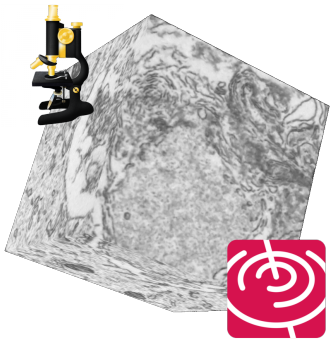
DenoisEM

User manual

Joris Roels
Frank Vernaillen
Anna Kremer
Amanda Goncalves
Jan Aelterman
Hiêp Q. Luong
Bart Goossens
Wilfried Philips
Saskia Lippens
Yvan Saeys



DENOISEM

DenoisEM

Joris Roels
Yvan Saeys
Ghent University
Flanders Institute for Biotechnology

Frank Vernaillen
Anna Kremer
Amanda Goncalves
Saskia Lippens
Flanders Institute for Biotechnology

Jan Aelterman
Hiêp Q. Luong
Bart Goossens
Wilfried Philips
Ghent University

Abstract

The recent advent of 3D in Electron Microscopy (EM) has allowed for detection of detailed sub-cellular nanometer resolution structures. While being a scientific breakthrough, this has also caused an explosion in dataset size, necessitating the development of automated workflows. Moreover, large 3D EM datasets typically require hours to days to be acquired. Accelerated imaging is possible through faster dwell-times, but comes at the cost of noise. Many denoising solutions exist, however, advanced techniques that better preserve ultrastructural detail, tend to be less accessible to the biomedical community due to low-level programming environments, complex parameter tuning or a computational bottleneck. To alleviate these issues, we present DenoisEM: an interactive and GPU accelerated denoising tool, accessible through ImageJ. DenoisEM comes with both classical and state-of-the-art denoising algorithms and works in an intuitive and user-friendly fashion. Fast parameter tuning and processing of large 3D datasets is enabled through large-scale parallel computing. Experimental results show that DenoisEM is one order of magnitude faster than existing denoising frameworks and can accelerate data acquisition by a factor of 4 without significantly affecting the data quality. Lastly, we show that image denoising benefits visualization and (semi-)automated segmentation and analysis of ultrastructure in various volume EM datasets.

Key words

Electron microscopy, denoising, GPU acceleration.

Contributing

DenoisEM is an open source project that was initiated by Ghent University and the Flanders Institute for Biotechnology. We stimulate the community to contribute to our project or report issues through our GitHub repository.

Get in touch

Feel free to contact us with any questions through the contact form of our project page.

Table of Contents

1. Installation

1.1 Prerequisites

DenoisEM requires the following hardware and software:

- OS: 64-bit Windows 7 or higher
- GPU setup:
  - We recommand an NVIDIA card with compute capability 3.0 or higher. For NVIDIA cards, we also recommend the corresponding driver and CUDA toolkit.
  - Alternative GPUs such as AMD, Intel HD graphics and Intel Xeon Phi are supported. In this case, we recommend the OpenCL backend.
- CPU only setup:
  - In this case, we recommend the OpenCL backend.
- ImageJ (we recommend Fiji, which is an extension that comes with several other useful plugins)
- GtkSharp 2.12.12 or higher
- Microsoft Visual C++ 2015 or higher redistributables
- .Net 4.5 or higher (should be included in your Windows installation)

1.2 Installation Instructions

1. Locate the root and plugins folder of your ImageJ/Fiji installation.
2. Save the Java Quasar bridge and DenoisEM jar in the plugins folder.
3. Download the Quasar runtime and extract it to the ImageJ/Fiji root directory.

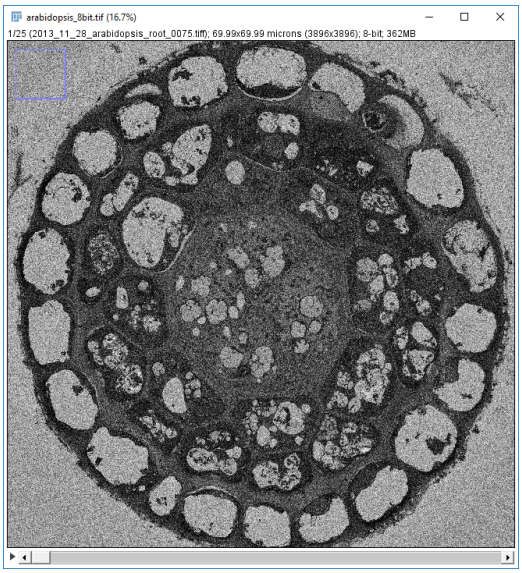
2. Usage

2.1 Getting Started

In this section, we describe how to denoise a 3D dataset in an interactive setup using DenoisEM.

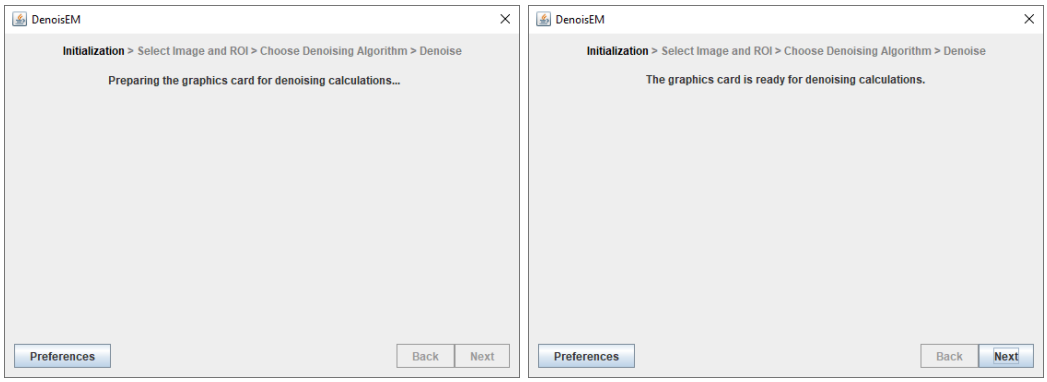
1. **Load the data in ImageJ**

   Load the image or image stack in ImageJ. (sample)



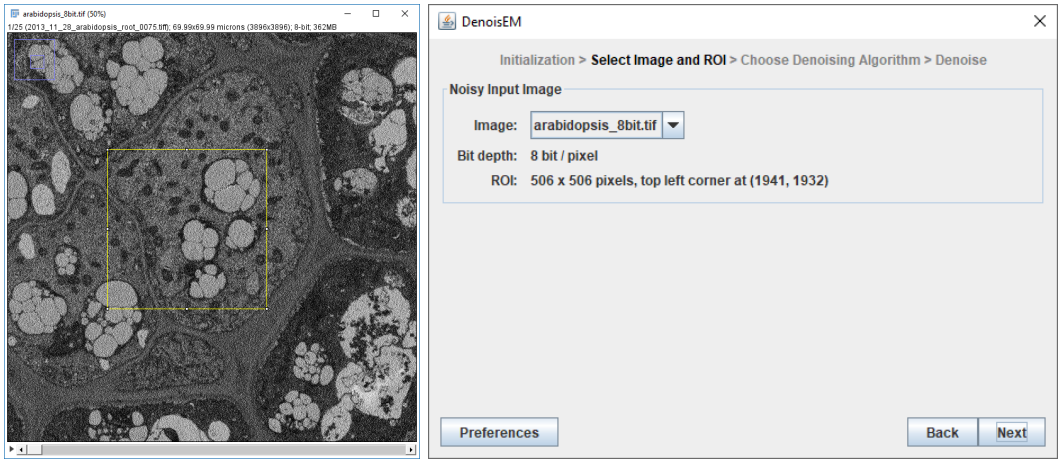
2. **Start the DenoisEM plugin**

   Start the DenoisEM plugin from the Plugins menu in ImageJ (Plugins $>$ DenoisEM $>$ Denoise), which brings up the main window. DenoisEM will automatically detect a GPU device and initialize the Quasar runtime backend.



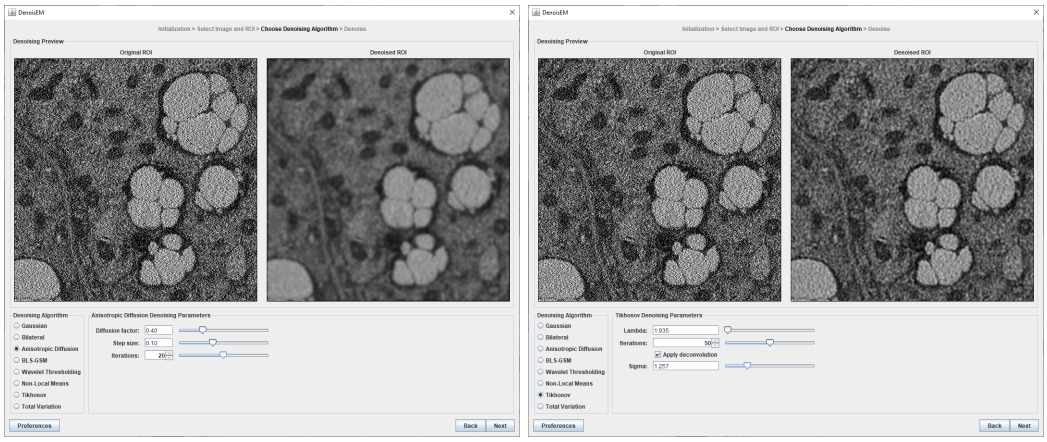
3. **Select region of interest**

   Select a region of interest that is sufficiently representable for the complete dataset using ImageJ's rectangular selection tool. Make sure the correct image is selected in the DenoisEM window and press 'Next'.



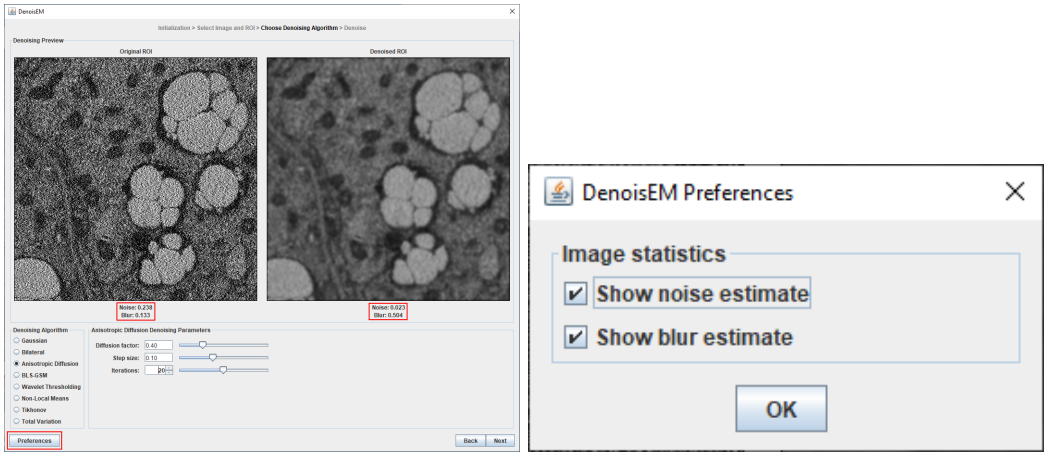
4. **Algorithm selection and parameter finetuning**

   Initially, DenoisEM will show the result of a Gaussian filter, but seven other restoration algorithms are available. They can be selected from the 'Denoising Algorithm' menu. The parameters of each algorithm can be tuned interactively by dragging the corresponding sliders or by entering the parameter value numerically. Once the desired restoration quality is obtained, press 'Next' to proceed to denoising the full 3D stack.



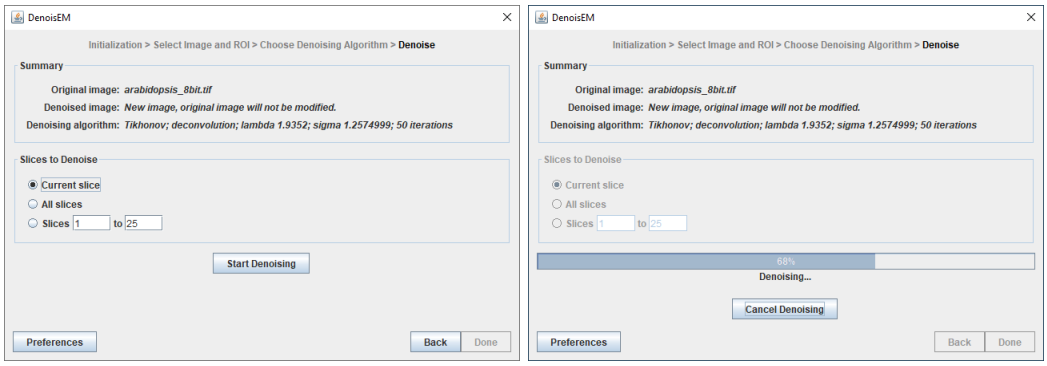
   Optionally, a real-time estimate of the noise and blur levels in the original and denoised images can be displayed below the preview images. This setting is available in the Preferences window. The noise level estimates the noise

standard deviation (based on median absolute deviation) after normalizing the pixel values to the range 0–1. The blur level is a scalar value that varies between 0 (sharp) to 1 (blurry) [1].



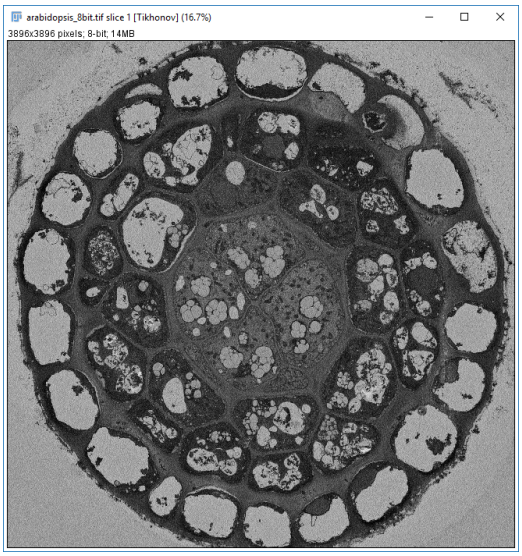
5. **Process complete dataset**

   Select the ‘All slices’ option and press ‘Start denoising’ to process the full 3D dataset. It is also possible to specify a subset of slices for denoising, or denoise only the current slice.



6. **Saving the final result**

   The denoised dataset will open in a separate window and can be saved using ImageJ.


arabidopsis_8bit.tif slice 1 [Tikhonov] (16.7%)
3896x3896 pixels; 8-bit; 14MB

2.2 Algorithm selection and parameter tuning

The plugin has several image restoration algorithms available, both classical and more recent techniques. What follows is a listing of the implemented techniques and a discussion on parameter influence and general performance. A concise overview of this is shown in table 1 and table 2.

Gaussian filter

The Gaussian filter is a special case of linear filters that is commonly used to reduce noise in images. Its only parameter is the standard deviation σ of the Gaussian filter mask. As $\sigma \to 0$, the mask converges to a Dirac pulse and the resulting filtered image $\hat{\mathbf{x}}$ will converge to the source image $\mathbf{y}$ as no local pixels are averaged. Consequently, there is no significant noise attenuation. However, as σ grows larger, the filter mask will resemble an local averaging operation and therefore significantly reduce noise.

Linear filters, and Gaussian filters in particular, owe their success (*e.g.* in [2]) to their computational efficiency: with a recent GPU (NVIDIA GTX 1070) our plugin is able to process 16MP images under 10 ms. The Gaussian filter also effectively suppresses noise in 'flat' regions (*i.e.* regions with a relatively constant intensity). However, high contrast edges will typically become blurred due to the filter kernel that is chosen independently of the image content (especially for large σ values).

Bilateral filter

Similar to the Gaussian filter, the bilateral filter [3] averages pixels locally to obtain the noise-free pixel value. However, the main difference is that it also takes into account pixel similarity. This is implemented using two functions f_{sp} and f_{int} which respectively model spatial and intensity similarity. In practice, they are implemented as Gaussians with respective damping parameters σ_{sp} and σ_{int}. Whenever σ_{sp} is chosen larger relative to σ_{int}, spatial-based filtering will be prioritized over intensity-based filtering and vice versa. Note that the bilateral filter converges to a Gaussian filter with standard deviation $\sigma = \sigma_{\text{sp}}$ as $\sigma_{\text{int}} \to \infty$.

In practice, the bilateral filter is a slight improvement over Gaussian filtering, but it still suffers from the same undesirable trade-off between edge blurring and noise suppression. The reason for this is that intensity-based pixel comparison is not always able to distinguish edges or texture from noise. However, by properly finetuning the damping parameters bilateral filtering should be able to outperform Gaussian filtering.

Anisotropic diffusion

Anisotropic diffusion [4] is a non-linear filter that attempts to filter along the edge direction to avoid blurring within the context of non-linear diffusion. Therefore, its

main parameter is the diffusion parameter κ. Small values will allow little diffusion and therefore little noise suppression, whereas larger values will improve denoising. The diffusion problem is an optimization problem and therefore also requires a step size η and number of optimization iterations N. Large values of N will approximate the diffusion process more accurately. A small step size will have the same effect, but may require more iterations.

The crucial parameter of anisotropic diffusion is its diffusion parameter. Larger values will increase noise suppression at the risk of directional (cartoon-like) edge blurring. Additionally, the diffusion equation is not always a perfect image prior. Therefore, the solution at convergence might not be the most accurate restoration. In some cases, it is beneficial to break off the optimization at a certain (smaller) amount of iterations.

Wavelet thresholding

Wavelet thresholding [5, 6] isolates and attenuates noise in an alternative (multiresolution) domain. Shrinkage of the multiresolution coefficients requires only a thresholding value T. Small values of T will leave the input image relatively unchanged. For example, when $T = 0$, the soft-thresholding operation is simply the identity operation and the resulting image will be identical to the input. As larger values are chosen for the threshold, more noise coefficients are shrunk and noise is therefore reduced more significantly.

The issue with wavelet shrinkage is that high-frequency edges with a low magnitude are also suppressed, especially when T is chosen too large. This leads to incorrect frequency information in the restored images (which typically manifests as ringing effects). For this reason, one might prefer smaller values of T, which will also result in less noise suppression.

Tikhonov restoration

Tikhonov restoration [7] exploits the fact that the total edge magnitude of an image should be low. This prior is mathematically formulated as a minimization problem that consists of two terms: a data fit and gradient magnitude penalty. These terms are weighed w.r.t. each other using a regularization parameter λ. Large values will prioritize gradient penalty over data fit and vice versa. The optimization requires a number of iterations N which is preferably large for accurate optimization. The Tikhonov prior can be used as a deconvolution method if the PSF (point spread function) $\mathbf{H}$ is incorporated in the data fit. To do so, the standard deviation σ of the Gaussian kernel should be accurately tuned so that the resulting kernel matches the system PSF. Note that in this case, the parameter λ also offers a trade-off between sharpening and noise suppression.

Tikhonov restoration uses a gradient penalizing prior which in practice will not always hold. For this reason, edges might be blurred when the parameter λ is

not properly tuned. This is especially important if deconvolution is included in the pipeline.

Total variation restoration

Total variation restoration uses the prior information that images consist of piecewise flat areas delineated by crisp edges. Similar to Tikhonov restoration, this comes down to an optimization problem that uses a regularization parameter λ to weigh the prior in contrast to the data fidelity term. Similarly, the optimization requires a number of iterations N which is preferably large for accurate optimization. Total variation can be used as a deconvolution method. This is not yet supported in the current version, but listed as future work.

Total variation restoration minimizes the total variation within the image. It is therefore expected that the restored images have cartoon-like artifacts when the regularization parameter is chosen too high.

BLS-GSM

The BLS-GSM method [8] decomposes the image into J scales and K oriented pyramid subbands, denoises the highpass subbands based on the noise level σ and inverts the pyramid transform. In practice, the noise level is not given and we use the MAD estimator as a first guess. Nevertheless, the user can adjust this if necessary. Whenever the noise level is underestimated, noise will be insufficiently suppressed and vice versa for overestimation. Note that noise is attenuated in the wavelet domain and requires a number of wavelet scales J and orientations K. High values will increase image restoration quality, but may also increase redundancy in the transformed coefficients, complicating the estimation of the restored coefficients.

BLS-GSM is the state-of-the-art in multiresolution-based image denoising due to its accurate noise modeling capabilities. Proper estimation of the noise level σ is therefore crucial. However, this comes at a significant computational cost: we do not achieve real-time performance for this method, but significant speedups compared to existing implementations.

Non-local means denoising

Non-local means [9] finds similar pixels by comparing local neighborhoods of both local and non-local pixels. The restored pixel value is obtained by weighted averaging where similar pixels are assigned larger weights. The decay of this weighting is controlled by a damping parameter h. In practice, the search neighborhood is defined by a window of size $W \times W$ where patches of $B \times B$ are compared to find similar pixels. Both the parameters W and W should preferably be large, but significantly influence computation time. For example, a large search window is

beneficial as it allows to find similar structures that may be further away from the reference pixel. Obviously, larger values of W would allow the algorithm to find more similarity within the image and therefore attenuate noise more effectively. In previous work [10], we found that noise in scanning electron microscopy (SEM) is highly correlated. We proposed a set of alternative (decorrelated) NLM weights $w'_{i,j}$ and improve the restoration. This option is also available in our plugin through a flag φ and recommended when the data originates from SEM devices. Similarly to Tikhonov restoration, we have also provided a state-of-the-art deconvolution algorithm based on the non-local means algorithm. This method relies on a regularization parameter λ that offers control over the classical trade-off between noise suppression and sharpening. Similar to Tikhonov deconvolution, an accurate estimation of the PSF should be provided through the Gaussian kernel standard deviation σ. Note that this comes at a significant computational cost.

A common issue with non-local means is that it fails to restore non-repetitive texture regions due to the lack of self-similarity. In this case, these noisy regions will typically remain in the image, as opposed to other methods that would introduce blur. More optimal results may be obtained by increasing the search window size. In the case of deconvolution, we recommend to first optimize the damping parameter h so that most of the noise is attenuated, before optimizing the regularization parameter. The reason for this is that the regularization parameter may introduce noise amplification if h is too low.

Algorithm	Parameters	Parameter Meaning	Parameter Effect
Gaussian filter	σ	Standard deviation of Gaussian kernel	Trade off between noise suppression and potential edge blurring
Wavelet thresholding [5, 6]	T	Threshold	Trade off between noise suppression and ringing artefacts
Anisotropic diffusion [4]	κ	Diffusion parameter	Trade off between noise suppression and potential edge blurring
	η	Step size	Small values approximate PDE more accurately
	N	Number of iterations	Large values approximate PDE more accurately
Bilateral filter [3]	σ_{int}	Intensity damping parameter	Prioritise intensity-based over spatial-based filtering, when chosen large
	σ_{sp}	Spatial damping parameter	Prioritize spatial-based over intensity-based filtering, when chosen large
Tikhonov denoising [7]	λ	Regularization parameter	Large values enforce higher noise suppression
	N	Number of iterations	Large values yield better optimization
Tikhonov deconvolution [7]	σ	Standard deviation Gaussian blur kernel	Should match the system PSF

Table 1. Algorithms implemented in the plugin and their corresponding parameters exposed to the user and their effect.

Algorithm	Parameters	Parameter Meaning	Parameter Effect
Total variation denoising [11]	λ	Regularization parameter	Large values enforce higher noise suppression
	N	Number of iterations	Large values yield better optimization
BLS-GSM [8]	σ	Noise standard deviation	Either insufficient noise suppression or edge blurring if estimated inaccurately
	J	Number of wavelet scales	Frequency scales are more accurately modelled at a redundancy cost
Non-local means denoising [9]	h	Similarity damping parameter	Provides control over the degree of averaging similar pixels
	B	Similarity window size	Larger window sizes are able to find more complex similar structures
	W	Search window size	Larger search windows allows more similar pixels to be found and improve denoising
	φ	Decorrelate	Use decorrelated weights from [10], specifically designed for SEM imaging
Non-local means deconvolution [12]	λ	Regularization parameter	Trade off between noise suppression and sharpening
	σ	Standard deviation Gaussian blur kernel	Should match the system PSF

Table 2. Algorithms implemented in the plugin and their corresponding parameters exposed to the user and their effect.

2.3 Re-using parameter settings

DenoisEM saves the used denoising algorithm and parameter settings as metadata when you save the image as a TIFF file. If you open the denoised image in ImageJ, you can show this metadata by selecting 'Properties' in the 'Image' menu.

3. FAQ

- **Which types of images does DenoisEM support?**

  DenoisEM supports 8-bit and 16-bit single channel images and image stacks. Extending DenoisEM to support multi-channel images is straightforward and planned as future work.

- **Can DenoisEM handle images with a large width or height in pixels?**

  Yes. DenoisEM will split large images into a set of overlapping tiles. Each tile will be denoised independently to avoid running out of GPU memory. Afterwards the tile overlaps will be discarded and the center portions of the tiles recombined into a seamless result image. This approach completely avoids tile edge artefacts.

- **Why does ImageJ refuse to perform certain operations and says my image is locked when I am using DenoisEM?**

  During denoising preview and during full denoising, DenoisEM locks the input image to avoid that it changes or disappears behind its back. If you want to modify the image, simply close the DenoisEM wizard to unlock the image.

- **Can I change the region of interest (ROI) during algorithm selection and parameter optimization?**

  Yes. In the DenoisEM wizard, click Back to return to the 'Select Image and ROI' step in the DenoisEM wizard. Then change the ROI either by dragging the existing ROI around, or by drawing a new ROI rectangle. Then click Next to return to the algorithm selection and parameter optimization panel. The denoising algorithm and the user chosen parameter values are not modified when changing the ROI. Similarly, the z-slice on which the denoising preview operates can only be changed while in the ROI selection panel.

- **How can I tell later on which parameters were used to denoise an image using DenoisEM?**

  DenoisEM stores the denoising algorithm and its parameters as image properties in ImageJ. These key-value pairs can be inspected in ImageJ via 'Image $>$ Properties...' These properties are also saved (and loaded) along with the pixel data to file formats such as TIFF.

- **Can I perform 'in place' denoising directly on the source image?**

  No. Currently, DenoisEM always creates a new image or image stack with the denoised result.

- **Why is there sometimes temporarily a red frame around the denoising preview? What does it mean?**

  The red frame indicates that the denoising calculations cannot be performed at interactive speed anymore. This happens for certain algorithm-parameter combinations that are exceptionally computationally expensive. As long as the red frame is visible, user changes to the algorithm parameters are ignored until DenoisEM completed the current denoising calculations, at which point the red frame disappears. This avoids a buildup of denoising work.

- **Can I apply the denoising algorithms with parameters values that are outside the range of the parameter sliders in the DenoisEM user interface?**

  Yes. While we tried to offer reasonable parameter ranges based on an estimate of the amount of noise present in the image, the user may wish to experiment with more extreme parameter settings. In that case, simply fill in the desired parameter value in the corresponding numeric edit field instead.

- **What about Linux and Mac support?**

  The Quasar backend, which is responsible for executing the denoising algorithms on the GPU, is already available for Linux. Mac support for Quasar would be possible too (via OpenCL) but is currently not planned. However, to simplify installation and testing we decided to release the first version of DenoisEM on Windows only.

- **Does DenoisEM work in a virtualized environment, such as VirtualBox?**

  We do not support DenoisEM running in a virtualized environment. The virtualization layer interferes with Quasar's need to directly access the graphics hardware. Limited testing indicates that DenoisEM will likely work, but will fall back to slower CPU parallelism instead of faster computation on the GPU.

References

[1] Crete F, Dolmiere T, Ladret P, Nicolas M (2007) The blur effect: perception and estimation with a new no-reference perceptual blur metric. *Human Vision and Electronic Imaging XII*, , . https://doi.org/10.1117/12.702790

[2] Kushwaha HS, Tanwar S, Rathore KS, Srivastava S (2011) De-noising filters for TEM (Transmission Electron Microscopy) image of nanomaterials. *Conference on Advanced Computing and Communication Technologies*, , pp 276–281. https://doi.org/10.1109/ACCT.2012.41

[3] Tomasi C, Manduchi R (1998) Bilateral filtering for gray and color images. *IEEE Conference on Computer Vision and Pattern Recognition* :839–846https://doi.org/10.1109/ICCV.1998.710815. Available at http://ieeexplore.ieee.org/document/710815/

[4] Perona P, Malik J (1990) Scale-Space and Edge Detection Using Anisotropic Diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12(7):629–639. https://doi.org/10.1109/34.56205

[5] Donoho DL, Johnstone JM (1994) Ideal spatial adaptation by wavelet shrinkage. *Biometrika* 81(3):425–455. https://doi.org/10.1093/biomet/81.3.425

[6] Selesnick IW (2002) The design of approximate Hilbert transform pairs of wavelet bases. *IEEE Transactions on Signal Processing* 50(5):1144–1152. https://doi.org/10.1109/78.995070

[7] Tikhonov ANAN, Leonov AS, Yagola AG (1998) *Nonlinear ill-posed problems* (Springer Netherlands, Dordrecht, Netherlands), .

[8] Portilla J, Strela V, Wainwright MJ, Simoncelli EP (2003) Image denoising using scale mixtures of Gaussians in the wavelet domain. *IEEE Transactions on Image Processing* 12(11):1338–1351. https://doi.org/10.1109/TIP.2003.818640

[9] Buades A, Coll B, Morel JMJM (2005) A non-local algorithm for image denoising. *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, Vol. 2, pp 60–65. https://doi.org/10.1109/CVPR.2005.38

[10] Roels J, Aelterman J, De Vylder J, Luong H, Saeys Y, Lippens S, Philips W (2014) Noise Analysis and Removal in {3D} Electron Microscopy. *Advances in Visual Computing* (Springer International Publishing), *Lecture Notes in Computer Science*, Vol. 8887, pp 31–40. https://doi.org/10.1007/978-3-319-14249-4{_}4. Available at http://dx.doi.org/10.1007/978-3-319-14249-4_4

[11] Rudin LI, Osher S, Fatemi E (1992) Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena* 60(1-4):259–268. https://doi.org/10.1016/0167-2789(92)90242-F

[12] Roels J, Aelterman J, De Vylder J, Luong H, Saeys Y, Philips W (2016) Bayesian deconvolution of scanning electron microscopy images using point-spread function estimation and non-local regularization. *International Confer-*

ence of the IEEE Engineering in Medicine and Biology Society, Vol. 2016-Octob, pp 443–447. https://doi.org/10.1109/EMBC.2016.7590735