# Median Filtering

We generate two 400\*400 artificial examples, one stripe picture and one checkerboard picture

Stripes, both with width 40.

We first add 10% salt-and-pepper noise on them and then implement median filtering with window size 7. Picture la.png is added Gaussian noise with sigma 25 and also denoised with median filtering. The results are shown below.

|  |  |  |
| --- | --- | --- |
|  | Noise picture | Denoised picture |
| stripe | **PSNR: 25.5399** | **PSNR: 38.8138** |
| checkerboard | **PSNR: 25.7351** | **PSNR: 32.8901** |
| La.png | **PSNR: 36.6663** | **PSNR: 31.9778** |

We can see for artificial examples with salt-and-pepper noise, median filtering remarkably removed the noise. PSNR increased after denoising. But for real picture la.png with Gaussian noise, it heavily blurred the picture and even decreased the PSNR.

Then we try different sigma for la.png. Generally, median filtering does denoise a bit for Gaussian noise with large sigma. But for smaller sigma, median filtering decreases PSNR more and results in a worse picture. Different window size in median filtering does not change the result significantly. We use window size 5. The results are shown below.

|  |  |  |
| --- | --- | --- |
| Sigma | Noise picture | Denoised picture |
| 0.1 | Inf | 34.9717 |
| 1 | 63.8336 | 34.9732 |
| 10 | 44.2157 | 35.0359 |
| 100 | 24.3810 | 28.2339 |
| 1000 | 14.7166 | 15.8785 |

# MRF-based Denoising with Gradient Ascent

First, we test with a 10\*10 small checkerboard artificial picture. With sigma chosen to be 10, 25 and 100, we can see the iteration needed to achieve the convergence increased as sigma increased. Below is the log-posterior picture curve for eta=1 and 3 different sigmas within 1000 iterations.

|  |  |  |  |
| --- | --- | --- | --- |
| Sigma | 10 | 25 | 100 |
| Log-posterior |  |  |  |

For sigma=100, eta=1, log-posterior is far from convergence within 1000 iterations.

For the same sigma, larger eta leads to fast convergence, but might produce unprecise results. Below is the observation for sigma=25 and 4 different etas within 1000 iterations.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| eta | 0.1 | 1 | 10 | 100 |
| Log-posterior |  |  |  |  |
| Lp final | -1459 | -961.2359 | -961.2708 | -961.2708 |

Implementing MRF to the same noise images as Task 2, we fixed sigma as 25 and eta as 1, we get the PSNR with MRF. The table below shows PSNR with median filtering and MRF.

|  |  |  |  |
| --- | --- | --- | --- |
| PSNR | Median filtering | MRF | Log-posterior curve |
| Salt-pepper stripe | 38.8138 | 21.1651 |  |
| Salt-pepper checkerboard | 32.8901 | 19.3212 |  |
| Gaussian La.png | 31.9778 | 29.7498 |  |

By checking the log-posterior curve we can see it’s reached the convergence, but the results are worse than median filtering, especially for artificial pictures. Because the likelihood model in MRF is based on Gaussian noise, to implement it to denoise salt-pepper noise would definitely lead to worse results. Besides, obviously the artificial examples do not correspond to Gaussian prior, so MRF model does not apply to them. For la.png with Gaussian noise, since the prior picture does not correspond to Gaussian distribution, the result is also not good.

If we initialize gradient ascent with the output of median filtering, the table below shows the different converging curve for log-posterior.

|  |  |
| --- | --- |
| PSNR | Blue: initialize with median-filtering  Red: initialize with noise image |
| Salt-pepper stripe  M: 21.3038  N: 21.1823 |  |
| Salt-pepper checkerboard  M: 19.4835  N: 19.3934 |  |
| Gaussian La.png  M: 29.6666  N: 29.7498 |  |

We can see for artificial pictures, initializing with median-filtered image won’t bring faster convergence. Actually, it brings a lower starting point and in turn slows down the converging speed. For real image la.png, however, median-filtering does help speed up the convergence, though they all converge to generally the same position in the end, with almost no effect for the final performance.

# A Different Prior

Stripes, eta=10, PSNR=25.4408



Checkerboard, eta=10, PSNR=25.4891



La.png

Eta = 1

PSNR=38.6504





Eta =10

PSNR = 38.5481



 

Gradient of the log-prior

 



# Independence Assumption

Independence assumption is reasonable, since it’s easy to tackle with and often brings not bad results. But in reality, most noise is obviously dependent, one pixel is almost always related to those around it, so noise is very likely to cluster.

For dependent noise, we add Gaussian noise to part of la.png so that the noise looks like stripes. Then we implement MRF for this noise image, the table below shows the results.

|  |  |  |
| --- | --- | --- |
|  | Images | PSNR |
| Noise image |  | 38.5510 |
| Gaussian prior |  | 28.3770 |
| Student prior |  | 32.3667 |

# Color image

For color images, the RGB image can be first transferred to Y Cb Cr channel, then apply the same denoising method on the Y channel, and back transfer the denoised Y channel and the original Cb and Cr channel to RGB image. Since Y channel represents the light intensity, which is most sensitive to human eyes.