior to ICA in denoising, on all tested images. Figure 5b shows details from images in the test-set, note the high visual quality of the GMM model when compared to the ICA result.

Why does this model work so well? One way to understand it is to recall that in a zero-mean Gaussian mixture model, every sample x is well approximated by the top m eigenvectors of the covariance matrix of the mixture component that it belongs to. If we consider the set of all m eigenvectors of all mixtures as a "dictionary" then every sample is approximated by a sparse combination of these dictionary elements. Since there are 200 mixture components, only (1/200) dictionary elements are "active" for each x so this is a very sparse representation. But unlike other models that assume sparsity (e.g. ICA and Sparse Coding), the active set is extremely constrained — only dictionary elements that correspond to the same component are allowed to be jointly active. We have recently learned that this "dual" interpretation of a GMM was independently given by [16] for the case of image-specific GMMs.

What do these dictionary elements model? Figure 6 depicts the eigenvectors of the 5 randomly selected mixture components from the learned model. Note that these have rich structures - while some resembles PCA eigenvectors, some depict forms of occlusions, modeling texture boundaries and edges. These are very different from the Gabor filters usually learned by sparse coding and similar models. It would seem that these structures contribute much to the expressive power of the model.

4.3. Comparison to State-Of-The-Art Methods

We compare the performance of EPLL with the proposed GMM prior with leading image restoration methods - both generic and image based. All the experiments were conducted on 68 images from the test set of the Berkeley Segmentation Database [10]. All experiments were conducted using the same noisy realization of the images. In all experiments we set $\lambda = \frac{N}{\sigma^2}$, where N is the number of pixels in each patch. We used a patch size of 8×8 in all experiments. For the GMM prior, we optimized (by hand) the values for β on the 5 images from the Berkeley training set - these were set to $\beta = \frac{1}{\sigma^2} \cdot [1, 4, 8, 16, 32, 64]$. Running times on a Quad Core Q6600 processor are around 300s per image with unoptimized MATLAB code.

[1] Downloaded from: http://www.mathworks.com/matlabcentral/fileexchange/26184-em-algorithm-for-gaussian-mixture-model

σ	KSVDG	FoE	GMM-EPLL
15	30.67	30.18	**31.21**
25	28.28	27.77	**28.71**
50	25.18	23.29	**25.72**
100	22.39	16.68	**23.19**

(a) Generic Priors

σ	KSVD	BM3D	LLSC	GMM-EPLL
15	30.59	30.87	**31.27**	31.21
25	28.20	28.57	**28.70**	**28.71**
50	25.15	25.63	**25.73**	**25.72**
100	22.40	**23.25**	23.15	23.19

(b) Image Based Methods

Table 2: Summary of denoising experiments results. Our method is clearly state-of-the-art when compared to generic priors, and is competitive with image based method such as BM3D and LLSC which are state-of-the-art in image denoising.

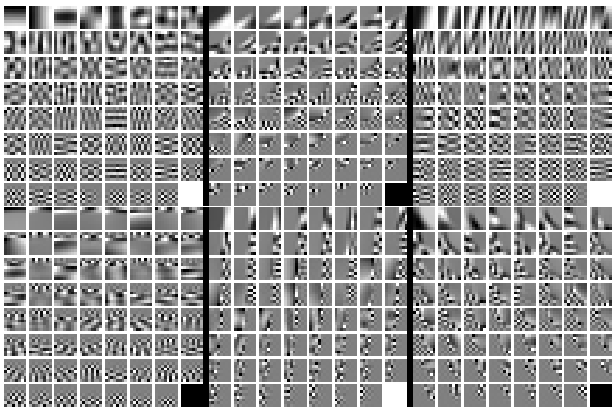
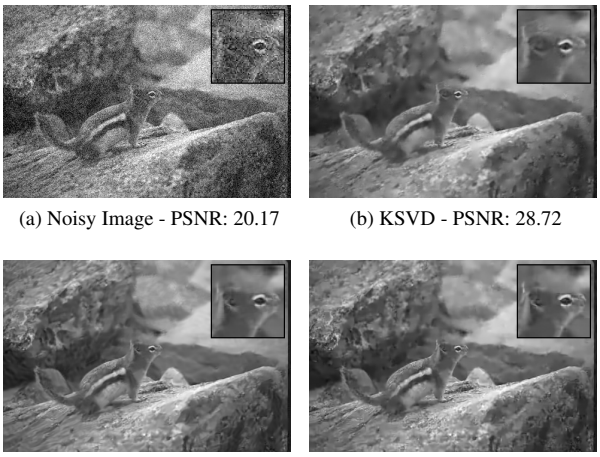


Figure 6: Eigenvectors of 6 randomly selected covariance matrices from the learned GMM model, sorted by eigenvalue from largest to smallest. Note the richness of the structures - some of the eigenvectors look like PCA components, while others model texture boundaries, edges and other structures at different orientations.



(a) Noisy Image - PSNR: 20.17 (b) KSVD - PSNR: 28.72

(c) LLSC - PSNR: 29.30 (d) EPLL GMM - PSNR: 29.39

Figure 7: Examples of denoising using EPLL-GMM compared with state-of-the-art denoising methods - KSVD [3] and LLSC [8]. Note how detail is much better preserved in our method when compared to KSVD. Also note the similarity in performance with our method when compared to LLSC, even though LLSC learn from the noisy image. See supplementary material for more examples.

4.3.1 Generic Priors

We compare the performance of EPLL and the GMM prior in image denoising with leading *generic* methods - Fields of Experts [6] and KSVD [3] trained on natural image patches (KSVDG). The summary of results may be seen in Table 2a - it is clear that our method outperforms the current state-of-the-art generic methods.

4.3.2 Image Based Priors

We now compare the performance of our method (EPLL+GMM) to image specific methods - which learn from the noisy image itself. We compare to KSVD, BM3D [5] and LLSC [8] which are currently the state-of-the-art in image denoising. The summary of results may be seen in Table 2b. As can be seen, our method is highly competitive with these state-of-the-art method, even though it is generic. Some examples of the results may be seen in Figure 7.

4.3.3 Image Deblurring

While image specific priors give excellent performance in denoising, since the degradation of different patches in the same image can be "averaged out", this is certainly not the case for all image restoration tasks, and for such tasks a generic prior is needed. An example of such a task is image deblurring. We convolved 68 images from the Berkeley database (same as above) with the blur kernels supplied with the code of [7]. We then added 1% white Gaussian noise to the images, and attempted reconstruction using the code by [7] and our EPLL framework with GMM prior. Results are superior both in PSNR and quality of the output, as can be seen in Figure 8.

(a) Blurred (b) Krishnan et al. (c) EPLL GMM

	Krishnan et al.	EPLL-GMM
Kernel 1 17×17	25.84	27.17
Kernel 2 19×19	26.38	27.70

Figure 8: Deblurring experiments

5. Discussion

Patch based models are easier to learn and to work with than whole image models. We have shown that patch models which give high likelihood values for patches sampled from natural images perform better in patch and image restoration tasks. Given these results, we have proposed a framework which allows the use of patch models for whole image restoration, motivated by the idea that patches in the restored image should be likely under the prior. We have shown that this framework improves the results of whole image restoration considerably when compared to simple patch averaging, used by most present day methods. Finally, we have proposed a new, simple yet rich Gaussian Mixture prior which performs surprisingly well on image denoising, deblurring and inpainting.

While we have demonstrated our framework using only a few priors, one of its greater strengths is the fact that it can serve as a "plug-in" system - it can work with any existing patch restoration method. Considering the fact that both BM3D and LLSC are patch based methods which use simple patch averaging, it would be interesting to see how would these methods benefit from the proposed framework.

Finally, perhaps the most surprising result of this work, and the direction in which much is left to be explored, is the stellar performance of the GMM model. The GMM model used here is extremely naive - a simple mixture of Gaussians with full covariance matrices. Given the fact that Gaussian Mixtures are an extremely studied area, incorporating more sophisticated machinery into the learning and the representation of this model holds much promise - and this is our current line of research.

Acknowledgments

The authors wish to thank Anat Levin for helpful discussions.

References

[1] J. Portilla, V. Strela, M. Wainwright, and E. Simoncelli, "Image denoising using scale mixtures of gaussians in the wavelet domain," *IEEE Transactions on Image Processing*, vol. 12, no. 11, pp. 1338–1351, 2003.

[2] A. Buades, B. Coll, and J. Morel, "A non-local algorithm for image denoising," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2, pp. 60–65, IEEE, 2005.

[3] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *Image Processing, IEEE Transactions on*, vol. 15, no. 12, pp. 3736–3745, 2006.

[4] Y. Hel-Or and D. Shaked, "A discriminative approach for wavelet denoising," *IEEE Transactions on Image Processing*, vol. 17, no. 4, p. 443, 2008.

[5] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image restoration by sparse 3D transform-domain collaborative filtering," in *SPIE Electronic Imaging*, 2008.

[6] S. Roth and M. Black, "Fields of experts," *International Journal of Computer Vision*, vol. 82, no. 2, pp. 205–229, 2009.

[7] D. Krishnan and R. Fergus, "Fast image deconvolution using hyper-laplacian priors," in *Advances in Neural Information Processing Systems 22*, pp. 1033–1041, 2009.

[8] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Non-local sparse models for image restoration," in *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 2272–2279, IEEE, 2010.

[9] Y. Weiss and W. Freeman, "What makes a good model of natural images?," *CVPR '07. IEEE Conference on*, pp. 1–8, June 2007.

[10] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. 8th Int'l Conf. Computer Vision*, vol. 2, pp. 416–423, July 2001.

[11] D. Geman and C. Yang, "Nonlinear image recovery with half-quadratic regularization," *Image Processing, IEEE Transactions on*, vol. 4, no. 7, pp. 932–946, 2002.

[12] D. Zoran and Y. Weiss, "Scale invariance and noise in natural images," in *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 2209–2216, Citeseer, 2009.

[13] B. Lindsay, "Composite likelihood methods," *Contemporary Mathematics*, vol. 80, no. 1, pp. 221–39, 1988.

[14] J. Domke, A. Karapurkar, and Y. Aloimonos, "Who killed the directed model?," in *CVPR 2008. IEEE Conference on*, pp. 1–8, IEEE, 2008.

[15] M. Carreira-Perpiñán, "Mode-finding for mixtures of Gaussian distributions," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 11, pp. 1318–1323, 2002.

[16] G. Yu, G. Sapiro, and S. Mallat, "Solving inverse problems with piecewise linear estimators: From gaussian mixture models to structured sparsity," *CoRR*, vol. abs/1006.3056, 2010.