Cleaning an image with GMM prior

Haggai Maron & Tal Amir

December 27, 2012

Part I

Derivation of the formula

We are working on 8×8 patches and assume that y is a noisy patch we see, x is the clean patch and k is the index of the Gaussian from which x was generated.
We introduce the following random variables:

$$X \in \mathbb{R}^{64}$$

$$Y \in \mathbb{R}^{64}$$

$$K \in \{1, ..., k\}$$

$$X|K = j \sim N(\mu_k, \Sigma_k)$$

$$Y = X + N$$

$$N \sim N(0, \sigma^2 I_{64})$$

$$K = j \; w.p \, \pi_k$$

We would like to use this model in order to get an estimation of the clean image x given the noisy patch y. If we use the MMSE estimation we know that

$$x_{MMSE} = E[X|Y = y] = \int_{\mathbb{R}^{64}} x \cdot f_{X|Y}(x|y)$$

We try to simplify this formula using the following identities:

$$f_{X|Y}(x|y) = \frac{f_{X,Y}(x, y)}{f_Y(y)}$$

$$f_{X|Y,K}(x|y, j) = \frac{f_{X,Y,K}(x, y, j)}{f_{Y,K}(y, j)}$$

$$f_{K|Y}(j|y) = \frac{f_{Y,K}(y,j)}{f_Y(y)}$$

so

$$f_{X|Y,K}(x|y,j) \cdot f_{K|Y}(k|y) = \frac{f_{X,Y,K}(x,y,j)}{f_Y(y)}$$

which implies

$$\sum_{j=1}^{k} f_{X|Y,K}(x|y,j) \cdot f_{K|Y}(j|y) = \sum_{j=1}^{k} \frac{f_{X,Y,K}(x,y,j)}{f_Y(y)} = \frac{f_{X,Y}(x,y)}{f_Y(y)} = f_{X|Y}(x|y)$$

so we got

$$x_{MMSE} = \int x \cdot f_{X|Y}(x|y)dx =$$

$$\sum_{j=1}^{k} \left[f_{K|Y}(j|y) \int x \cdot f_{X|Y,K}(x|y,j)dx \right] =$$

$$\sum_{j=1}^{k} \left[f_{K|Y}(j|y) \cdot E[X|Y=y,K=j] \right]$$

and

$$E[X|Y=y,K=j] =$$

$$\mu_j + \Sigma_j \left[\Sigma_j + \sigma^2 I_{64} \right]^{-1} (y - \mu_j) =$$

$$\mu_j + \frac{1}{\sigma^2} \left[\frac{1}{\sigma^2} I + \Sigma_j^{-1} \right]^{-1} (y - \mu_j)$$

So we are left with calculating the following quantity: $f_{K|Y}(j|y)$:

$$f_{K|Y}(j|y) = \frac{f_{Y|K}(y|j) f_K(j)}{f_Y(y)}$$

We know that

$$Y|K = j \sim N(\mu_j, \Sigma_j + \sigma^2 I_{64})$$

so

$$f_{Y|K}(y|j) = N\left(y; \mu_j, \Sigma_j + \sigma^2 I_{64}\right) =$$

$$\frac{1}{(2\pi)^{32} \det\left(\Sigma_j + \sigma^2 I_{64}\right)^{\frac{1}{2}}} \exp\left(-\frac{1}{2} (y - \mu_j)^T \left[\Sigma_j + \sigma^2 I_{64} \right]^{-1} (y - \mu_j) \right)$$

and the only thing we need to calculate is $f_Y(y)$ which is given by

$$f_Y(y) = \sum_{j=i}^{k} f_{Y|K}(y|j) f_K(j)$$

We shall use our results in order to obtain the expression for $\hat{x}_{MMSE}$:

$$\hat{x}_{MMSE} = \sum_{j=1}^{k} \left[f_{K|Y}(j|y) \cdot E[X|Y=y, K=j] \right] =$$

$$\sum_{j=1}^{k} \left[\frac{f_{Y|K}(y|j) f_K(j)}{f_Y(y)} \cdot \left(\mu_j + \frac{1}{\sigma^2} \left[\frac{1}{\sigma^2} I + \Sigma_j^{-1} \right]^{-1} (y - \mu_j) \right) \right] =$$

$$\frac{1}{f_Y(y)} \cdot \sum_{j=1}^{k} \left[f_{Y|K}(y|j) f_K(j) \cdot \left(\mu_j + \frac{1}{\sigma^2} \left[\frac{1}{\sigma^2} I + \Sigma_j^{-1} \right]^{-1} (y - \mu_j) \right) \right] =$$

$$\frac{1}{f_Y(y)} \sum_{j=1}^{k} \left[\pi_j \cdot N\left(y; \mu_j, \Sigma_j + \sigma^2 I_{64}\right) \left(\mu_j + \frac{1}{\sigma^2} \left[\frac{1}{\sigma^2} I + \Sigma_j^{-1} \right]^{-1} (y - \mu_j) \right) \right]$$

Where

$$f_Y(y) = \sum_{s=1}^{k} \pi_s \cdot N\left(y; \mu_s, \Sigma_s + \sigma^2 I_{64}\right)$$

Note that the formula

$$\hat{x}_{MMSE} = \sum_{j=1}^{k} \left[f_{K|Y}(j|y) \cdot E[X|Y=y, K=j] \right]$$

makes sense - it is a weighted average of the conditional means according to a specific Gaussian, where the weight is the responsibility of the Gaussian to the observed patch y.

Part II
Results

We have tested this Algorithm on 7 patches: 2 randomly chosen from a natural image, two were generated from the GMM learnt distribution and 2 were synthetically generated by us.
For every patch we present a 2 row image of the following structure:

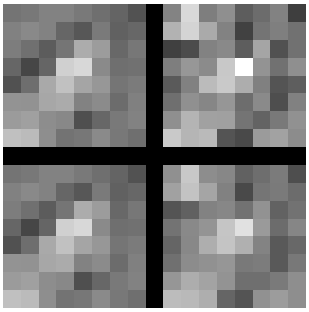
$$\begin{bmatrix} original & noisy \\ original & denoised \end{bmatrix}$$

We added noise with 'imnoise' function using $\sigma = 0.1$

1 Natural patches

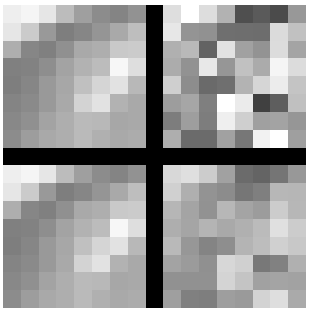
We used two patches from a natural image [peppers.png] and used the algorithm on them.

1.1 peppers



Noisy patch PSNR: 18.391600 MSE: 941.716903
Denoised patch PSNR: 20.305012 MSE: 606.148722

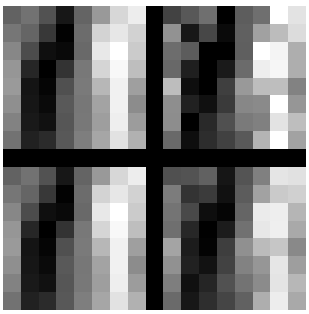
1.2 peppers2



Noisy patch PSNR: 21.396915 MSE: 471.399222
Denoised patch PSNR: 25.380485 MSE: 188.378731

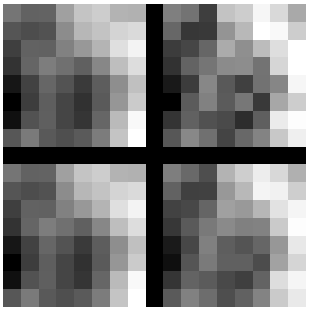
2 Patches from the learnt distribution

2.1 MoG1



Noisy patch PSNR: 20.121831 MSE: 632.262255
Denoised patch PSNR: 22.948736 MSE: 329.766651

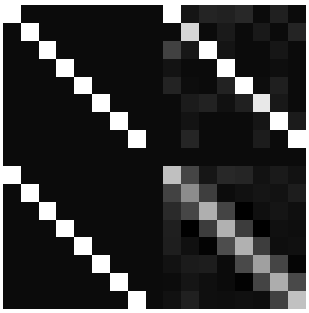
2.2 MoG2



Noisy patch PSNR: 20.207040 MSE: 619.978065
Denoised patch PSNR: 22.283907 MSE: 384.317046

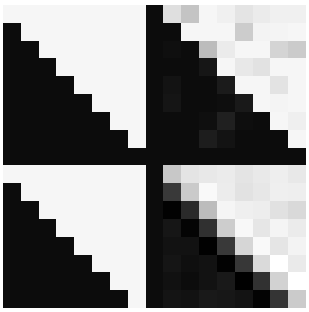
3 Synthetic patches

3.1 ID



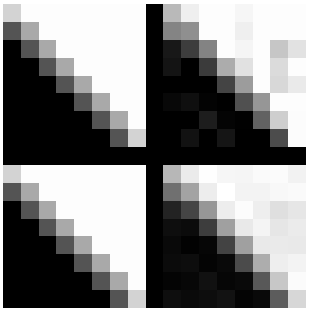
Noisy patch PSNR: 24.545944 MSE: 228.289529
Denoised patch PSNR: 15.558663 MSE: 1808.065149

3.2 Upper Triangular



Noisy patch PSNR: 23.824859 MSE: 269.522175
Denoised patch PSNR: 20.408397 MSE: 591.889534

3.3 Smooth upper triangular



Noisy patch PSNR: 22.886766 MSE: 334.505860
Denoised patch PSNR: 24.863643 MSE: 212.185667

Part III
Discussion

It looks like the learnt prior's ability to denoise both natural patches and patches that were generated from the distribution itself is good. Its performance on patches that we generated for the test [i.e. the id patch and upper triangular patch] was not good and the algorithm actually degraded the noisy patch by adding 'smoothness' to it. In order to test this we added another test on an upper triangular matrix that was convolved with a blur kernel, and on this patch [smooth upper triangular] the algorithm worked well [so it indeed looks like our prior prefers smooth patches in some way].