Image Processing Coursework - Non-local Means Denoising

mbtj48

1 Description

Generally, non-local means denoising uses multiple squared neighbourhoods "similar" to a given neighbourhood of pixels, and then calculates the value of the pixels in that neighbourhood based on the weighted average of these similar neighbourhoods. This leaves two main areas where implementation becomes difficult: finding similar neighbourhoods in the image and how specifically to average them.[6]

Formally, non-local means is defined as:

$$NL[u](p) \;=\; \frac{1}{C(p)} \int_{\Omega} f(p,q)\; u(q) dq \qquad [1,\ 2]$$

Here we have a filter applied to the image u at a pixel p which lies within the image area Ω. Where, $u(q)$ is the original unfiltered pixel value at position q and $C(p)$ is the normalising factor of the function. Finally, we have the function $f(p,q)$, which calculates the weight of the similarity of this pixel to other neighbourhoods, this is commonly the Gaussian weighting function which uses a normal distribution of weightings relating the two pixel values. This typically acts as the Euclidean distance between the two pixels: $d(\,B(p),\,B(q)\,)$. [2] Due to this algorithm's continuous nature we cannot extrapolate a direct implmentation from this definition, so we have to manipulate the algorithm to use discrete pixel values.

2 Implementations

The pixelwise implementation introduced in [1] considers every pixel and its neighbourhood in the image, calculating the weight based on the squared Euclidean distance between two pixels which acts as the direct similarity between the colour vectors. These are then entered into an exponential kernel, providing similar pixels with a higher weight and weights smaller than $2\sigma^2$ are set to 1, so they are deemed not similar at all. The values found from this update the center pixel of each neighbourhood. (Where modifications include limiting the search area, which is essential for the deafult pixelwise implementation as well as preventing repeat calculations. These will be expanded in section 5). The formula for applying pixelwise to a image $u = (u_1, u_2, u_3)$, at a given pixel p is:

$$u_i(p) = \frac{1}{C(p)} \sum_{q \in B(p,r)} u_i(q)\, w(p,q) \qquad [2]$$

Clearly, it has large similarities to the integral definition at the start of the report. With $C(p)$ acting as the normalising function, $u_i(q)$ as the unfiltered value of another pixel q at the center of another patch. Finally $w(p,q)$ as the weight between the two pixels, defined earlier.

The patchwise implementation introduced in [2] works very similarly to the pixelwise implementation, however after calculating how similar other neighbourhoods are, it denoises all of the neighbourhood of p rather than just p itself. This is done by finding the average of every pixel within a neighbourhood first, and is then repeated across all possible neighbourhoods which then gives N^2 estimates for every pixel, these are then averaged to calculate the final value for every pixel. Formally, the additional formula for patchwise over pixelwise is:

$$u_i(p) = \frac{1}{N^2} \sum_{Q=Q(q,f)|q \in B(q,f)} Q_i(p) \qquad [2]$$

Separable NLM is a highly efficient implmentation, which separates the image into row and column vectors then processes the whole vector at once. This updates the formula for the euclidean distance to

$$d^2_{ij} = \begin{cases} \overline{V}(l_{ii}) + \overline{V}(l_{jj}) - 2\overline{V}(l_{ij}) & i \geq j \\ \overline{V}(l_{ii}) + \overline{V}(l_{jj}) - 2\overline{V}(l_{ji}) & j > i \end{cases} \qquad [3]$$

In order to calculate the output of the image the formula above is used to process rows then columns, while also doing columns then rows, as the process is not commutative, and find the average of the two results. See Figure 3 in [3]. That paper then further talks about finding optimal parameters and proofs of the complexity.

Efficiencies

The initial pixelwise implementation described above has a completxity of: $\mathcal{O}(N^4K^2)$ for an image with dimension $N \times N$ However, the common default pixelwise implementation limits the research window for similar pixels, which improves the complexity to: $\mathcal{O}(N^2S^2K^2)$ [3] where S is the size of the search window and K is the size of the neighbourhood. Patchwise is less efficient with a complexity of more like: $\mathcal{O}((N^2)^2S^2K^2{+}N^2) \rightarrow \mathcal{O}(N^4S^2K^2)$. This is because patchwise performs an extra N^2 calculations to find the weight across every neighbourhood which are then averaged at the end. However, the extra computation for patchwise generally results in a higher PSNR value [2] when compared to the initial pixelwise implementation in [1]. Alternatively, many minor enhancements have been found for these implementations; for example, the separable NLM described above has a complexity of $\mathcal{O}(N^2S)$. Therefore making it linear in calculating the new value for every pixel. Generally this increase in speed

is a success for PSNR performance over similar, more complex algorithms. See the images and performance stats in [3]

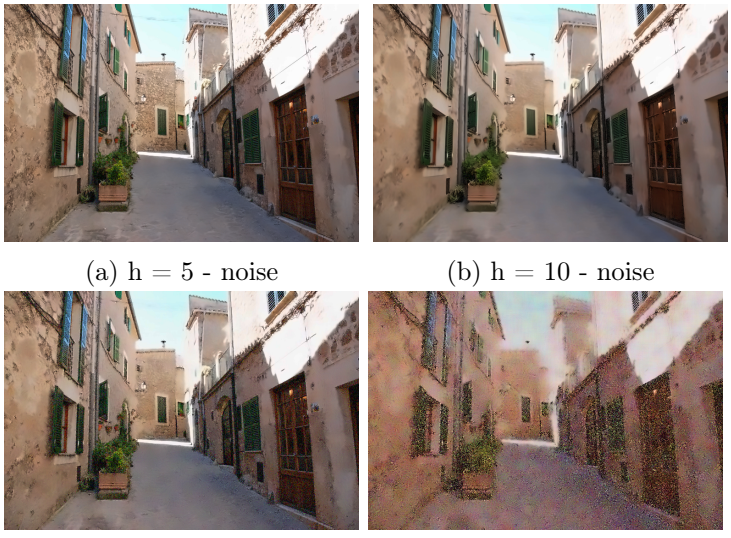
3 Parameters

(a) Original (b) Noise

(c) High noise (d) Extreme noise

Figure 1: Original Images

In order to compare the given parameters h, templateWindowSize, and searchWindowSize, I will fix two at their recommended values of 10,7,21 respectively, and alter the third value with a low value, a recommended value and a high value for each noisy image. This provided me with 28 images to compare from each original image. See below for the original images:

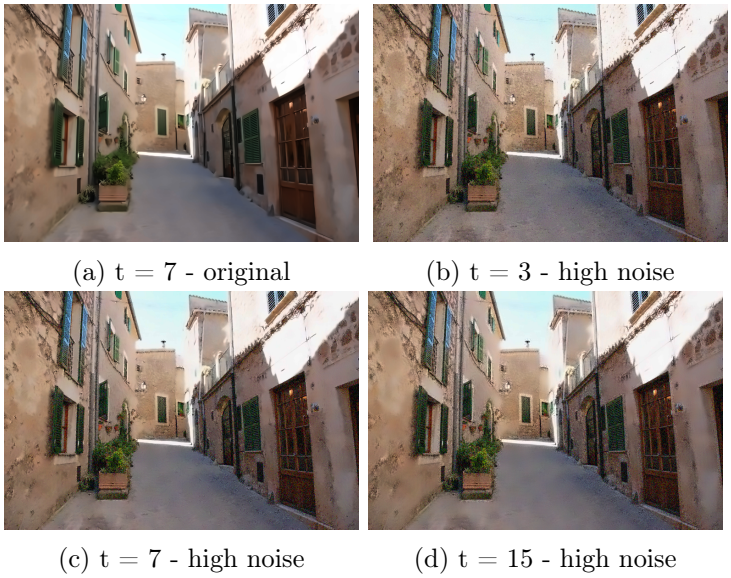


(a) h = 5 - noise (b) h = 10 - noise

(c) h = 10 - high noise (d) h = 30 - ext noise

Figure 2: Varied h Images

Starting with parameter h, which is the strength of the gaussian filter[4]. This provides the largest change to 1a. I tested values of 5,10,30 and some instances of 20. Initially, at values of 5, changes are not noticable when applied to 1d, this is evident through calculating the PSNR value for the image, which shows a 0.006 improvement. However improves when applied to 1c, completely removing noise from the sky and removing noise from some of the sunlit buildings, but gives a worse PSNR value by 0.1. Using the same h futher removes the noise from 1b and retains detail when applied to 1a (See image 2a). Increasing this value to 10 starts to lose detail when applied to low or no noise images and cannot improve any noise in the extreme noise image, however almost entirely removes noise from the high noise image (See image 2c). Further, this shows one of the largest increases in PSNR moving from 28.60 to 29.65. Moving on to values of 30 just provides extremes of using a value of 10, lots of detail is lost even on the high noise image, although it can successfully remove some noise from the extreme noise image but the image lacks far too much in quality (See image 2d).



(a) t = 7 - original (b) t = 3 - high noise

(c) t = 7 - high noise (d) t = 15 - high noise

Figure 3: Varied *templateWindow* Images

Now altering the templateWindowSizes through 3,7,15. This affects the size of the local neighbourhood of similar pixels [4] and affect the image quite clearly, with lower values resulting in higher detail within similar areas. This is because the neighbourhood is smaller and will smooth out each neighbourhood less. See 3a where this removes some minor detail from 1a. Moving onto image 3c, produced with a neighbourhood of size 7; it is clearly close to the optimal value for high noise images, especially compared to image 3b which uses a neighbourhood of size 3, and improvements from the original are hard to notice. However, the PSNR value improves massively from 28.60 to 29.71 and therefore the image has still improved. Altering the neighbourhood sizes on the extreme noise image shows nearly no affect and therefore little improvement after denoising.

(a) s = 45 - original

(b) s = 11 - noise

(c) s = 11 - high noise

(d) s = 21 - high noise

Figure 4: Varied *searchWindow* Images

I finally iterated the searchWindowSize through 11,21,45. This is the proximity in which the denoising will search for similar neighbourhoods and is typically much smaller than the whole image to speed up computation to produce similar results. [4] Generally, the higher values resulted in greater smoothing of surfaces, however with a value of 45, I felt it removed too much detail across all noisy images (See Image 4a). In image 4b, the noise is blurred however similar areas have a "brushed" effect, which is the side affect of having a small search window for finding similar neighbourhoods Image 4d shows how a search window of 21 affects image 1c; clearly it keeps enough detail in rock faces wihout smoothing the floor surfaces, which is almost perfect considering the noise in image 1c.

4 Comparisons

Gaussian Filtering - The image is put through a gaussian filter, which primarily acts in removing noise through blurring. It does this by looking at the neighbourhood of a pixel and assigns the new intensity of that pixel based on the difference of the surrounding pixels intensity, passed through a gaussian distribution matrix (the filter) [6]. Typically, this does not perform as well as non local means (In [1], they find the mean square error to be just less than double for NLM) but is much more efficient at calculating the output image. [1]

(a) Noise - PSNR 31.61

(b) High noise - PSNR 30.14

Figure 5: A 5×5 gaussian filter applied to 1b & 1c

Above, I applied the cv2.GaussianBlur function to the same alley image from 1a. Clearly, it does perform well at removing some noise however detail is quite heavily lost when the filter size is increased. If I compare the PSNR value for 5a to the NLM, it lies in the average found by the range of parameters I used, with the best being 31.9. For 5b it out performs the NLM PSNR values for the parameters I used, however I feel the images look better after the NLM denoising, for example see image 3c.

Anisotropic Filtering [1] - Anisotropic filtering aims to improve the blurring caused by Gaussian filtering by convolving the image at a point u only in the direction orthogonal to $Du(x)$. This is defined as:

$$AF_h u(x) = \int G_h(t) u(x + t\frac{Du(x)^{\perp}}{|Du(x)|})dt \quad [1]$$

Generally, for reducing noise in the image, this is more computationally efficient that NLM, however NLM performs higher when improving PSNR. For example, see the mean square error table from [1] which shows that anisotropic filtering performs better than the gaussian filter, slightly, but still cannot perform anywhere near NLM.

Spacial Median Filtering [6] - The algorithm iterates through every pixel in the image, and exchanges its value to the median value of pixels in its neighbourhood. This is a local filter, and generally is much more efficient than the NLM variants, but typically does not perform as well at improving PSNR.

(a) Original - PSNR 32.19

(b) High noise - PSNR 30.14

Figure 6: A 5×5 median filter applied to 1a & 1c

Above, I applied the cv2.medianBlur function to the alley image from 1a. It performs similarly to the Gaussian blur in terms of PSNR values, however looking at 6b, certain edges have been removed as they are extremes in those neighbourhoods. This could be combined with an edge sharpening algorithm such as the laplacian in order to perform better visually and therefore perform better than NLM.

Spacial Mean Filtering [6] - This filter is similar to spacial median filtering, however instead of finding the median of the region of interest, we find the mean. Similar to the other algorithms, this aims to reduce noise through smoothing and blurring, and therefore loses a lot of detail with medium to large size filters.

(a) Original - PSNR 30.37 (b) Ext noise - PSNR 28.46

Figure 7: A 5×5 mean filter applied to 1a & 1d

The images above show how the 5×5 mean filter affects the alley image. Evidently, a lot of detail has been lost through too much blurring, however I feel this is the best evidence for improvements on image 1d; this is further evident when looking at PSNR values which are not rivalled with NLM, but perform much worse on images with less noise.

For consistency, I kept the filter sizes the same for all of the examples above, however in reality the optimal filter sizes are much smaller than 5.

Deep Neural Network for Image Denoising [5] - Recently, progression in the denoising research field has moved towards denoising using unsupervised neural networks. A recent version seen in [5], uses 3 different layers in the neural network. A convolution and ReLU layer, a convolution (consists of many hidden layers in the network which convolve with following layers), batch normalisation (a method used to improve speed and performance of deep learning neural networks) and ReLU (the output function to active certain nodes in the network) layer, and finally just a convolutional layer. See Figure 1 in [5] for the diagram for the network. This proposed algorithm, DnCNN, uses a residual learning method to train a residual mapping of $\mathcal{R}(y) \approx v$ (from the noise function $y = x + v$), which then gives $x = y - \mathcal{R}(y)$. This then uses a residual mean square error formula between desired residual images and estimated ones from noisy input, in order to backpropagate and train the layers of the neural network. Although there is little material directly comparing this to NLM, it performs much much higher on the 12 commonly used testing images in Figure 3 from [5], however this algorithm requires a long period of time for training before it can perform really well.

5 Modifications

Generally, modificatations are well documented and varied. Firstly, paper [7] introduces probabilistic early termination to elimate patches which are not similar to the current patch. This is done by calculating the partial sum of euclidean distances and determine with x probability whether x exceeds a threshold and terminate the current computation. Further, see paper [8], which uses the preclassification of a search window to determine whether the window is in an edge section or a surface/ textured section. From this, the algorithm then decides whether it should use NLM or local mean this improves both the run time with a minor increase in PSNR performance. Also, see [9], which uses singular value decomposition to affectively factorise the matrix containing the region of interest, allowing the columns of V to act as a basis for the space of image patches. From this we can also use the SVD property to approximate the patch in order to speed up NLM through eliminating the fraction of the pixels that yield the largest norm. The results in [9], show that this method performs higher than the prefiltering modification and the limited range modification.

6 Applications

References

[1] Buades, Antoni, Bartomeu Coll, and J-M. Morel. "A non-local algorithm for image denoising." 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). Vol. 2. IEEE, 2005.

[2] Buades, Antoni, Bartomeu Coll, and Jean-Michel Morel. "Non-local means denoising." Image Processing On Line 1 (2011): 208-212.

[3] Ghosh, Sanjay & Chaudhury, Kunal. (2016). Fast separable nonlocal means. Journal of Electronic Imaging. 25. 023026. 10.1117/1.JEI.25.2.023026.

[4] OpenCV - Image denoising https://docs.opencv.org/3.4/d5/d69/ tutorial_py_non_local_means.html

[5] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. "Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising" IEEE Transactions on Image Processing 2017: 3142 - 3155

[6] Ioannis Ivrissimtzis, Image Processing, Lecture 4 & 5, Image Noise & Spacial Filtering

[7] Ramanathan Vignesh, Byung Tae Oh and C.-C.Jay Kuo, "Fast Non-Local Means (NLM) Computation With Probabilistic Early Termination"

[8] Hernández-Gutiérrez, I.V., Gallegos-Funes, F.J. & Rosales-Silva, A.J. Improved preclassification non local-means (IPNLM) for filtering of grayscale images degraded with additive white Gaussian noise. J Image Video Proc. 2018, 104 (2018) doi:10.1186/s13640-018-0346-y

[9] Jeff Orchard, Mehran Ebrahimi, Alexander Wong, "Efficient nonlocal-means denoising using the SVD"

[10] Singh, P., Shahnawazuddin, S. & Pradhan, G. Circuits Syst Signal Process (2018) 37: 4527. https://doi.org/10.1007/s00034-018-0777-9

[11] Pierrick Coupé, Pierre Yger, Sylvain Prima, Pierre Hellier, Charles Kervrann, et al.. An optimized blockwise nonlocal means denoising filter for 3-D magnetic resonance images.. IEEE Transactions on Medical Imaging, Institute of Electrical and Electronics Engineers, 2008, 27 (4), pp.425-41.

Appendices

(1a)

(1c)

(1b)

(1d)