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.

Proposed method

The proposed denoising algorithm identifies similar patches within the entire image and averages them to suppress the noise. Averaging patches with similar information suppresses 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,

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

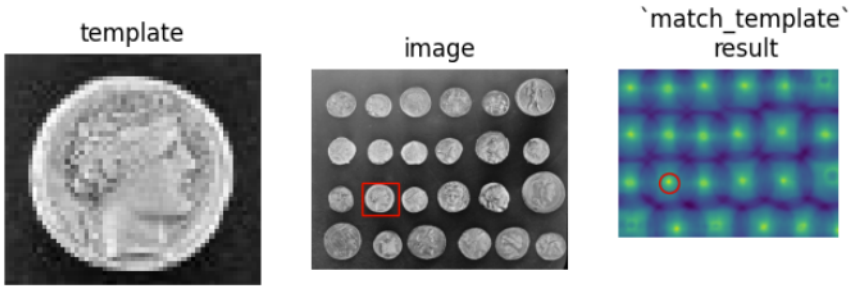

Figure 1. Template matching example [1]

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 2 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,

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

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 1. An image, a template taken from the image, and the corresponding result can be seen in figure 1. 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.

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

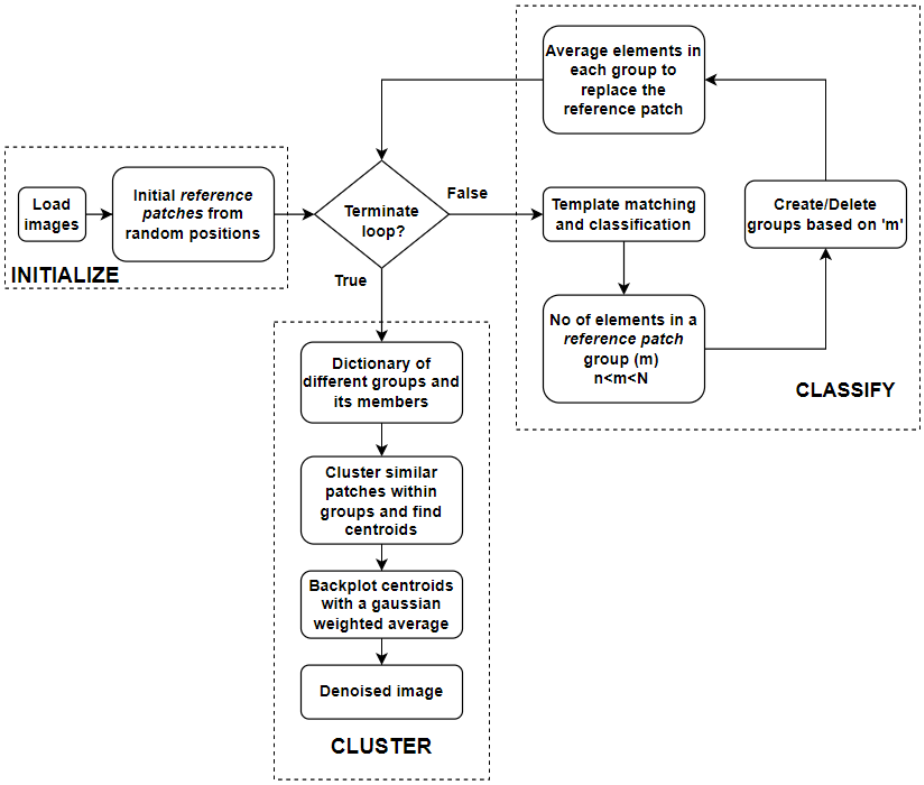

Figure 2. Flowchart of the algorithm

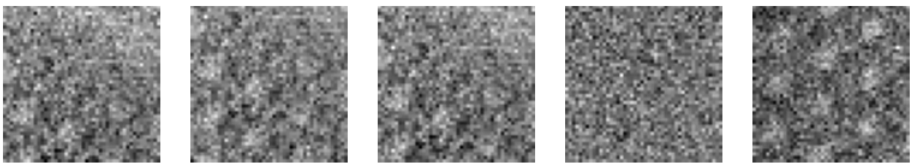

Figure 3. Initial *reference patches* (of size 46*46 pixels), taken from different regions of the input image

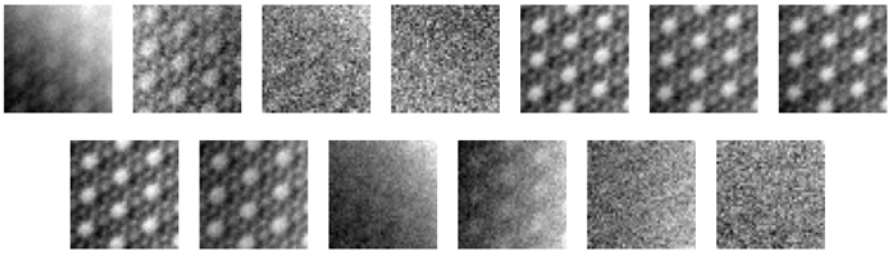

Figure 4. Final *reference patches*

In the 'classify' section shown in figure 2, the patches at every position in the image are classified into different groups based on the cosine similarity values, and these groups are represented by a *reference patch*. If x and y represent a pixel location, then the patch at (x,y) is obtained from pixel locations (x to $x+m$, y to $y+m$), where $m*m$ is the patch size. When the cosine similarity is applied, each of such $m*m$ patches in the image is compared with every $m*m$ *reference patch*. For an image of size $p*q$ and a *reference patch* of size $m*m$, the cosine similarity result has a size $(p-m+1)*(q-m+1)$. The result has values between 0 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 n *reference patches*. The result for each *reference patch* is of shape $(p-m+1)*(q-m+1)$. All results are stacked into an $n*(p-m+1)*(q-m+1)$ array. From this array, the *reference patch* that matches the best at every location is identified. Hence, the $n*(p-m+1)*(q-m+1)$ array is reduced to a $(p-m+1)*(q-m+1)$ array which contains the best fitting reference patch index at every position. 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 more 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 shoown) 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, the average might differ from the actual patch by a large extent.
- Averaging a large range of data also changes the range of pixel values, altering the brightness after reconstruction.

These problems can be solved by averaging over a small group with very closely matched patches. Clustering is applied within every group (represented by the final *reference patch*) to create smaller subgroups, i.e., clusters are created within large groups. The number of clusters in a group, is obtained by dividing the number of patches in a group by 'c^2', where c is user defined. In other words, the signal-to-noise ratio can be adjusted by changing the 'c' value. The centroids found from clustering now represent a large group that was before represented by a single *reference patch*. Centroids used for back plotting are multiplied by a $2D$ Gaussian weight. A weighted average is performed when more than one centroid is used at a position. Gaussian weight smoothens the edges of the centroids, thus preventing artifacts in the reconstructed image.

The denoising algorithm can be seen in the section 1. The implementation of the denoising algorithm can be found on github[3].

Parameters of the algorithm and stability

The following are the algorithm parameters.

- **Size of the patches**: Patches represent features in an image. Patch sizes should neither be too small nor too large. A patch size of $46*46$ performed the template matching task reasonably well, and hence this was the size used for most of the experiments.

[3] https://github.com/mbanil/img-denoiser

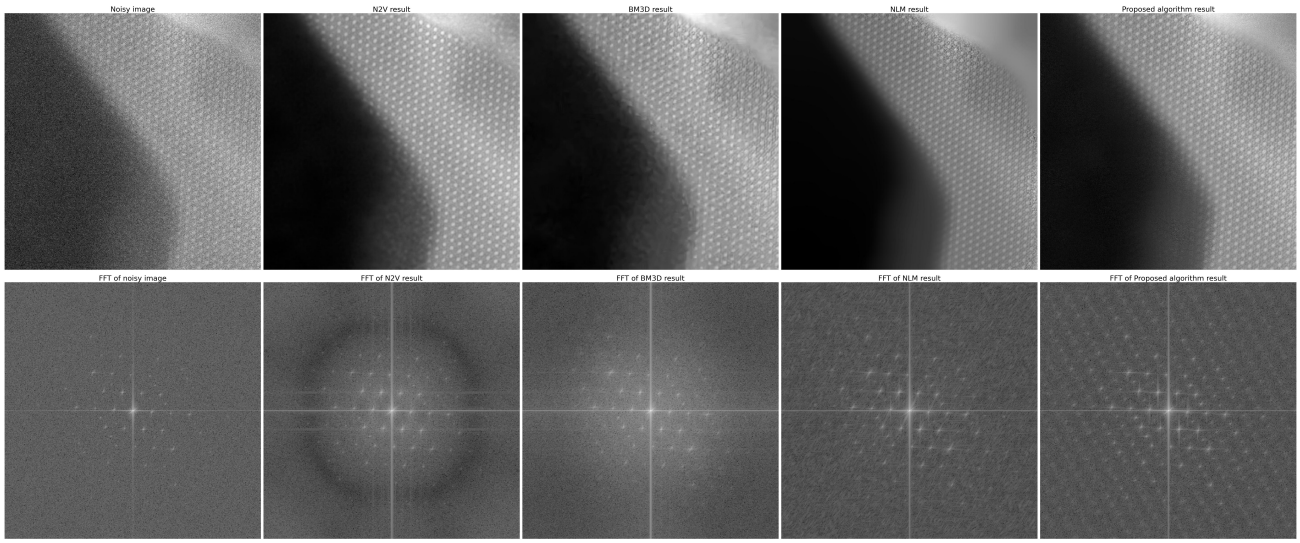

Figure 5. Comparison of results obtained from different methods

- **Position of the initial patches**: The position of the initial *reference patches* has to be selected such that the patches represent different regions in the image. These dissimilar patches are selected by randomly initializing the templates such that no two templates are close to each other.
- **Minimum number of patches in a group**: When there are very few patches, averaging them will not significantly suppress the noise. Hence groups with members lesser than this value are deleted.
- **Maximum number of patches in a group**: The maximum number of patches in a group (represented by a *reference patch*) is set to 100. The first reason is to avoid over-generalization. If the group size is big, a single *reference patch* will represent too many patches, which is undesirable.

  More groups are generated if this parameter has a lower value, resulting in more *reference patches*. The number of *reference patches* is proportional to the number of times the cosine similarity is applied. Too many *reference patches* can affect the overall computation speed of the algorithm. Hence a trade-off is required.
- **The number of clusters within a group**: This parameter must be set based on how much self-similarity is present in the input image. If there is much self-similarity, few clusters can be made because more patches are very close to each other. Hence a centroid can be used to represent more patches. If there is not much self-similarity, more clusters are required to group more similar patches. Hence, this parameter depends on the level of self similarity in the input image.

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 5. 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. A

The following can be concluded from the figure 5:

- 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.

[4] https://numpy.org/doc/stable/reference/generated/numpy.fft.fft2.html

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 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 5 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.

Confidence Map

Comparison with example images

Noise Suppression

The main reason for noise suppression is the averaging of very similar patches in the image. Suppression of noise is proportional to $\sqrt{n}$, where n is the number of patches used[3].

Figure **??** shows a visualization where each marker represents a patch and same colored markers represent similar patches. It can be observed that each group has many similar patches. As explained in the above paragraph, averaging similar patches will suppress the noise present. Since, sometimes the number of patches in a group can be too large, which leads to problems shown in figure **??**, clustering is done to make smaller subgroups.

Consider a group of patches that are represented by the same colored squares in figure **??**. The group members become the samples for clustering, and the number of clusters depends on the parameter 'c' that the user can define. There are around 30 groups shown, and hence clustering is done around 30 times. As explained earlier, hierarchical clustering is the clustering method used. The parameter 'c' should be chosen based on the extent of self-similarity in the input image. For this particular image shown in figure **??**, a value of 2.7 was used. Upon experimenting, the ideal range for this parameter was between 1.5 (for images with less similar components) and 2.7 (for images with more similar components). The number of clusters in a group is found by,

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

$$w = \left\lceil \frac{n}{c^2} \right\rceil \tag{6}$$

where w is the number of clusters, n is the number of group members, and $\lceil \rceil$ represents the conversion to the next highest integer.

Sometimes clustering can also fail to create smaller clusters when the clustering parameter is not chosen correctly. In figure **??**, there are seven members, and choosing a clustering parameter of 2.7 will result in one cluster, which is not desirable as one of the members is dissimilar from the rest of the group. Clustering parameter having a higher value has an advantage as the noise suppression is more. However, it can introduce some artifacts if there is not much self-similarity in the image.

Once the different centroids are obtained, they are back plotted to get the denoised image. Centroids overlap while back plotting and are averaged with a $2D$ Gaussian weight. Averaging here further suppresses the noise. The number of patches that are averaged at every pixel location is visualized in figure **??**. The visualization also considers the number of patches that form a centroid in the clustering process. One can observe that over 200 patches have been averaged at certain regions, which suppresses the noise by over 90% (since suppression of noise is proportional to $\sqrt{n}$, where n is the number of patches used[3].).

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 hierarchical clustering. The variance should be small if the centroid is a good representation of its member patches. 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. Figure **??** shows the centroids which is a good generalization (centroid-1) and which is not (centroid-2). The centroid-2 has been taken for demonstration purposes by having very few clusters in a group.

The confidence map of the denoised image is calculated by combining the variances for different centroids as they are back plotted. The algorithm[11] for combining the variances has been mentioned in appendix under algorithm 2.

The overall variance map, i.e., the confidence map of a denoised image is as shown in figure **??**. 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.

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. 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*) ;
$c \leftarrow 15$;
for $c \geq 0$ **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 $cnt \leq minGroupSize$ **then**
            list ← delete(*list*, *ID*);
        else if $cnt \geq maxGroupSize$ **then**
            list ← splitGroup(*list*, *ID*, *maxGroupSize*);
        end if
    end for
    refPatches ← average_patches_with_same_id(*list*) ;
    $c \leftarrow c$-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*);

Algorithm 2 Algorithm for calculating variance

1: $weight \leftarrow weightA + weightB$;
2: $delta \leftarrow avgB - avgA$;
3: $M2 \leftarrow (M2\_A + M2\_B + delta^2 * weightA * weightB)$ / n ;
4: $Var\_AB = M2 / (n - 1)$;