Comparative Analysis of EM Algorithm and Markov Random Fields for Image Denoising

Team Members: Athanasios Taprantzis, Sripushkar Julapally

ISyE 6416 - Computational Statistics

Project Code: GitHub

Problem Statement

Image restoration remains a fundamental challenge in computational statistics and computer vision. Real-world images are often corrupted by noise or contain missing regions (inpainting). Tough lighting conditions or faulty equipment can often mean high amounts of noise or missing pixels in image stills. This project aims to compare and evaluate two principled statistical approaches for image denoising: Expectation-Maximization (EM) with Gaussian Mixture Models (GMM) and Markov Random Fields (MRFs). While both methods leverage statistical modeling, they represent fundamentally different paradigms—EM operates in a patch-based feature space, while MRF models spatial dependencies directly in the image domain.

Motivation: Understanding the relative strengths, computational requirements, and performance characteristics of these approaches provides valuable insights for selecting appropriate methods in practical applications and demonstrates the versatility of statistical modeling for inverse problems.

Data Sources

We used standard benchmark datasets to ensure reproducibility and fair comparison:

- Set12 and BSD68 datasets for denoising evaluation

  These are standard grayscale denoising datasets and benchmarks

- MNIST and CIFAR-10 for controlled experiments

MNIST provides simple high contrast images of handwritten digits. CIFAR-10 consists of 60,000 color images divided into 10 classes.

- Custom synthetic images with known ground truth for method validation
- Real photographs (with real noise) with simulated corruption for qualitative assessment

All datasets are publicly available and require no special permissions. For denoising, we simulated noise using salt-and-pepper and gaussian noise.

Methodology

Image denoising aims to restore fidelity that has been corrupted by some noise. This can be thought of as a Maximum a Posteriori (MAP) estimate where we know the corrupted image and we are trying to restore the prior. Estimating this prior can be done through a multitude of methods. The two we reviewed in our work are the following:

Expectation-Maximization with Gaussian Mixture Models (EM-GMM)

The key insight for EM-GMM is that we can estimate the prior by matching the corrupted image to other "clean" images. The core idea of the approach is that we hope to have some large enough database of clean images so that we can extract the corrupted image's (call it y) patches and match them back to a clean patch we have already seen.

Mapping the noisy image as a whole to some other clean image is pointless as it is extremely unlikely we have the exact same image in our database. Likewise, if the patches are very small, we are not gaining enough context about the patch and the noise present in it.

We ideally try to map patches of appropriate size from y to some learned prior patch we have in our database.

Technical Analysis

Learning the patch priors can be achieved through various approaches. We chose to use Gaussian Mixture Models to learn the most prevalent patches across our database of images. This is the same approach that Zoran and Weiss [1] as well as Luo et al. [2] chose to use.

With a learned prior we will now define the loss function we will try to optimise. We want the loss to model our need to find an image that closely matches y but also matches some of our learned patches with the hopes that the learned priors help us construct an image that holds the signal of y and gets rid of the noise. The loss is:

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

where EPLL is the Expected Patch Log Likelihood which models how close the patches of x are to the learnt priors and is precisely:

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

The loss models the distance our restored image x has from the y (given some transformation A which in our case is simply the Identity Matrix) while EPLL term rewards any x whose patches are closely aligned with our prior patches.

Optimisation

Optimising the above loss would yield us the restored image x. However, as Zoran and Weiss[1] note, this is a hard optimisation problem. Trying to optimise it directly is incredibly difficult and is additionally dependent on the distribution we chose to model the prior patches (GMM in our case).

In our work we used two optimisation methods:

A. As proposed in the original paper [1], we use Half Quadratic Splitting: we introduce an auxiliary set of patches $\{x^i\}_1^N$ which will act as an intermediary between our restored image x and the corrupted original image y. Our new loss is:

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

The optimisation is now a two-fold problem. Optimising x given $z^i \quad \forall i$ can be thought of as trying to make x approximate both the corrupted image y and have some parity with the priors z^i. On the other hand optimising z given x is equivalent to updating z such that it is as close as possible to both x's patches and the prior patches.

Optimisation here needs to occur in two steps, one for x and one for z. Zoran and Weiss [1] tell us that there exists a closed form solution for optimising x:

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

Note that $P_j^T z^j$ is the image constructed from the overlap of all the patches z^i while $P_j^T P_j$ is a frequency matrix denoting the number of patches that overlap at any pixel.$P_j^T z^j$ is the image constructed from the overlap of all the patches z^i while $P_j^T P_j$ is a frequency matrix denoting the number of patches that overlap at any pixel.

They also give a closed form solution for an *approximate* MAP estimation for z (note that $\hat{x}$ in the equation maps to z^i in our loss function):

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

Using these formulas we can perform optimisation fairly efficiently and effectively. The choice of λ and β is dependent on the data. In our work we used values that we experimentally confirmed to produce the best results.

B. Instead of attempting to follow the algebraic solutions to the above optimisation problem, we used Automatic Differentiation [3] to approximate the loss's gradient and used the Adam optimisation method to update x towards the estimated gradient. This method is surprisingly simple and effective but lacks the efficiency of method (A) due to the fact that we gradient descent takes several steps to reach minimums - which could make convergence slow - while (A) has formal guarantees for the closed form solutions we presented, making it a much faster but potentially unstable method.

   We used pytorch's built-in computation graph to keep track of the variable's relationship and to perform Automatic Differentiation.

We tested both optimisation methods as they yielded noticeably different results.

Implementation

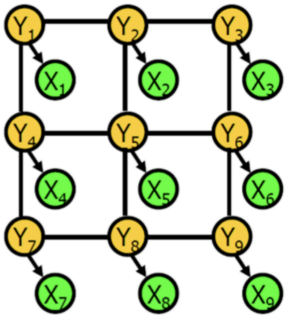
Here is a concise overview of our EPLL algorithm implementation:

1. Extract overlapping patches from clean images
2. Learn GMM parameters using EM algorithm on the clean patches
3. Extract overlapping patches from clean images
4. Optimise based on (A) or (B)
5. Stop on convergence or after fixed number of iterations

For method (A) we used a list of increasing β values as suggested by Zoran and Weiss [1] to slowly encourage convergence of x and z.

Markov Random Fields (MRF)

We can treat an image as an energy-based system. We treat an image as a state of a system which exhibits certain energy at particular states. The underlying assumption is that an image exhibits a lot of similarity across neighbouring pixels, while the ratio of neighbouring pixels that vary drastically from one another are few and far between. Thus, we can assume that an image is a state of a system whose energy tends to be lower the closer the pixels are to their neighborhood (in terms of value).



The visualisation on the right is our system; $x_{i,j}$ are the given, corrupted pixels and we know there is some relationship between the true pixels $y_{i,j}$, the corrupted pixels, and their neighbourhood. Notice that the neighborhood is *other random variables*. Thus, we will be working with a Markov Random Field and not simply a Markov Chain.$x_{i,j}$ are the given, corrupted pixels and we know there is some relationship between the true pixels $y_{i,j}$, the corrupted pixels, and their neighbourhood. Notice that the neighborhood is other random variables. Thus, we will be working with a Markov Random Field and not simply a Markov Chain.

Technical Analysis

We can build a loss we will try and minimise. This will treat the problem of denoising as a problem of minimsing the energy of the system we just described.

Notice that since we have an energy-based system we can use the Boltzmann distribution to model the posterior probability:

$$p(y_{i,j}|N(y_{i,j})) = \frac{1}{Z}exp(-E(y_{i,j}|X))$$

$y_{i,j}$ is the current estimate of the denoised image while $X = N(y_{i,j})$ is the corrupted image.$y_{i,j}$ is the current estimate of the denoised image while $X = N(y_{i,j})$ is the corrupted image.

Let us now define what the energy will be:

$$L(y_{i,j}|X) = \sum_{z \in N(y_{i,j}) \setminus x_{i,j}} ||z - y_{i,j}||_{n_1}^{n_1} + \lambda ||x_{i,j} - y_{i,j}||_{n_1}^{n_1}$$

The first term makes the model penalise estimations where the pixels stray away from their neighbourhood's mean. This refers to the markov blanker of pixel $y_{i,j}$. The second term penalises estimates that stray away from the original (noisy) image. This is crucial as this forces our model to remove noise but without losing parity with the corrupted image - in the hopes of only preserving the signal and ignore the noise. Notice that without the second term, the model would collapse to some fixed colour since all pixels would have the same colour and the system would be at a state of minimal energy. λ is a simple hyperparameter that balances out the effect of each term.$y_{i,j}$. The second term penalises estimates that stray away from the original (noisy) image. This is crucial as this forces our model to remove noise but without losing parity with the corrupted image - in the hopes of only preserving the signal and ignore the noise. Notice that without the second term, the model would collapse to some fixed colour since all pixels would have the same colour and the system would be at a state of minimal energy. λ is a simple hyperparameter that balances out the effect of each term.

Looking at the above it becomes clear that minimising the loss/energy is equivalent to maximing the posterior probability.

Optimisation

The classic approach would be to optimise the above function. Instead we will take a different approach. We already showed what the probability distribution of each pixel is given the energy function above. If we could somehow sample from that underlying distribution, we would - on average - sample pixels with higher probability more frequently than others with less probability. This is equivalent to saying that sampling from that distribution will yield pixel that will - overall - result in a state of lower energy/loss.

We can use Metropolis-Hastings to sample from teh underlying distribution of random pixels. After enough samples we hope to see the image converge to a lower energy state which would minimise the

loss we defined above and would denoise the image. An additional benefit of the Metropolis-Hastings is that we can effectively avoid calculating the normalising coefficient for the Boltmann distribution which can be very expensive to calculate and is additionally dependent on the choice of norm.

Implementation

Implementing the above is fairly straight forward once the intuition is in place:

1. Pick an order of the pixels from the corrupted image
2. For each pixel (in the order chosen) sample from the distribution we showed previously
3. Update the value of the pixel with the sampled value
4. Repeat until convergence

Experimental Design & Results

During development of the models we primarily worked with custom images, MNIST and CIFAR-10 along with various types and levels of noise. In the experiments we present we mainly focused on the Set12 and BSD68 datasets for reproducibility.

For **quantitative evaluation** of the results we used Peak-Signal-to-Noise-Ratio (PSNR) and Structural-Similarity-Index (SSI) as our two metrics. For **qualitative evaluation** we visually inspected the artifact types and structure preservation. For efficiency evaluation we used computation time for each method till convergence.

Results

Experimentation for MRF Method

To evaluate how the MRF based Monte Carlo denoiser works with different parameters, we ran an experiment tuning two variables: the total number of Metropolis-Hastings iterations and the fidelity weight. We used a control image corrupted with Gaussian noise.

Iterations	β	PSNR	SSIM	Time (s)
500,000	0.5	21.89	0.4017	4.03
500,000	1	21.91	0.4024	4.39
500,000	1.5	21.88	0.4011	4.19
500,000	2	21.89	0.4013	4.03
2,000,000	0.5	24.74	0.5415	15.97
2,000,000	1	24.8	0.5374	15.85
2,000,000	1.5	24.82	0.532	15.98
2,000,000	2	24.72	0.5228	15.98
5,000,000	0.5	24.5	0.6865	40.71
5,000,000	1	25.12	0.6448	40.61
5,000,000	1.5	25.32	0.6067	44.78
5,000,000	2	25.3	0.5766	41.69

Best PSNR Configuration

- PSNR: 25.32
- Parameters: 5,000,000 iterations, $\beta = 1.5$

Best SSIM Configuration

- SSIM: 0.6865
- Parameters: 5,000,000 iterations, $\beta = 0.5$

Observations

Iterations: Increasing the number of iterations seems to improve both PSNR and SSIM. At 500k iterations the method has clearly not converged; 5M iterations show much better visual and numerical performance. This agrees with the theoretica backbone of MCMC as more iterations will better converge to the posterior distribution.

Effect of β:

- Lower β places more weight on smoothness, which tends to improve SSIM but can blur fine detail.
- Higher β enforces fidelity to the noisy observation, improving PSNR at the cost of more visible grain.
- Trade-off: The results show a tradeoff between structural similarity and signal fidelity. $\beta = 0.5$ leads to the most perceptually smooth reconstructions, while $\beta = 1.5$ best preserves pixel wise detail.

This experimentation helped us with the hyperparameter tuning of β.

Experimentation for EM Method

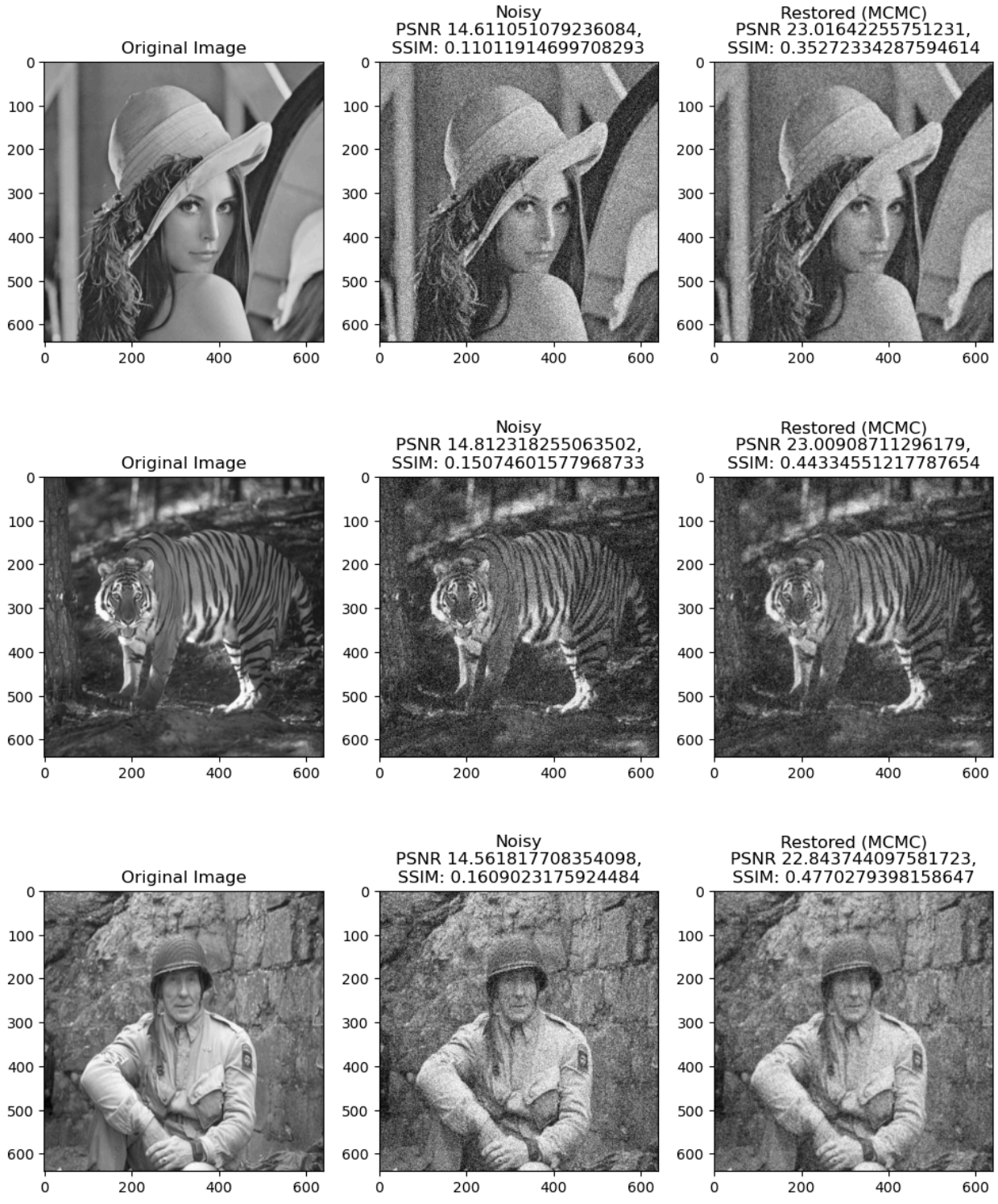
Due to hardware constraints, the model runs extremely slow as the our selection of prior patches increases, making it extremely time consuming to try out new image sets for do extensive testing for hyperparameter tuning. For our priors, we fitted a GMM model using *Diagonal Covariance Matrices* instead of Full Covariance Matrices. This naturally means less performance for our model but our image set - consisting of just 12 images - kept surpassing 40GB of allocated memory when used to learn the GMM components, making our model crash constantly. Additionally, we increased the stride when extracting patches from the image set to reduce the large matrix multiplications required to compute the updated z_i patches.

With a few sample trainings and following Zoran and Weiss's [1] empirical advice for intialising the hyper-parameters, we set λ to $\frac{1}{\sigma^2}$ and set the β equal to λ and kept steadily increasing it throughout the iterations.

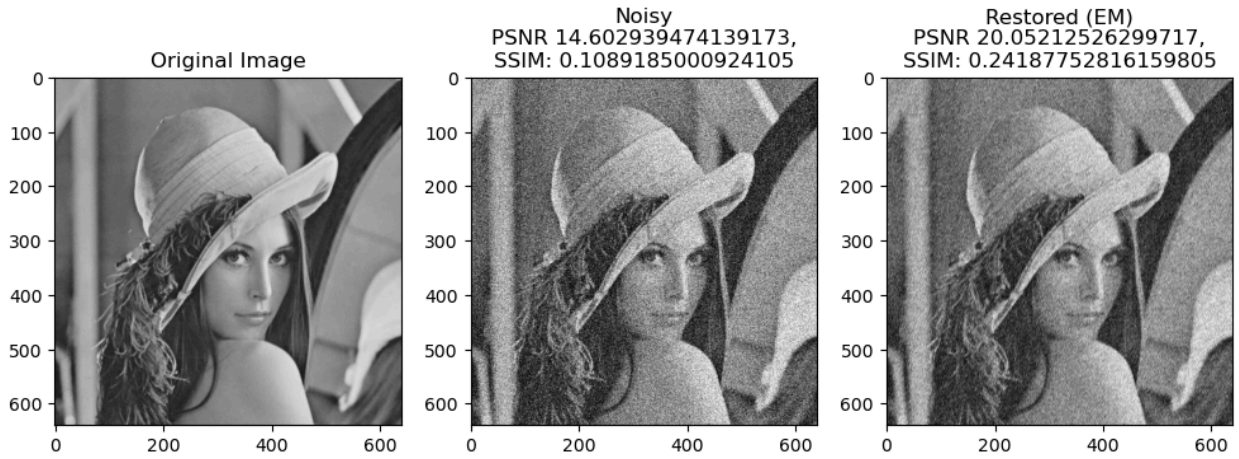
Experimental Results - Gaussian Noise | σ = 50

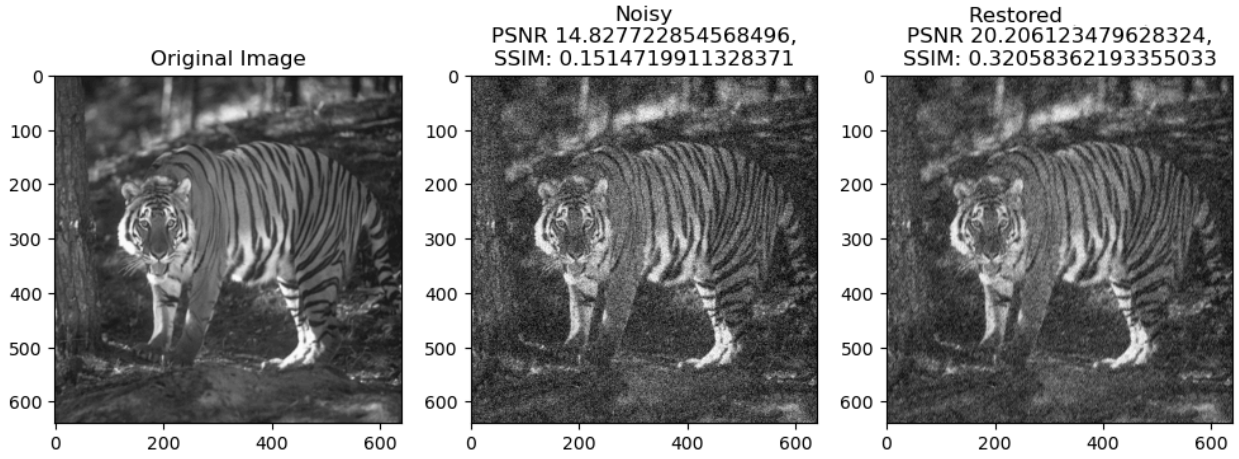
MCMC with β = 0.5 (Mean training time = 9.42 mins):

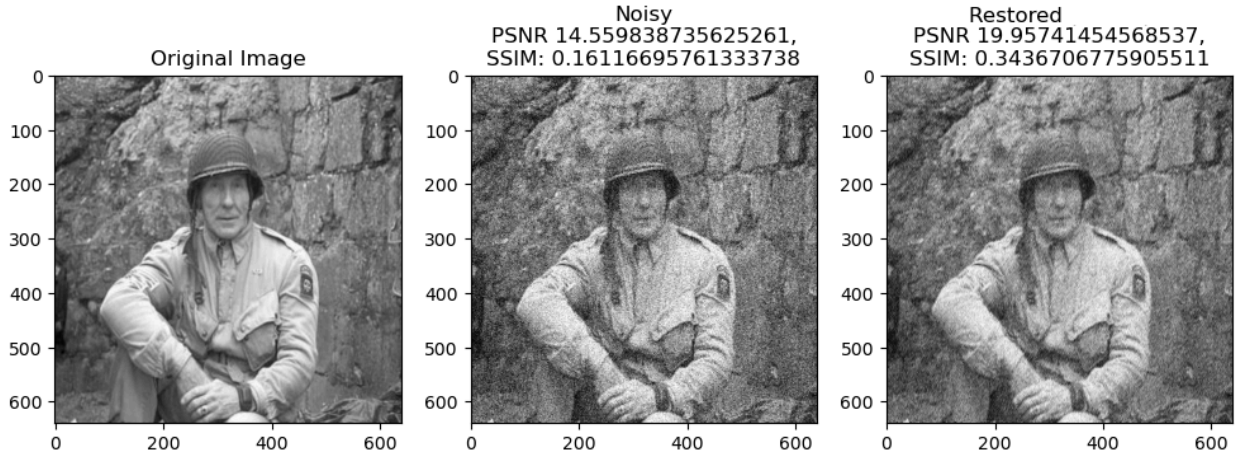
MCMC with β = 1.5 (Mean training time = 8.89 mins):



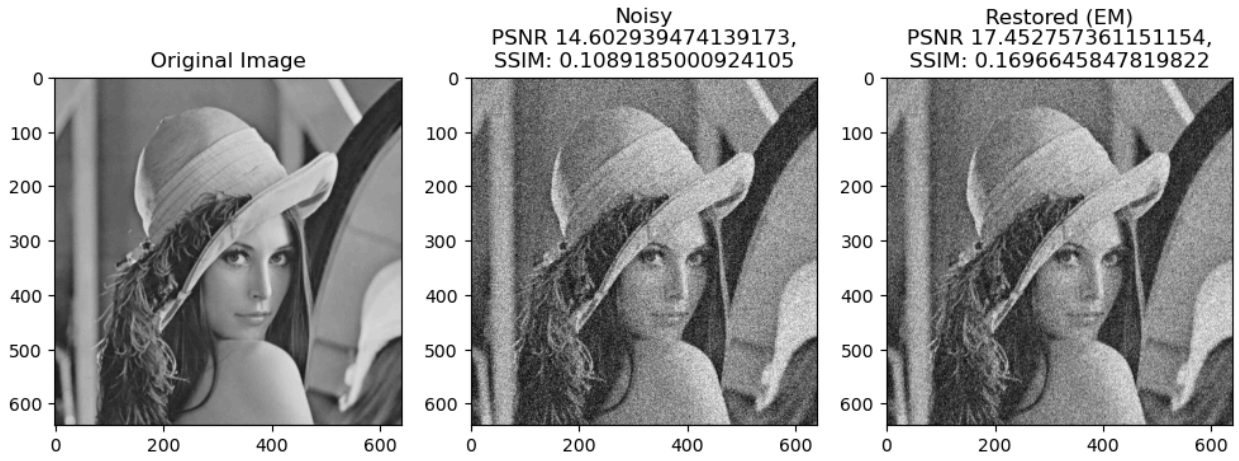
EM/EPLL - Half Quadratic Split (Mean training time = 2.5 mins):

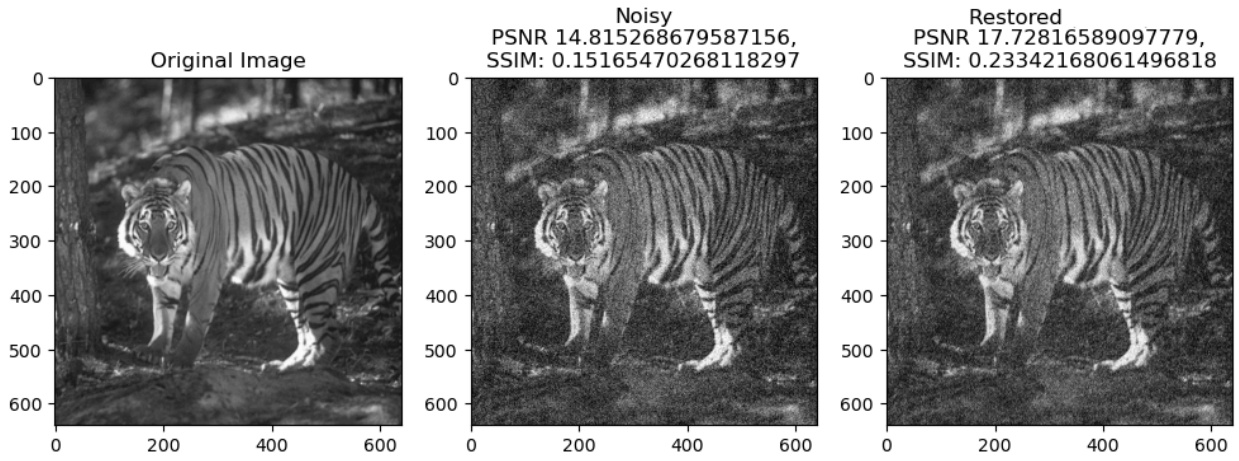


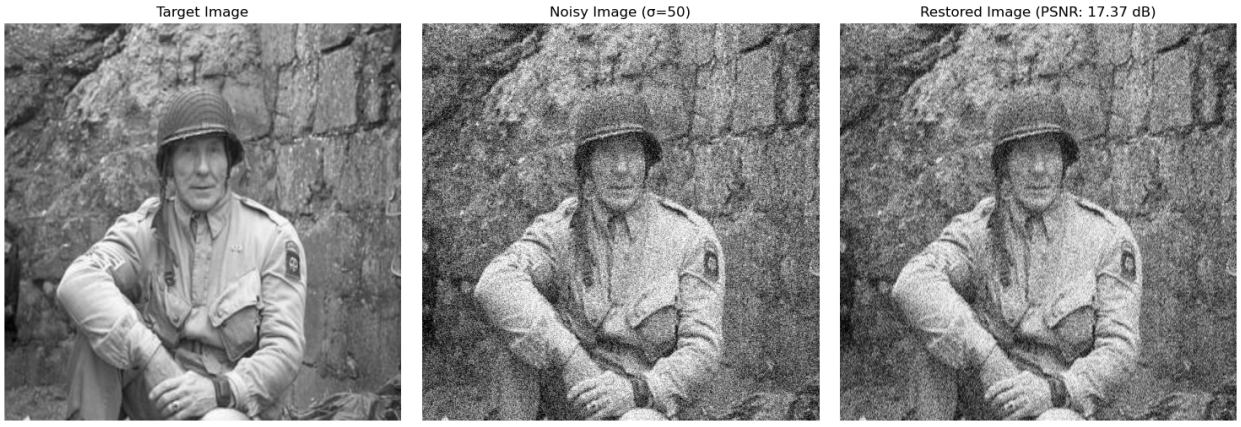




EM/EPLL - Auto-differentiation (Mean training time = 5 mins, Learning Rate = 1.0):







Conclusion

The results above reveal a lot of important points for both methods but leave a few details out of the equation. Results are similar across the board but the Monte Carlo method seems to be performing consistently better in both metrics. Visually the result also looks more pleasing. However, we used fifty million iterations to achieve this result before stopping. This took roughly nine minutes to complete which certainly not ideal for all usecases.

Given the fact that there are very few hyperparameters and the model does not depend on any priors or external images, its results are not only good, but robust for real-world usage and generally stable.

The EPLL algorithm also gave us good results. The quality of the images is slightly worse than the Monte Carlo method but the results are satisfactory nonetheless. In addition, denoising the actual image took significantly less time - around two and a half minutes. This is significantly better than the previous method.

We need to address a few things here. First, the performance of the EPLL algorithm - with either optimisation method - is wildly dependent on a few hyperparameters of choice. In our experiments, we had to increase the stride of the patches to 3 to prevent using above 30GB of memory during the denoising process. This means that our results are worse than what their theoretical optimal would be. In addition, the quality of the output of the EPLL method is strongly reliant on the patch priors. In our testing we simply used the rest of Set12's images as our database of patches which is less than ideal. We also kept hitting walls with memory constraints due to the sheer amount of patches that needed to be fitted. Batching the fitted data is a good solution to this problem but we chose to stick with the traditional application of GMM. With more compute, access to larger amounts of memory, batching, and a much larger and more complete dataset the results will surely improve significantly. Lastly, fitting the GMM took almost an hour for our case - with 200 components fitted on 1,100,000 patches. Unless the prior patches are given, this could be another bottleneck in real-world applications as fitting a good prior on a representative dataset is a non-trivial task. Looking at Zoran and Weiss' results, our experiments are worse performing due to the lack of a proper diverse prior - we do also use much heavier noise than they used in their experiments, making restoration a much harder problem.

Concluding, both methods are incredibly flexible denoising tools. Markov Random Fields provide a deceptively simple framework for denoising images. It is a robust and easily interpretable method that can reliably denoise images up to a certain threshold. EPLL, on the other hand, is an involved method that requires much attention to both hyper-parameter tuning and the patch priors used. Its complexity, however, can be worth for its high potential. With a good prior the results can be much sharper than the soft and blurry results other methods produce. Coupled with its fast convergence rate, this method can be very powerful with the right conditions.

References

1. Zoran, D., & Weiss, Y. (2011). From learning models of natural image patches to whole image restoration.

2. Luo, Enming, et al. “Adaptive image denoising by mixture adaptation.” *IEEE Transactions on Image Processing*, vol. 25, no. 10, Oct. 2016, pp. 4489–4503, https://doi.org/10.1109/tip.2016.2590318.

3. Baydin, Atilim Gunes, et al. *Automatic Differentiation in Machine Learning: A Survey*, 2018, https://doi.org/10.48550/arXiv.1502.05767.

4. Kingma, Diederik P., and Jimmy Ba. “Adam: A Method for Stochastic Optimization.” *arXiv.Org*, 30 Jan. 2017, arxiv.org/abs/1412.6980.

5. Li, S. Z. (2009). Markov random field modeling in image analysis.

6. Geman, S., & Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images.

7. Levin, A., et al. (2009). Understanding blind deconvolution algorithms.

8. Y. Cao, Y. Luo and S. Yang, "Image Denoising With Gaussian Mixture Model," 2008 Congress on Image and Signal Processing, Sanya, China, 2008, pp. 339-343

9. T. -L. Chen, "A Markov Random Field Model for Medical Image Denoising," 2009 2nd International Conference on Biomedical Engineering and Informatics, Tianjin, China, 2009, pp. 1-6