Pattern matching based denoising for images with repeated sub strucutures.

Alice Author[1,*], Bob Author[2], Christine Author[1,2,+], and Derek Author[2,+]

[1]Affiliation, department, city, postcode, country
[2]Affiliation, department, city, postcode, country
[*]corresponding.author@email.example
[+]these authors contributed equally to this work

ABSTRACT

Example Abstract. Abstract must not include subheadings or citations. Example Abstract. Abstract must not include subheadings or citations. Example Abstract. Abstract must not include subheadings or citations. Example Abstract. Abstract must not include subheadings or citations. Example Abstract. Abstract must not include subheadings or citations. Example Abstract. Abstract must not include subheadings or citations. Example Abstract. Abstract must not include subheadings or citations.

Please note: Abbreviations should be introduced at the first mention in the main text – no abbreviations lists. Suggested structure of main text (not enforced) is provided below.

Introduction

Transmission Electron Microscopy (TEM) imaging has been helpful to solve numerous scientific questions in life and material sciences[1,2] . However, at times the noise in the acquired images corrupt the signal beyond a useful level. Noise in images can appear due to intrinsic reasons like problems in the sensors or the digital circuits, or due to external factors like the environment. Image denoising plays a vital role especially in suppressing microscopy image noise.

The TEM imaging contrast is based on the interaction between specimen and a multi keV electron beam. While the optical resolution for modern TEM system can be below one angstrom, the high energy electrons often lead to a fast degradation of the sample. For such samples only few electrons can be used for imaging. This leads to a degradation of the images resolution, thus making the image hard to interpret. Therefore, after image acquisition, numerical processing is required to enhance the image quality.

To get maximal image quality with minimal dose, various image denoising algorithms were proposed in the past. Conventional denoising methods[3,4] use only noisy images for the denoising task, whereas most modern methods involving a deep neural network[5,6] require clean images as ground truths for training. Since clean electron microscopy images are not available in most cases, modern denoising methods that require clean images as ground truths cannot be used. However, there are also some deep neural network based methods that use noisy supervision[7] and self supervision[8] . Although, conventional and self-supervised methods have shown success in denoising images, these methods improved the denoising quality only by a relatively small margin for images with repeated patterns. Hence, a new denoising algorithm is proposed and its results are analyzed and compared with the state-of-the-art methods showing significant gain in image quality.

Method

The proposed denoising algorithm identifies similar patches within the entire image and averages them to suppress the noise that is randomly distributed. Since the noise is assumed to be having zero mean, it cancels out when multiple patches are averaged. However the base signal remains the same throughout and averaging does not disrupt the signal. Hence combining different patches result in the denoised image, close to the actual signal value.

Let x_i be the i^{th} noisy patch, k be the number of patches that are averaged, s_i, and n_i be the signal and noise in the i^{th} patch respectively. Then,

$$x_i = s_i + n_i \tag{1}$$

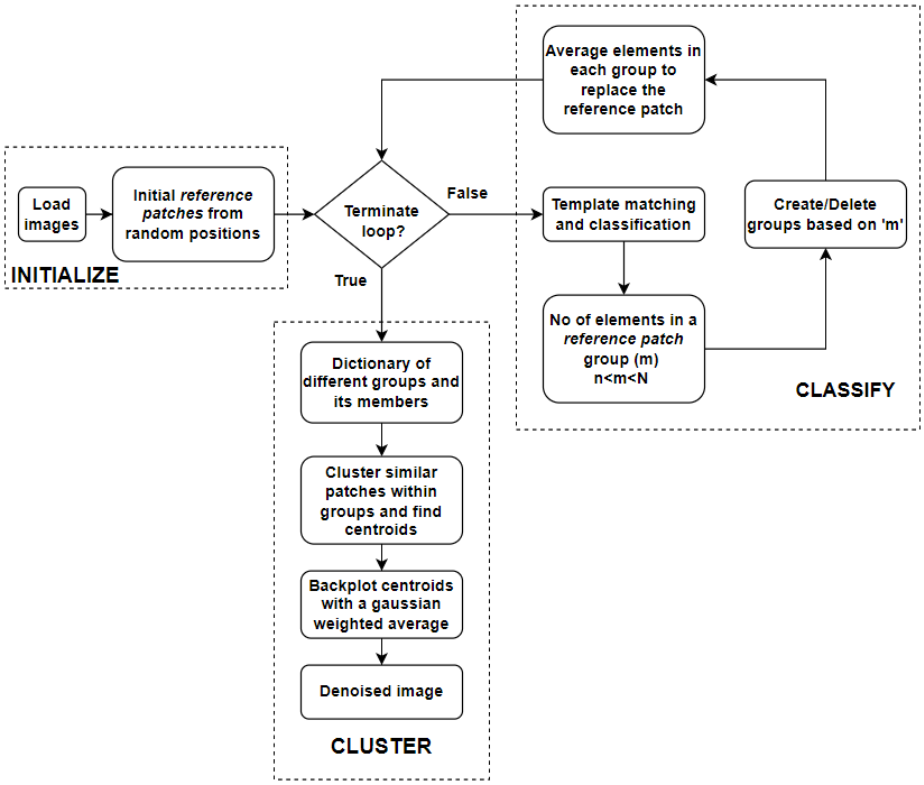

Figure 1. Flowchart of the algorithm

Averaging over k patches results in an expected value,

$$E[\frac{1}{k}\sum_{i}^{k}x_i] = E[\frac{1}{k}\sum_{i}^{k}(s_i+n_i)] \tag{2}$$

Since the noise is expected to be having zero mean and the base signal is expected to be the same, this ideally means that,

$$E[\frac{1}{k}\sum_{i}^{k}x_i] = s \tag{3}$$

Hence the variance of the averaged patch is small, i.e,

$$Var[\frac{1}{k}\sum_{i}^{k}x_i] = \frac{1}{k^2-1}\sum_{i}^{k}Var(n_i) \tag{4}$$

since $Var(s_i) = 0$ for all i. Hence averaging patches with the same signal suppresses noise.

Outline of the proposed algorithm

The proposed algorithm groups similar patches at two levels. Normalized convolution is used to broadly group similar patches within an image and clustering is used to more finely group closely matching patches within the groups obatined during the first step. The flowchart of the algorithm is shown in figure 1 and the two levels of pattern matching is represented by 'classify' and 'cluster' sections of the flowchart.

Normalized convolution between a template and an image results in local cosine similarity. Cosine similarity measures similarities between two vectors[9] by finding the cosine of the angle between them. If the vectors are in the same direction (i.e., similar), cosine similarity is maximal. It is mathematically represented as,

$$similarity = cos(\theta) = \frac{A\cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n}A_iB_i}{\sqrt{\sum_{i=1}^{n}A_i^2}\sqrt{\sum_{i=1}^{n}B_i^2}} \tag{5}$$

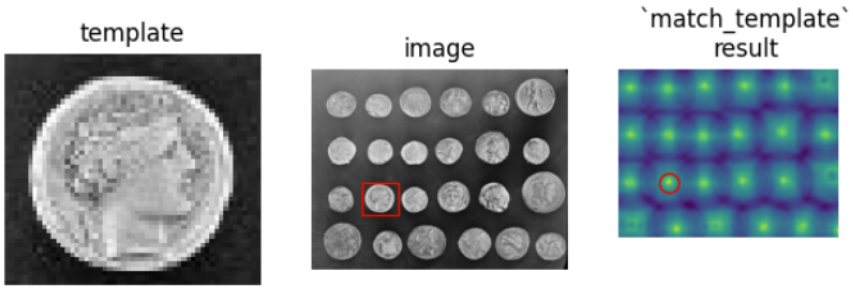

Figure 2. Template matching example [1]

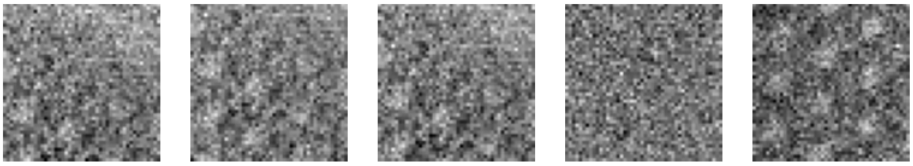

Figure 3. Initial *reference patches* (of size 46*46 pixels), taken from different regions of the input image

where A and B are two vectors, and θ is the angle between them. The cosine similarity value lies between -1 and 1. This is similar to the result obtained by template matching , which is represented in figure 2. An image, a template taken from the image, and the corresponding result can be seen in figure 2. The maximum value in the result which is marked in red, represents the template's location in the image. It corresponds to the top-left corner of the template. Values close to the maximum represent the patches similar to the template. Hence cosine similarity can be used to match patches with images, similar to template matching.

The time complexity of the convolution operation is $O(m^2n^2)$, where $m*m$ is the size of the patch and $n*n$ is the size of the image, which is for large m worse than the run-time of the template matching algorithm. Complexity of the template matching algorithm is $O(n^2log(n^2))$[10], where $n*n$ is the image size. Since the convolution operation is widely used in convolutional neural networks, there are Python libraries that support GPU computation for performing the convolution operation. Running the computations on the GPU makes the algorithm much faster.

The algorithm begins with the initialization of random patches of size $m*m$, which are used for matching other patches of size $m*m$ in the image. The patches that are used as templates for matching are referred to as *reference patches*. One example for the initial choice of *reference patches* can be seen in figure 3.

In the 'classify' section shown in figure 1, the patches at every position in the image are classified into different groups based on the *reference patch* using cosine similarity. When cosine similarity is applied, each patch of size $m*m$ in the image is compared with every other $m*m$ *reference patch*. The result has values between -1 and 1 as the results are normalized. The location with a perfect match is represented by value 1. The patches similar to the *reference patch* are represented by values close to 1.

Cosine similarity is carried out with all n *reference patches* and all the results are stacked into an array. From this array, the best fitting reference patch index at every position can be determined. Now, the patches in the image that are most similar to the *reference patches* are identified and grouped together.

In the next step, the newly formed groups are deleted or split into finer groups based on the group size. The reasoning behind this process is that groups with few members contribute hardly to any denoising, while overly large groups might lead to a loss of detail. Finally, for each group the old *reference patch* is replaced by the average of all the members in that group. In the next iteration, cosine similarity and the classification steps repeat with these new *reference patches*. We choose to terminate the "classify" section when the total number of reference patches in three consecutive iterations remains the same. Figure 4 shows the reference patches generated after 15 iterations. On comparing figure 1 and figure 4, one can observe that the noise in the final reference patches has been significantly suppressed.

If the final *reference patches* are used for back plotting (i.e. to replace the patches in their corresponding groups), there are still some artifacts (as shown) present because of the following reasons.

- There can be patches in a group that are less similar to the *reference patch*. When these dissimilar patches are averaged,

[2] https://scikit-image.org/docs/dev/auto_examples/features_detection/plot_template.html

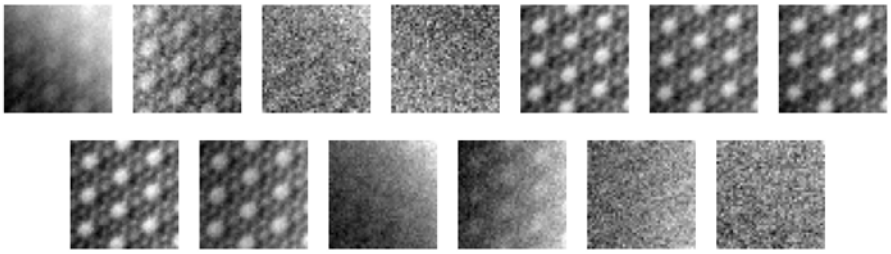

Figure 4. Final *reference patches*

the average might differ from the actual patch by a large extent.

- Cosine similarity is only sensitive to the structure for any two patches and ignores a constant positive (brightness). Therefore, back plotting might not recover local brightness variations.

These problems can be solved by averaging over a small group with very closely matched patches. To achieve this, clustering is applied within every group (represented by the final *reference patch*) to create smaller subgroups. The number of clusters in a group can be adjusted by a user set parameter. In other words, the signal-to-noise ratio can be adjusted by changing this parameter value. While previously the whole group was represented by a single *reference patch*, it is now represented by the centriods of the subgroups due to the clustering. Centroids are back plotted with a $2D$ Gaussian weighted average. These Gaussian weights smoothen the edges of the centroids, thus preventing artifacts in the reconstructed image.

A pseudo implementation of the algorithm is shown in the additional information section 1. The implementation of the denoising algorithm can be found on github[3].

Parameters of the algorithm and stability

The algorithm requires a few parameters which the users can tune in order to get optimal performance. The parameters include size of the features defined by patch size, position of the initial patches and depending on the amount of denoising desired, the upper and lower limit of the group size for cosine similarity classification. Finally, the group size for clustering can also be adjusted. This is closely related to the desired signal to noise enhancement. If the user desires an improvement of approximately N, the average number of elements in a subgroup should be N^2.

Results

The proposed algorithm is mainly developed to denoise TEM images having repeated structures. The noisy image, the results from all the methods and their Fast Fourier transforms (FFT)[4] can be seen in figure 6. The FFT converts data from the spatial domain into the frequency domain. *The white structures in the FFT represent information in the images???*. The signal corresponding to the low frequency components are represented at the center of the FFT and higher frequency components are present as we move away from the center. Noise corresponds to the high frequency components of the FFT and is present close to the edges.

The following can be concluded from the figure 6:

- The noisy image (of size 512*512 pixels) contains a lot of grainy structures which makes it hard to interpret the information. This is also reflected in the FFT, where the white structures are faintly visible at the center.
- N2V[8] is a self supervised, deep learning based image denoising method. N2V was trained with the patches of the noisy image. The training was done with $64 * 64$ patches, 100 epochs and a neighboring radius of 5.

  The results from N2V show a decently reconstructed circular structures and the noise in the black region of the image has been removed fairly well. However, the substructures have not been reconstructed well which can be seen in the zoomed in sections. The FFT shows the enhanced white structures at the center and the edges mostly look dark representing the suppression of noise.
- BM3D[4] is one of the widely used classical denoising methods. BM3D uses collaborative filtering in the transform domain for denoising images and it is a non blind denoising method, which means that the standard deviation of the

[3] https://github.com/mbanil/img-denoiser
[4] https://numpy.org/doc/stable/reference/generated/numpy.fft.fft2.html

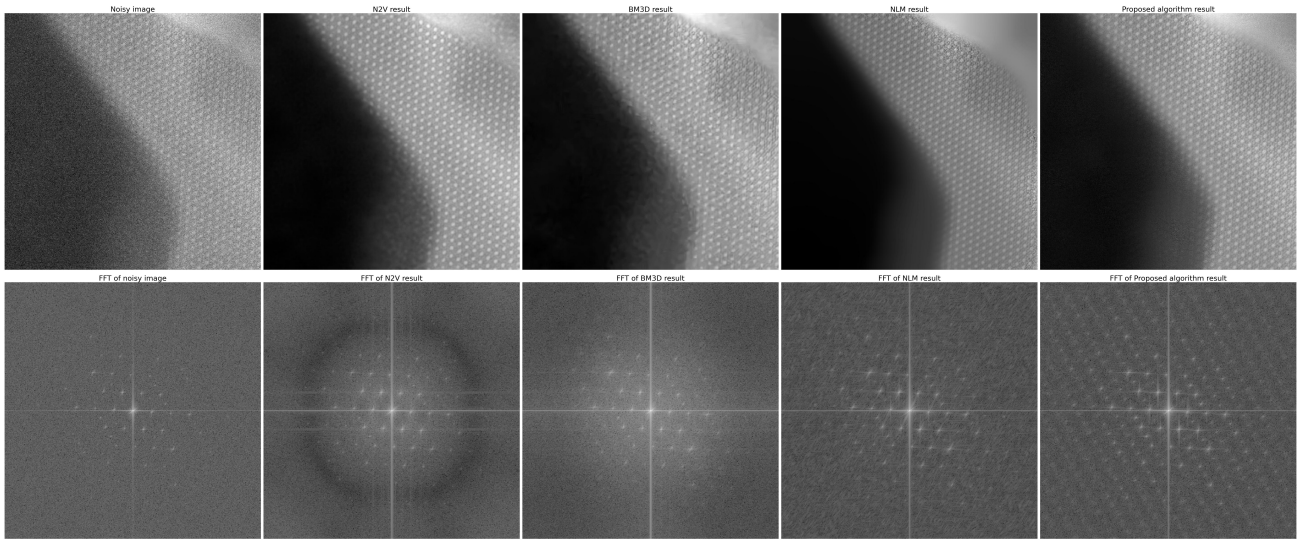

Figure 5. Comparison of results obtained from different methods

noise is required for denoising. The standard deviation was estimated with trial and error. The best results we obtained for a standard deviation of 0.06 for the normalized image.

The results from BM3D are similar to that obtained from N2V. The denoising effect is visible but the images are still not very useful for further analysis. This is also supported by the Fourier transform.

- Non Local Means (NLM)[3] is a conventional image denoising method that finds similar patches of images within a region and averages them to suppress noise. An NLM implementation[5] with a patch size of $46 * 46$, a search area of $100 * 100$ and a cut off distance of 0.36 was used.

  The result shows a good level of denoising. The circular structures and sub structures between them are visible fairly well. At most regions, the the level of noise suppression is good. But at some regions, the existence of noise can still be seen. The FFT also shows stronger white structures supporting the indicating the enhancement of image features. Overall, the results look good and more interpretable.

- Results with proposed denoising algorithm were obtained with a patch size of 46*46, a group size was between 5 and 100, and the clustering parameter, c equal to 2.7 were used.

  From the denoised result, it could be observed that the quality of denoising is marginally better than that from NLM. The image noise levels are suppressed fairly well and the information from the image can be well interpreted. The substructures between the circular structures are also better visible. From the FFT too, it can be seen that the white structures are more prominently visible. Some features can also be seen in the high frequency region which was previously not visible so well.

Apart from the qualitative improvement in the results, the proposed algorithm has a bigger advantage with regards to the computational time. The runtime of the proposed method for the images in figure 6 was 19.4 seconds, where as NLM, which was the closest to our results had a runtime of 171.4 seconds. This optimization in the runtime is particularly helpful when denoising image stacks from Transmission Electron Microscopy. The images in the stacks are **sometimes/often?** similar. In such cases, our algorithm not only matches the templates in the current image, but also the patches from other images in the stack. This significantly improves the quality of the results and with GPU computations, the computation speed is quite fast. For reference, the denoising of an image stack with 10 images of size 1024*1024 pixels took 281.8 seconds. The computations were carried out on a computer with 128 GB of RAM, Intel Xenon processor (16 CPUs), and Nvidia RTX6000 GPU.

Comparison with sample images

Obtaining a noise free image is often very difficult when it comes to microscopy. For quantitative comparison of results, we have generated a noise free image (ground truth) artificially using python. The generated image tries to imitate the microscopy

[5] https://scikit-image.org/docs/stable/auto_examples/filters/plot_nonlocal_means.html

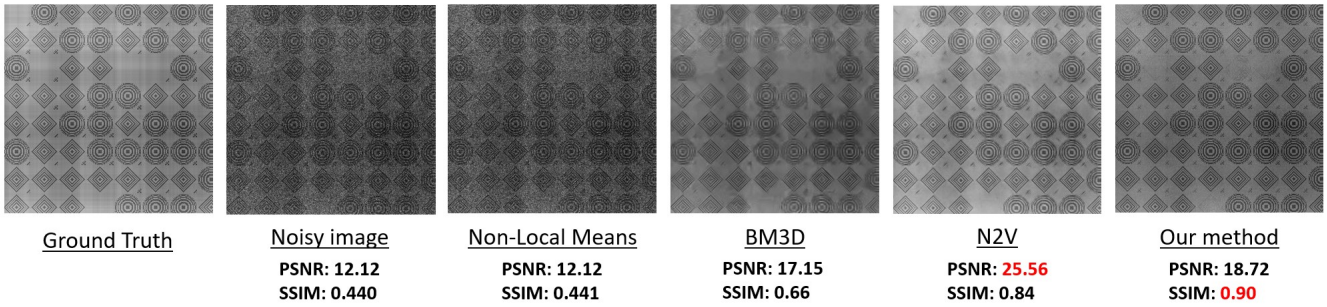

Figure 6. Comparison of results obtained from different methods

images which are best suited for applying our algorithm, i.e. images with similar patterns spread across the image. Noisy image is simulated by adding Poisson noise to the generated image. Different denoising methods have been applied on this noisy image and the Peak Signal to Noise Ratio (PSNR) and Structural Similarity Index Metric (SSIM) values of the denoised images have been found with respect to the ground truth. The ground truth, noisy image and the results from different methods can be seen in the figure 6.

The results from Non-Local Means show minimum improvement. This is because that this method requires similar image patches that are present close to each other, which is not always the case in the generated image. BM3D processes the image in the Fourier domain and hence suppressing the high frequency components. This results in the sharp features of the image being less prominent. N2V does a better denoising job which is reflected in it's PSNR value. However, on close inspection it can be observed that the high frequency components appear slightly blurred. This is where our method performs better. The pattern matching used to find similar pattern across the image ensures the correctness while preserving the sharpness of the image. This is also reflected in a higher SSIM value.

It should be noted that PSNR is based on mean squared error and is a distortion based evaluation metric. In image restoration there is always a trade-off between the distortion and the perceptual quality of the restored image. Often it is not possible to achieve both simultaneously[11]. In the figure 6, for our method even though we do not achieve the best PSNR value, we obtain the best results w.r.t SSIM which is a more perceptual metric[11].

Confidence map

From the above discussions, it can be observed that the algorithm can introduce artifacts. It is important to recognize this to determine if the denoised image is satisfactory. Since it is challenging to detect minute artifacts from Fourier analysis, a confidence map was developed. This confidence map is based on the variance within each cluster obtained after applying clustering. The variance should be small if the centroid is a good representation of its members. Also, a good centroid should not have any patterns in its variance. Patterns in the variance show that the centroid does not generalize its members well.

The confidence map of the denoised image is calculated by combining the variances[12] for different centroids as they are back plotted. The overall variance map, i.e., the confidence map of a denoised image is as shown in figure 7. This result has been produced for demonstration purpose with a very few initial *reference patches* selected very close to each other. The bright regions in the confidence map correspond to the denoised image artifacts. For instance, a bright region can be seen in the confidence map at the top-right position. An artifact can be found by inspecting the same region on the denoised image. Similarly, irregularities in the black region on the left side of the denoised image can be recognized by the brighter regions of the confidence map.

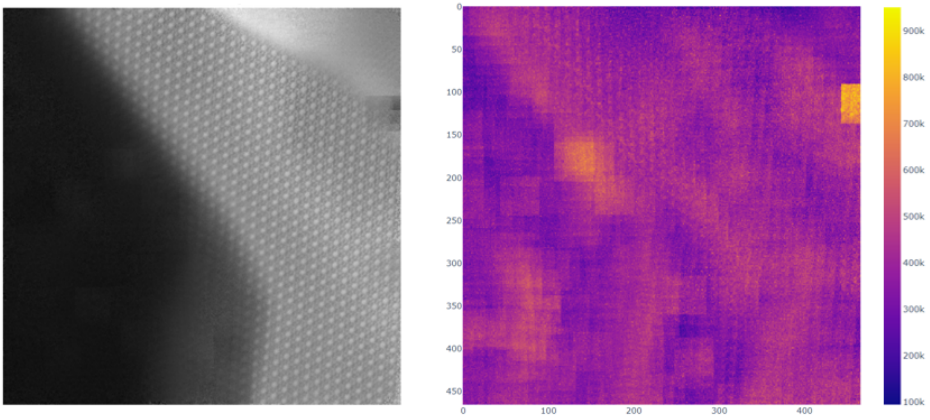

Figure 7. Comparison of results obtained from different methods

Discussion

The Discussion should be succinct and must not contain subheadings.

References

1. Curry, A., Appleton, H. & Dowsett, B. Application of transmission electron microscopy to the clinical study of viral and bacterial infections: Present and future. *Micron* **37**, 91–106, DOI: https://doi.org/10.1016/j.micron.2005.10.001 (2006).
2. Wang, Z. L. & Lee, J. L. Chapter 9 - electron microscopy techniques for imaging and analysis of nanoparticles. In Kohli, R. & Mittal, K. (eds.) *Developments in Surface Contamination and Cleaning (Second Edition)*, 395–443, DOI: https://doi.org/10.1016/B978-0-323-29960-2.00009-5 (William Andrew Publishing, Oxford, 2008), second edition edn.
3. Buades, A., Coll, B. & Morel, J.-M. Non-Local Means Denoising. *Image Processing On Line* **1**, 208–212 (2011).
4. Mäkinen, Y., Azzari, L. & Foi, A. Collaborative filtering of correlated noise: Exact transform-domain variance for improved shrinkage and patch matching. *IEEE Trans. Image Process.* **29**, 8339–8354, DOI: 10.1109/TIP.2020.3014721 (2020).
5. Zhang, K., Zuo, W. & Zhang, L. Ffdnet: Toward a fast and flexible solution for cnn-based image denoising. *IEEE Transactions on Image Process.* **27**, 4608–4622 (2018).
6. Zhang, K., Zuo, W., Chen, Y., Meng, D. & Zhang, L. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE transactions on image processing* **26**, 3142–3155 (2017).
7. Lehtinen, J. *et al.* Noise2noise: Learning image restoration without clean data. *CoRR* **abs/1803.04189** (2018). 1803.04189.
8. Krull, A., Buchholz, T.-O. & Jug, F. Noise2void-learning denoising from single noisy images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2129–2137 (2019).
9. Alake, R. Understanding Cosine Similarity And Its Application (2021).
10. Lewis, J. Fast normalized cross-correlation. *Ind. Light. Magic* **10** (2001).
11. Blau, Y. & Michaeli, T. The perception-distortion tradeoff. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6228–6237, DOI: 10.1109/CVPR.2018.00652 (2018).
12. Chan, T. F., Golub, G. H. & LeVeque, R. J. Updating formulae and a pairwise algorithm for computing sample variances. In *COMPSTAT 1982 5th Symposium held at Toulouse 1982*, 30–41 (Springer, 1982).

LaTeX formats citations and references automatically using the bibliography records in your .bib file, which you can edit via the project menu. Use the cite command for an inline citation, e.g.[?].

For data citations of datasets uploaded to e.g. *figshare*, please use the `howpublished` option in the bib entry to specify the platform and the link, as in the `Hao:gidmaps:2014` example in the sample bibliography file.

Acknowledgements (not compulsory)

Acknowledgements should be brief, and should not include thanks to anonymous referees and editors, or effusive comments. Grant or contribution numbers may be acknowledged.

Author contributions statement

Must include all authors, identified by initials, for example: A.A. conceived the experiment(s), A.A. and B.A. conducted the experiment(s), C.A. and D.A. analysed the results. All authors reviewed the manuscript.

Additional information

Algorithm

Algorithm 1 Denoising Algorithm

Input: Noisy image
Result: Denoised image
imgs ← loadImgs() ;
set [*templateSize, minGroupSize, maxGroupSize, clustParam*] ;
refPatches ← generate_initial_ref_patches(*imgs*, *templateSize*) ;
c ← 15 ;
for *c* ≥ 0 **or** (check if the number of *refPatches* in the last two iterations has changed) **do**
    tmpMatchResult ← [ ] ;
    for *img* in *imgs* **do**
        for *refPatch* in *refPatches* **do**
            tmpMatchResult.append(template_matching(*img*, *refPatch*)) ;
        end for
    end for
    maxValue ← max(*tmpMatchResult*, axis=2) ;
    refPatchIndex ← '*refPatches*' index at *maxValue* ;
    [*sorted_maxValue, positionX, positionY*] ← sortWithIdx(*maxValue*);
    t = *templateSize* ;
    list[idx,posX,posY] = [ ];
    for each *s,posX,posY* in *sorted_maxValue, positionX, positionY* **do**
        if *maxValue[posX,posY]* **not equal** 0 **then**
            maxValue[*posX: posX*+(*t*/4); *posY: posY*+(*t*/4)] ← 0 ;
            idx ← *refPatchIndex*[*posX*,*posY*] ;
            list.append([*idx*,*posX*,*posY*]) ;
        end if
    end for
    [*count,refPatchID*] ← count_patches_with_same_id(*list*);
    for *cnt,ID* in *count,refPatchID* **do**
        if *cnt* ≤ *minGroupSize* **then**
            list ← delete(*list*, *ID*);
        else if *cnt* ≥ *maxGroupSize* **then**
            list ← splitGroup(*list*, *ID*, *maxGroupSize*);
        end if
    end for
    refPatches ← average_patches_with_same_id(*list*) ;
    c ← *c*-1 ;
end for
subGroup[*centroid, posX, posY*] = [ ];
for *refPatchID* in *refPatches* **do**
    patches_with_sameRef ← get_patches_with_same_id(*list*,*refPatchID*);
    numClusters ← count(*patches_with_sameRef*)/*clustParam* ;
    subGroup.append(cluster(*patches_with_sameRef, numClusters*)) ;
end for
gaussianWeight ← createGaussianWeight();
denoisedImg ← backPlot(*subGroup*, *gaussianWeight*);