Introduction

We want to explore different smoothing algorithms applied on brain MRI data (*sub-11_T1w.nii.gz*).

Main Work

There are several parts in the code which consists in:

1) Upload of the NIFTI image **sub-11_T1w.nii.gz** from the folder data. Make sure that the folder exists with the file. If not, please modify the line: *vol = niftiread('data/sub-11_T1w.nii.gz')* to point to the location with the file.
   The image is flipped following the instructions given for the function to generate noisy images to facilitate image manipulation. Slices 102 and 119 are plotted to verify if the data is in the expected format.
2) Before starting any filtering, we run a grid search, looking for the optimal parameters of each filter with the lowest root mean squared error. Unless needed do not uncomment this part as it is resource intensive and takes a while.
   We have these results:

Filter	Parameter	Range	Optimal Parameter	Optimal RMSE
Gaussian Filter	σ	[0.5, 1, 1.5, 2]	0.5	13.6
Median Filter	Support Size	[3, 5, 7]	3	21.68
Bilateral Filter	σ_D σ_R	[1, 2, 3] [30, 50, 70]	σ_D=1 σ_R = 30	10.537
Perona-Malik	α κ T	[0.1, 0.25, 0.5] [5, 10, 15] [5, 10, 15]	α=0.1 κ=15 T=10	10.16

3) Different levels of noise are generated (**applyNoise**) with different scale σ levels (10,20 and 30). Noisy slices, 102 and 119 are plotted with their noise levels. (Fig. 1)

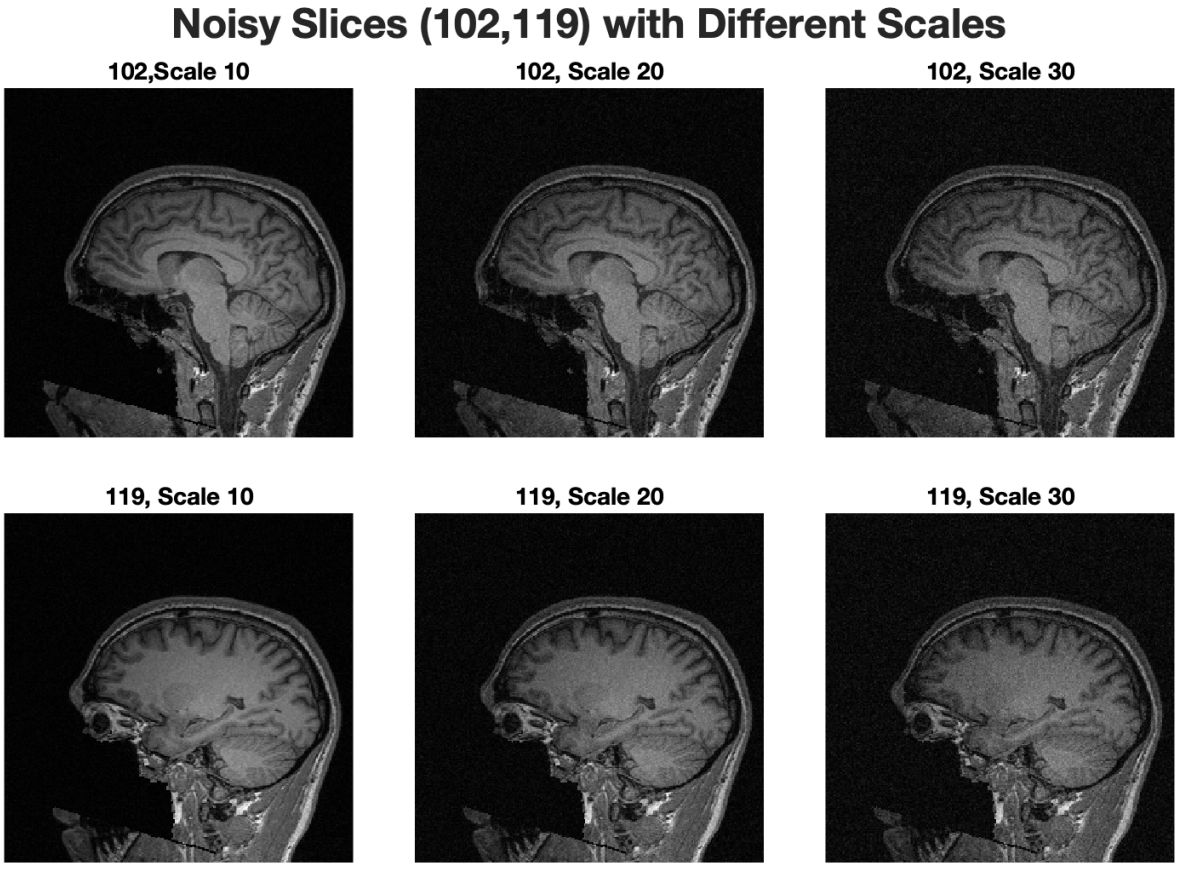

Fig.1: slices 102 and 119 with different noise levels (10,20 and 30).

4) Smoothed images are generated from the original image using four filters:

- **Gaussian Filter**
- **Median Filter**
- **Bilateral Filter**
- Scalar anisotropic diffusion filter (**Perona-Malik Filter**)

After tuning the filter parematers using grid search, the parameters for each filter are as follows:

Filter	Filter Size	Padding	Other Parameters
imgaussfilt3	2 * ceil (2 * σ) + 1 = 3	Replicate	• σ = 0.5
medfilt3	[3 x 3 x 3]	Symmetric	-
Bilateral Filter	-	-	• σ_D: Standard deviation for the domain (spatial) Gaussian kernel:1 • σ_R: Standard deviation for the range Gaussian kernel: 30
Perona-Malik	-	-	• α: Update rate: .1 • κ: Smoothness parameter: 15 • T: number of iterations: 10

- **imgaussfilt3** and **medfilt3** are MATLAB built-in functions.
- The bilateral filter (**BilateralFilterGraySep**) follows the algorithm which proposes to accelerate the filtering processing time by separating the 2D Gaussian computation into the 1D domain kernel followed by the 1D range kernel:

Algorithm 2 BilateralFilterGraySep(I, σ_d, σ_r) see reference [1]

Input: I, a grayscale image of size $M \times N$; σ_d, width of the 2D Gaussian domain kernel; σ_r, width of the 1D Gaussian range kernel
Output: A new filtered image of size $M \times N$
$(M, N) \leftarrow \text{Size}(I)$
$K \leftarrow \lceil 3.5 \cdot \sigma_d \rceil$
$I' \leftarrow \textit{Duplicate}(I)$
for all image coordinates $(u, v) \in M \times N$ **do**
  $S \leftarrow 0$
  $W \leftarrow 0$
  $a \leftarrow I(u, v)$
  for $m \leftarrow -K, \ldots, K$ **do**
    $b \leftarrow I(u + m, v)$
    $w_d \leftarrow e^{-\frac{m^2}{2\sigma_d^2}}$
    $w_r \leftarrow e^{-\frac{(a-b)^2}{2\sigma_r^2}}$
    $w \leftarrow w_d \cdot w_r$
    $S \leftarrow S + w \cdot b$
    $W \leftarrow W + w$
  end for
  $I'(u, v) \leftarrow \frac{1}{W} \cdot S$
end for
$I'' \leftarrow \textit{Duplicate}(I)$
for all image coordinates $(u, v) \in M \times N$ **do**
  $S \leftarrow 0$
  $W \leftarrow 0$
  $a \leftarrow I'(u, v)$
  for $n \leftarrow -K, \ldots, K$ **do**
    $b \leftarrow I'(u, v + n)$
    $w_d \leftarrow e^{-\frac{n^2}{2\sigma_d^2}}$
    $w_r \leftarrow e^{-\frac{(a-b)^2}{2\sigma_r^2}}$
    $w \leftarrow w_d \cdot w_r$
    $S \leftarrow S + w \cdot b$
    $W \leftarrow W + w$
  end for
  $I''(u, v) \leftarrow \frac{1}{W} \cdot S$
end for
return I''

Bilateral (Separated) Filter Algorithm

- Perona-Malik Filter is implemented in the function **PeronaMalik,** which takes as input parameters a 2D image, α, κ and T and is described by the algorithm:

Algorithm 3 PeronaMalik(I, α, κ, T)

Input: I , a grayscale image of size $M \times N$; α , update rate; κ, smoothness parameter; T, number of iterations;
Returns the modified image I.

Specify the conductivity function:
$g(d) := e^{-(d/\kappa)^2}$

$(M, N) \leftarrow \text{Size}(I)$
Create maps $D_x, D_y : M \times N \to \mathbb{R}$
for $n \leftarrow 1, \ldots, T$ **do**
  for all coordinates $(u, v) \in M \times N$ **do**

$$D_x(u, v) \leftarrow \begin{cases} I(u+1, v) - I(u, v) & \text{if } u < M - 1 \\ 0 & \text{otherwise} \end{cases}$$

$$D_y(u, v) \leftarrow \begin{cases} I(u, v+1) - I(u, v) & \text{if } v < N - 1 \\ 0 & \text{otherwise} \end{cases}$$

  end for
  for all coordinates $(u, v) \in M \times N$ **do**
    $\delta_0 \leftarrow D_x(u, v)$
    $\delta_1 \leftarrow D_y(u, v)$

$$\delta_2 \leftarrow \begin{cases} -D_x(u-1, v) & \text{if } u > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\delta_3 \leftarrow \begin{cases} -D_y(u, v-1) & \text{if } v > 0 \\ 0 & \text{otherwise} \end{cases}$$

    $I(u, v) \leftarrow I(u, v) + \alpha \cdot \sum_{k=0}^{3} g(|\delta_k|) \cdot \delta_k$
  end for
end for
return I

Peron-Malik Algorithm

The bilateral and Perona-Malik filters process 2D images using two helper functions: **computeBilateralFilter** and **computePeronaMalikFilter**. These functions filter the

entire image one slice at a time. For efficiency and adherence to best coding practices, these functions could eventually be combined into a single function.

5) Examine effects of smoothing on edges
   To better appreciate the impact of the four filters on edges, two regions of interest (ROI, function **getROI**) are defined for each level of noisy images (10, 20 and 30), and for the smoothed images related to the four filters (Gaussian, Median, Bilateral and Perona-Malik); the ROIS zoom into slices 102 or 119 of these images to focus on areas with sharp edges.

For each noise level, the ROIs and their smoothed versions, as processed by the four filters, are displayed for slices 102 and 119 (Fig. 2,3 and 4).

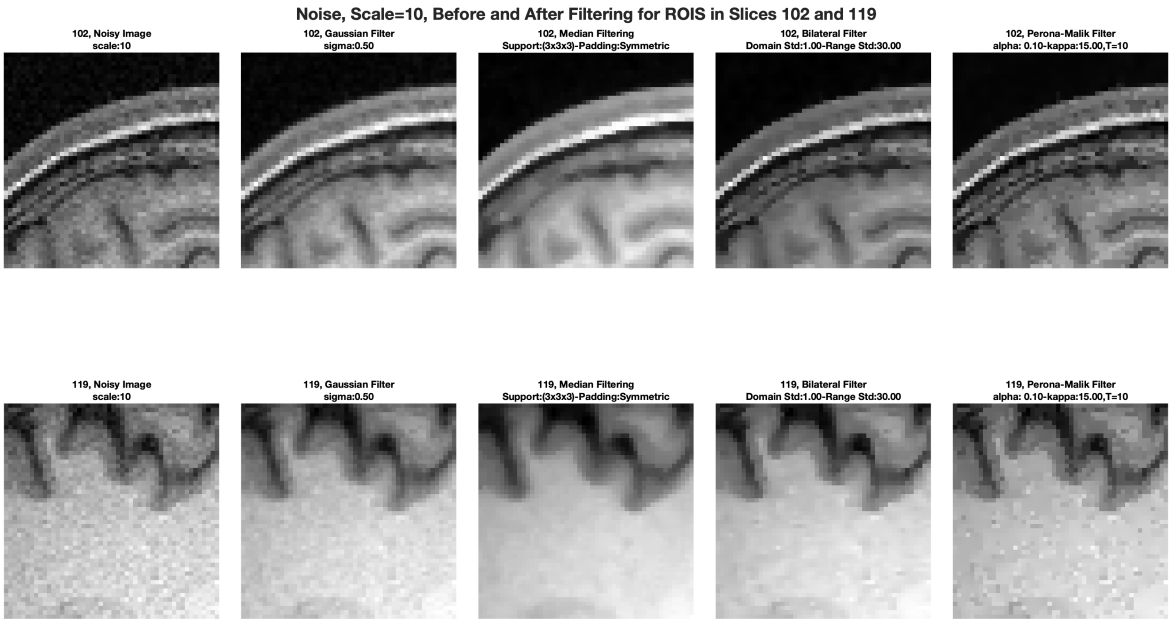

Fig. 2: At Noise Level 10: The first row displays the noisy ROI of slice 102 on the left, followed by its filtered versions using the four filters: Gaussian, Median, Bilateral, and Perona-Malik, in subsequent columns. The second row shows the noisy ROI of slice 119 on the left, along with its filtered versions using the same filters.

The subsequent plots illustrate similar information for ROIs at noise levels 20 and 30.

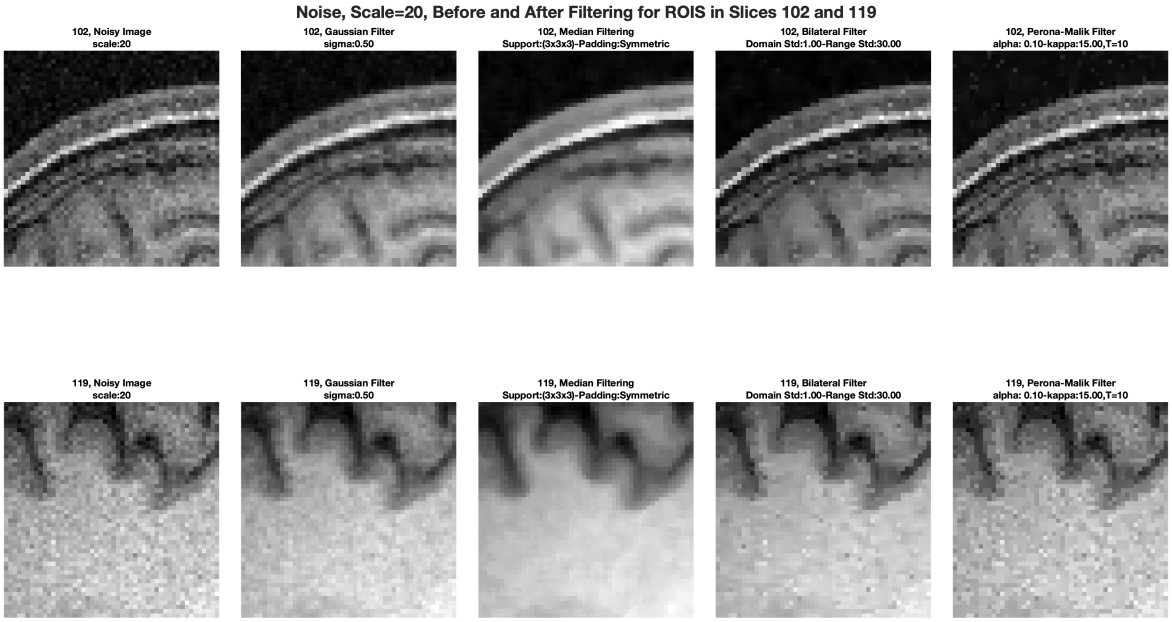

Fig. 3.: noisy ROIS at level 20, and its smoothed ROIS using Gaussian, median, bilateral and Perona-Malik filters.

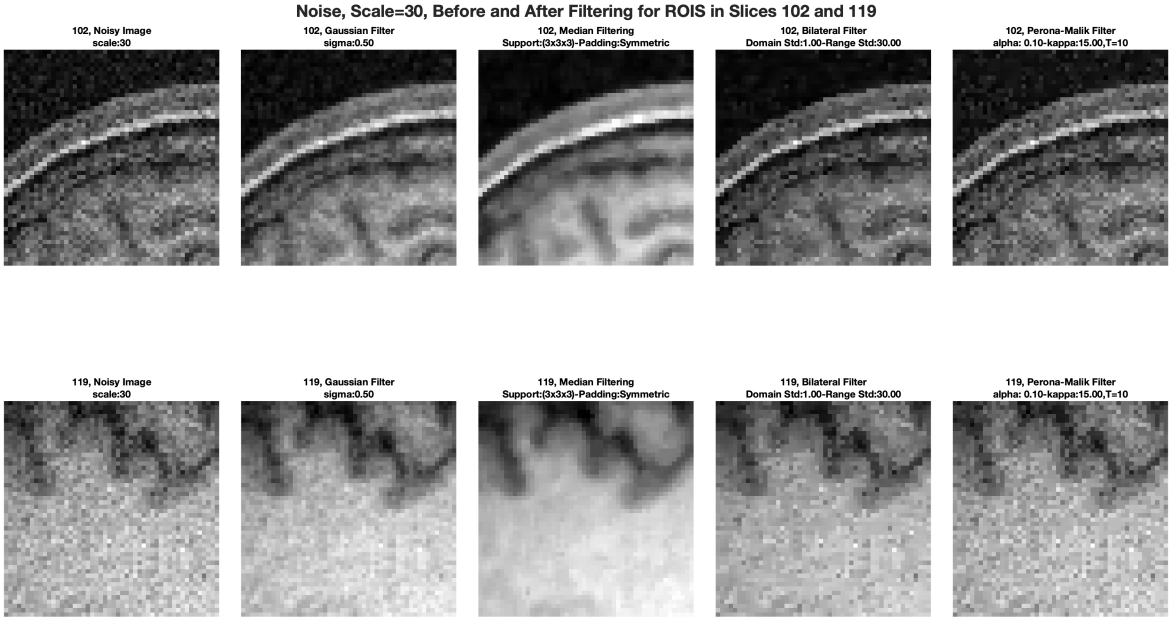

Fig. 4.: noisy ROIS at level 30, and its smoothed ROIS using Gaussian, median, bilateral and Perona-Malik filters.

- Both slices (102 and 119) display similar patterns with each filter; however, the variation in structure between the two slices highlights the filters' differential impacts on distinct anatomical features.
- Slice 119 ROI appears to have more complex structures that test the filters' ability to preserve edges without introducing artifacts, especially at higher noise levels.

Looking at these plots several comments can be made:

Filter	Noise Level 10	Noise Level 20	Noise Level 30
Gaussian	Effective smoothing but some blurring, leading to loss of edge sharpness.	Blurring increases, noticeable loss of fine details and edge clarity.	Significant blurring and detail loss.
Median	Reduces noise while maintaining edge details; balanced performance.	Still balances noise reduction and edge preservation but begins to lose fine details.	Less effective in maintaining edge clarity; smoothing leads to loss of details.
Bilateral	Preserves edges effectively with minimal artifacts; good performance.	Continues to maintain edge details effectively, though some artifacts appear.	Performs reasonably, still preserves edges, but artifacts start to increase.
Perona-Malik	Maintains edges well with minimal blurring; effective for low noise.	Slight artifact development while preserving edges; effective but needs tuning.	Most noticeable artifact introduction.

In addition, we can observe that in terms of:

- **Trade-offs Between Smoothing and Detail Preservation:**
  - Filters like the Gaussian and median, while effective in smoothing, tend to blur fine details, which might not be ideal when edge clarity is critical.
  - The bilateral and Perona-Malik filters are better at preserving edges, but both at higher noise levels introduce some artifacts.

6) Quality Performance Analysis

5.1) Voxel Squared Error Difference

We start by computing the per voxel squared error differences between the smoothed images generated using the four filters and the original image (**getFilterPerformance**) and that for each noise level (10,20 and 30). Next for each slice, 102 and 109; we show the error difference using a color map with a gradient color defined as red being the highest squared error and blue the lowest. It allows us to visually compare the

performance of the different filters under various noise levels (**plotFilterPerformances**) (Fig. 5,6 and 7).

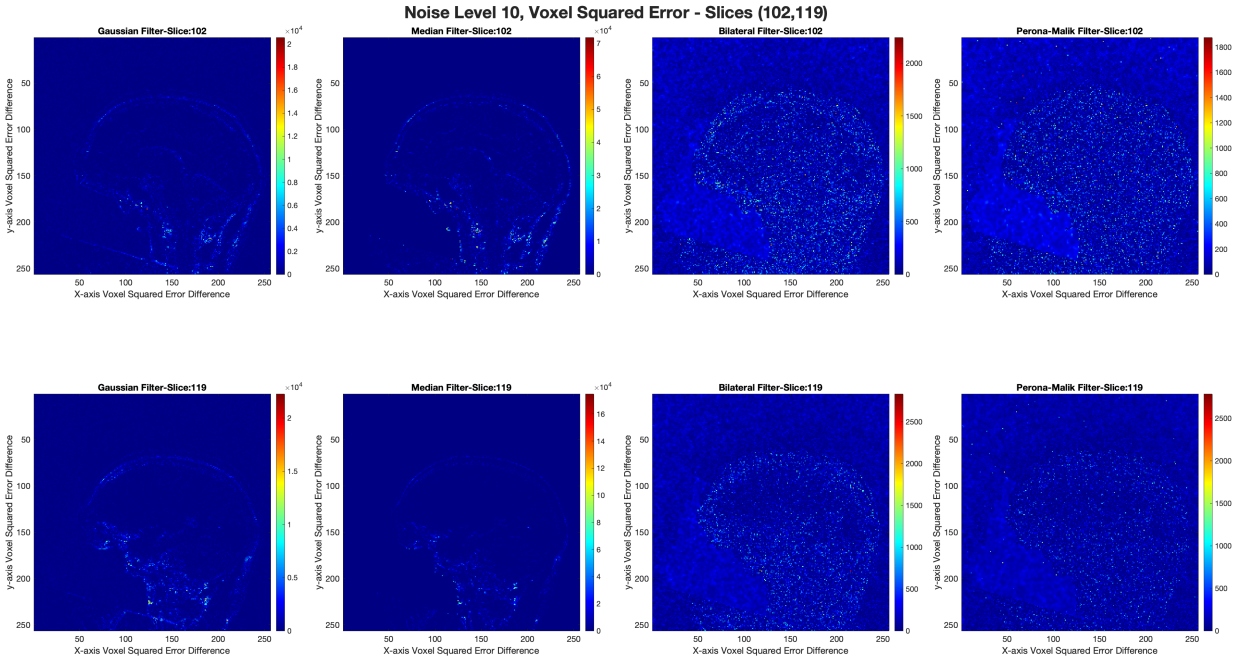

Fig. 5: At Noise Level 10: The first row displays the voxel squared error difference between the original image and the smoothed image generated using Gaussian filter, Median filter, Bilateral Filter and Perona-Malik filter. The second row shows the voxel squared error difference of slice 119 using the same filters.

The subsequent plots illustrate similar information at noise levels 20 and 30.

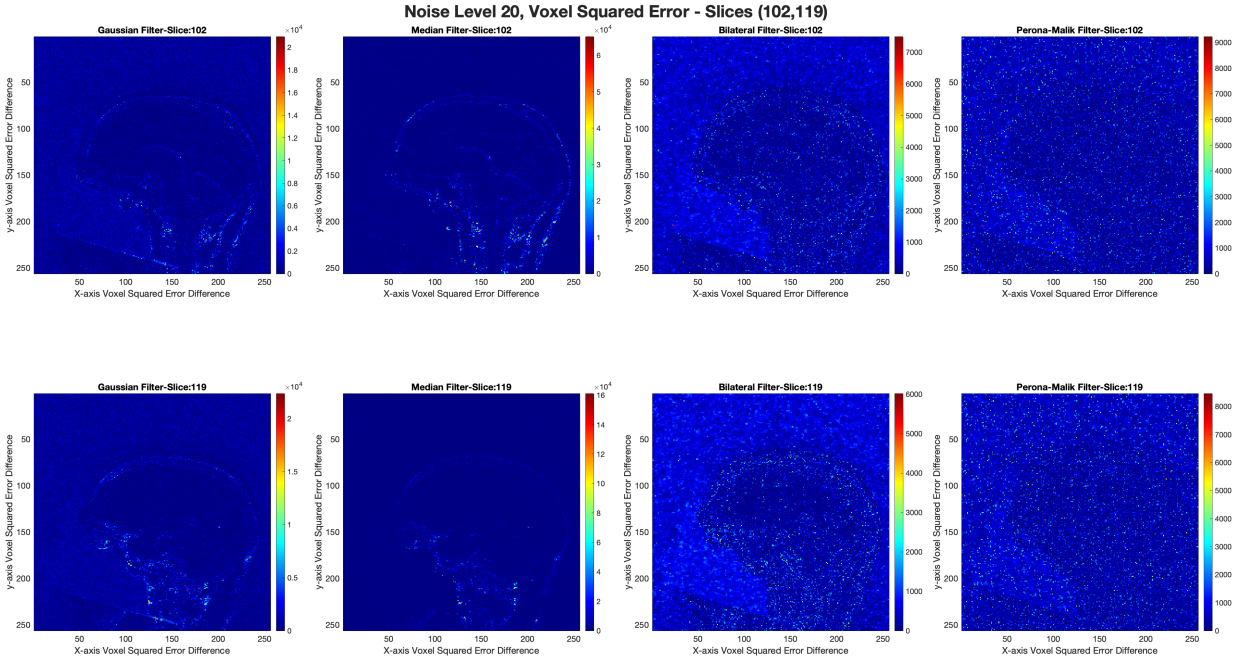

Fig. 6: Voxel Squared Error at level 20 applying filters: Gaussian, Median, Bilateral and Perona-Malik

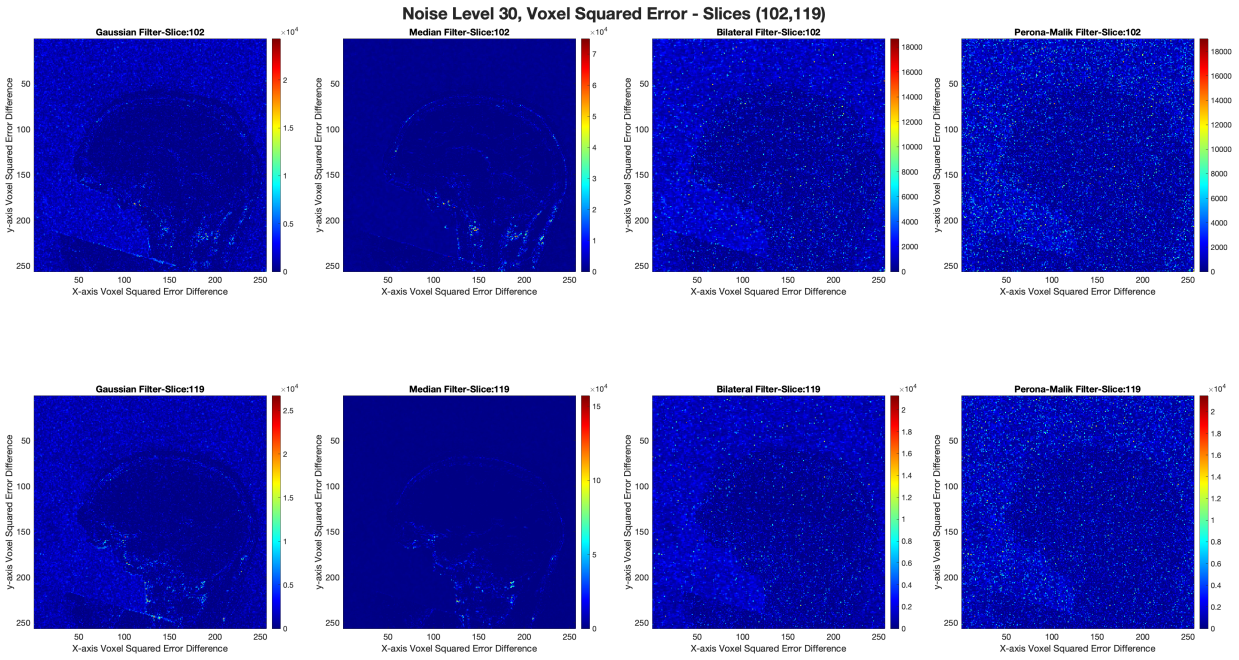

Fig. 7: Voxel Squared Error at level 30 applying filters: Gaussian, Median, Bilateral and Perona-Malik.

For these four filters, the highest errors are related to the edges. These filters may require tuning, but more details could be found in future work section.

Filter	Noise Level 10	Noise Level 20	Noise Level 30
Gaussian	Shows low error intensities; effective in noise reduction but some loss in edge details.	Increased error intensities, especially around edges.	Increased error intensities, especially around edges.
Median	Similar performances as the Gaussian filter.	Similar performances as the Gaussian filter.	Similar performances as the Gaussian filter.
Bilateral	Good edge preservation and low error intensities across most regions; effective filtering.	As the noise level increases, the errors decrease, the brain silhouette disappears showing that the filter improves in performances.	Similar performances compared to level 20. Some artifacts appear.
Perona-Malik	The performances are like the ones observed with the bilateral filter.	The performances are like the ones observed with the bilateral filter.	The performances are like the ones observed with the bilateral filter.

5.2) Mean Square Error

We compute the Mean Square errors for the three noise levels and the four filters (**getRMSE**) and plot it (**plotRMSE**) (Fig.8).

- Overall, as the noise level increases, the MSE values for all filters increase, indicating that higher noise levels result in greater errors, regardless of the filter used.
- At lowest noise level (10), the bilateral and Perona-malik filters have the lowest RMSE followed by the median filter and the Gaussian filters.
- The **Gaussian Filter** maintains a moderate MSE increase and appears to be competitive, especially at higher noise levels.
- The **median filter** initially performs worse than the Gaussian filter (noise level 10) but still shows a less severe increase in MSE at noise level 30 compared to the Gaussian filter.
- **The bilateral filter has a lower MSE** at noise level 10 and remains relatively stable compared to the median and Perona-Malik filters. Its increase in MSE is relatively linear to the noise suggesting maybe an easy improvement to the algorithm using a linear factor.
- **The Perona-Malik filter** shows the lowest MSE at noise level 10, demonstrating its effectiveness in edge preservation and noise reduction at lower noise levels. However, its MSE increases rapidly at noise level 30, passing the error values of the

Gaussian and Median filters. This suggests that while the Perona-Malik filter is initially effective, it may not handle high noise levels as efficiently, when using the current parameter configuration.

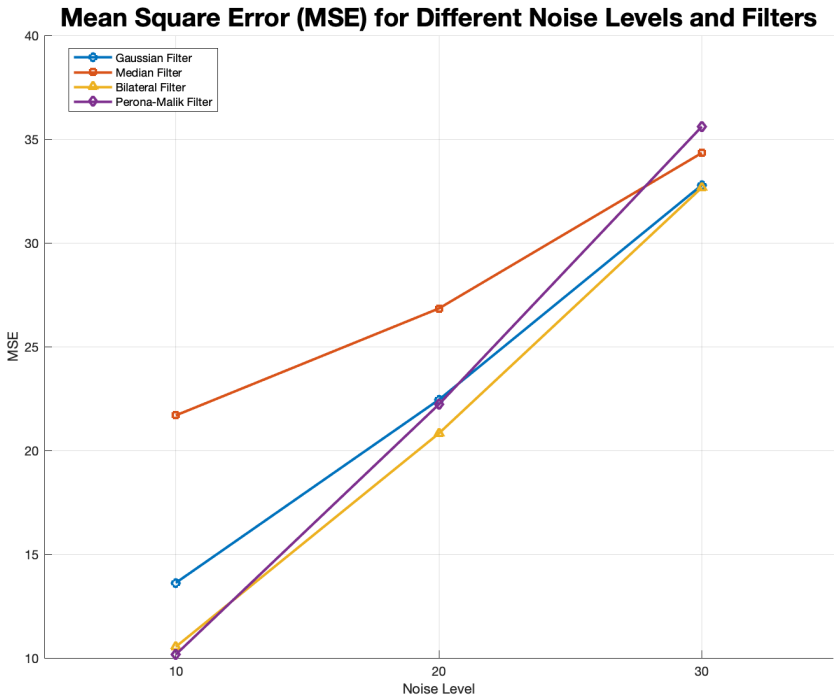

Fig. 8: MSE for Different Levels of Noise and filters: Gaussian, Median, Bilateral and Perona-Malik.

Challenges

- The initial challenge was understanding how to tune the different filters to obtain a filtering visually noticeable
- The bilateral and Perona-Malik algorithms were straightforward to implement but special care had to be made for pixels close to the edges of the images.

Future Directions

- The bilateral algorithm could be made more efficient by precomputing the weights related to spatial Gaussian using a KxK grid and pixels at the borders of the image could be handled using one of the common padding strategy (replicate, symmetric or zeros).
- The different filter parameters could be tuned further using a grid search and for loss function the voxel squared error difference, or other loss functions related to image quality assessment (IQA) metrics (for ex., Peak Signal to Noise Ratio (PSNR), Mean Absolute Error (MAE), and Structural Similarity Index (SSIM)).
- Among a variety of specific improvement that can be made, we highlight the following:

1. **Adaptive Gaussian Filtering**: Instead of using a fixed standard deviation (sigma) across the image, an adaptive approach can be employed where sigma varies depending on local image characteristics (e.g., intensity gradient).
2. **Median Filter**

- **Adaptive Window Size**: The standard median filter uses a fixed window size, which may not be ideal for all regions of the image. An adaptive approach that adjusts the window size based on local noise levels or edge presence can improve its ability to reduce noise while maintaining image details.
- **Weighted Median Filter**: Instead of treating all pixels in the window equally, a weighted median filter can prioritize pixels closer in intensity to the central pixel. This modification can help maintain edges more effectively while still reducing noise.
- **Edge-Preserving Variants**: Developing an edge-preserving median filter that incorporates information about local image gradients.

3. **Bilateral Filter**

- **Optimization of Parameters (Domain and Range Sigma)**: Fine-tuning the domain (spatial) and range (intensity) parameters based on image characteristics. An automated or adaptive parameter selection mechanism could dynamically adjust these values for different regions of the image.
- **Bilateral Guiding**: This method uses a guidance image (e.g., an edge map or an image with enhanced contrast) in addition to the original image for filtering.

4. **Perona-Malik Filter**

- **Dynamic Parameter Adjustment**: The Perona-Malik filter uses parameters like kappa and alpha that control diffusion rates. Introducing an adaptive mechanism that adjusts these parameters based on local image characteristics (e.g., gradient magnitude) can prevent over-smoothing and reduce artifacts.
- **Stopping Criterion**: The filter is iterative and often runs for a set number of iterations (T). A more sophisticated stopping criterion that halts the process when a convergence or stability condition is met (such as when the rate of change in the image error becomes minimal) can minimize artifacts and over-processing.
- **Alternative Diffusion Functions**: Modifying the diffusion function used in the Perona-Malik equation to incorporate additional edge-preserving terms (e.g., incorporating curvature information); can improve the filter's ability to reduce noise while retaining fine structures.
- **Hybrid Diffusion Models**: Combining the Perona-Malik filter with other edge-preserving methods, like the bilateral filter, could result in a hybrid model that leverages the strengths of both techniques to minimize noise while maximizing edge preservation.