Introduction

For this assignment, using a 3D MRI brain image data (*sub-11_T1w.nii.gz*), we want to:

- Implement the Geometry Preserving Anisotropic Diffusion Filter (**GPAD**)
- Examine effects of smoothing on edges by enlarging over a small Region of Interest (ROI) using slices 102 and 119.
- Quantify the GPAD filter performances by calculating per voxel squared error difference between smoothed data and original non-noisy data.
- Analyze mean square error (MSE) for three noise levels (10,20 and 30) and five filters: Gaussian, Median, Bilateral, Scalar diffusion and GPAD filter.

Main Work

1) We upload 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.
2) The image is flipped to facilitate image manipulation.
3) Before starting any filtering, we run a grid search, looking for the optimal parameters of the filter with the lowest root mean squared error. Unless needed; please do not uncomment this part as it is resource intensive and takes a while.

   We found that best parameters to obtain the lowest MSE as follows:

Parameter	Description	Range	Optimal Parameter
T	Number of iterations	[5, 10, 15]	10
dt	Time increment	[0.1, 0.2, 0.3]	0.2
σ_g	Width of Gaussian for smoothing the gradient	[0.5, 1, 1.5]	1
σ_s	Width of Gaussian for smoothing the structure matrix	[1, 2, 3]	2
a_1	Diffusion parameter (along contour)	-	0.5
a_2	Diffusion parameter (perpendicular to contour)	-	0.9

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

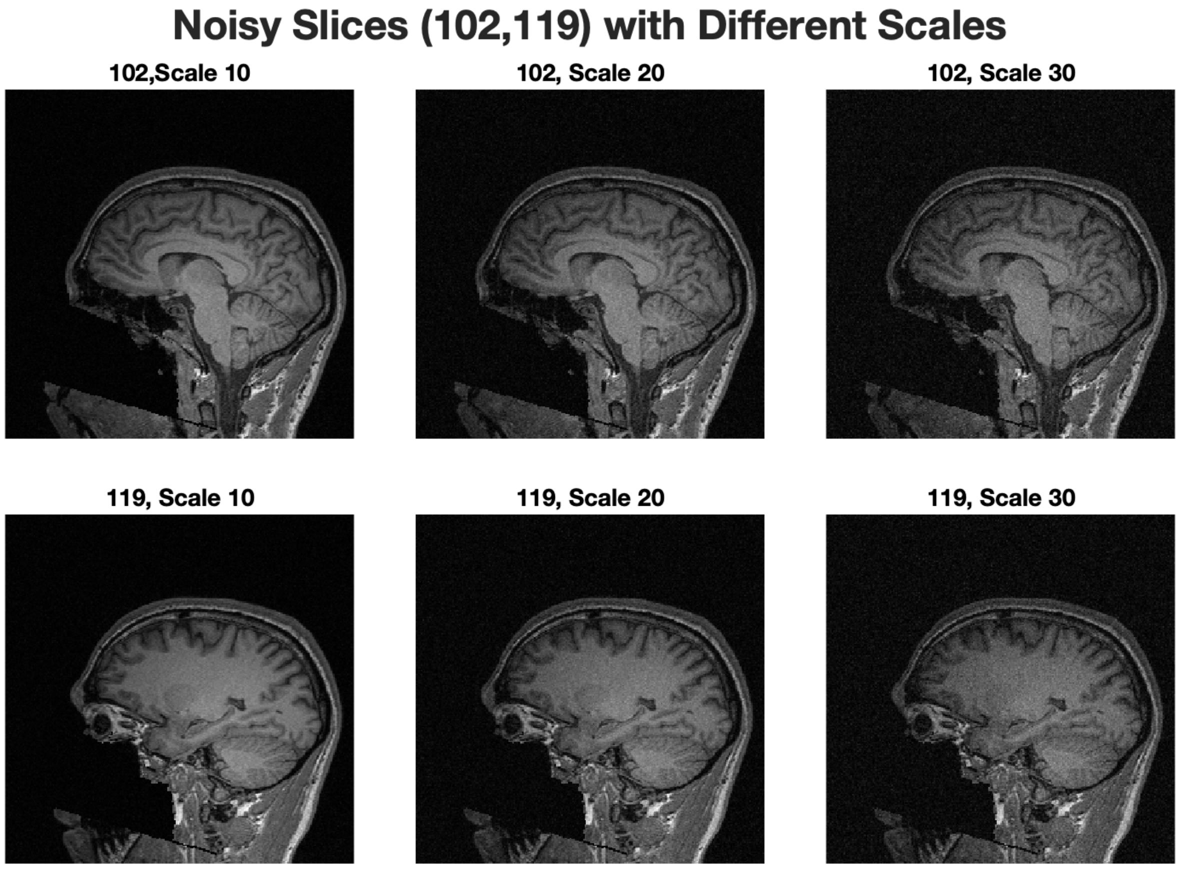

Fig.1: slices 102 and 119 with different noise levels (10,20 and 30).

5) The brain MRI image is smoothed using the GPAD filter.

- TschumperleDeriche Filter is implemented in the function **GeometryPreservingAnisotropicDiffFilter,** which takes as input parameters:
    - I: a 2D image
    - dt: time increment
    - σ_g: width of Gaussian for smoothing the gradient
    - σ_s: width of Gaussian for smoothing the structure matrix G
    - a_1 and a_2: diffusion parameters for direction of min//mx variations

The algorithm returns the modified image I.

Geometry Preserving Anisotropic Diffusion Filter Algorithm

Algorithm 1 TschumperleDericheFilter($I, T, d_t, \sigma_g, \sigma_s, a_1, a_2$)

Input: $\mathbf{I} = (I_1, ..., I_K)$, image of size $M \times N$ with K channels;
T, number of iterations; d_t, time increment;
σ_g, width of Gaussian for smoothing the gradient; σ_s, width of Gaussian for smoothing the structure matrix; a_1, a_2, diffusion parameters for directions of min./ max. variation, respectively.
Returns the modified image I.

```
 1: Create maps:
 2: $\mathrm{D} : K \times M \times N \to \mathbb{R}^2$
 3: $\mathrm{H} : K \times M \times N \to \mathbb{R}^{2\times 2}$
 4: $\mathrm{G} : M \times N \to \mathbb{R}^{2\times 2}$
 5: $\mathrm{A} : M \times N \to \mathbb{R}^{2\times 2}$
 6: $\mathrm{B} : K \times M \times N \to \mathbb{R}$
 7: for $t \leftarrow 1, \ldots, T$ do
 8:     for $k \leftarrow 1, \ldots, K$ and all coordinates $(u, v) \in M \times N$ do
 9:         $\mathrm{D}(k, u, v) \leftarrow \begin{pmatrix} (I_k * H_x^{\triangledown})(u, v) \\ (I_k * H_y^{\triangledown})(u, v) \end{pmatrix}$
10:         $\mathrm{H}(k, u, v) \leftarrow \begin{pmatrix} (I_k * H_{xx}^{\triangledown})(u, v) & I_k * H_{xy}^{\triangledown})(u, v) \\ (I_k * H_{xy}^{\triangledown})(u, v) & I_k * H_{yy}^{\triangledown})(u, v) \end{pmatrix}$
11:     end for
12:     $\mathrm{D} \leftarrow \mathrm{D} * H_G^{\sigma_d}$
13:     for all coordinates $(u, v) \in M \times N$ do
14:         $\mathrm{G}(k, u, v) \leftarrow \sum_{k=1}^{K} \begin{pmatrix} (\mathrm{D}_x(k, u, v))^2 & \mathrm{D}_x(k, u, v) \cdot \mathrm{D}_y(k, u, v) \\ \mathrm{D}_x(k, u, v) \cdot \mathrm{D}_y(k, u, v & (\mathrm{D}_y(k, u, v))^2 \end{pmatrix}$
15:     end for
16:     $\mathrm{G} \leftarrow \mathrm{G} * H_G^{\sigma_g}$
17:     for all coordinates $(u, v) \in M \times N$ do
18:         $(\lambda_1, \lambda_2, e_1, e_2) \leftarrow EigenValues(\mathrm{G}(u, v))$
19:         $\hat{e}_1 \leftarrow \begin{pmatrix} \hat{x}_1 \\ \hat{y}_1 \end{pmatrix} = \frac{e_1}{\|e_1\|}$
20:         $c_1 \leftarrow \frac{1}{(1+\lambda_1+\lambda_2)^{a_1}}$, $c_2 \leftarrow \frac{1}{(1+\lambda_1+\lambda_2)^{a_2}}$
21:         $\mathrm{A}(u, v) \leftarrow \sum_{k=1}^{K} \begin{pmatrix} c_1 \cdot \hat{y}_1^2 + c_2 \cdot \hat{x}_1^2 & (c_2 - c_1) \cdot \hat{x}_1 \cdot \hat{y}_1 \\ (c_2 - c_1) \cdot \hat{x}_1 \cdot \hat{y}_1 & c_1 \cdot \hat{x}_1^2 + c_2 \cdot \hat{y}_1^2 \end{pmatrix}$
22:     end for
23:     $\beta_{max} \leftarrow -\infty$
24:     for $k \leftarrow 1, \ldots, K$ and all coordinates $(u, v) \in M \times N$ do
25:         $\mathrm{B}(k, u, v) \leftarrow trace(\mathrm{A}(u, v) \cdot \mathrm{H}(k, u, v))$
26:         $\beta_{max} \leftarrow max(\beta_{max}, |\mathrm{B}(k, u, v)|)$
27:     end for
28:     $\alpha \leftarrow d_t / \beta_{max}$
29:     for $k \leftarrow 1, \ldots, K$ and all coordinates $(u, v) \in M \times N$ do
30:         $I_k(u, v) \leftarrow I_k(u, v) + \alpha \cdot \mathrm{B}(k, u, v)$
31:     end for
32: end for
33: return $I$
```

Notice that A and H are two symmetric matrices, thus we can take advantage of MATLAB vectorization to compute the trace using elementwise multiplication (see **computeBAndBetaMax** function):

```
B(k, :, :) = A11 .* H11 + 2 * A12 .* H12 + A22 .* H22;
```

6) We examine effects of smoothing on edges

- We examine the effects of smoothing on edges by extracting a small Region of Interest (ROI) using slices 102 and 119. We call the function **getROI** to retrieve the ROIs for slices 102 and 119 from the noise images (level 10, 20 and 30) and from the smoothed image generated by the GPAD filter.
  We then, display the ROIS at each noise level and their smoothed equivalent from the image processed using the GPAD filter for slices 102 and 119 (Fig. 2,3 and 4).

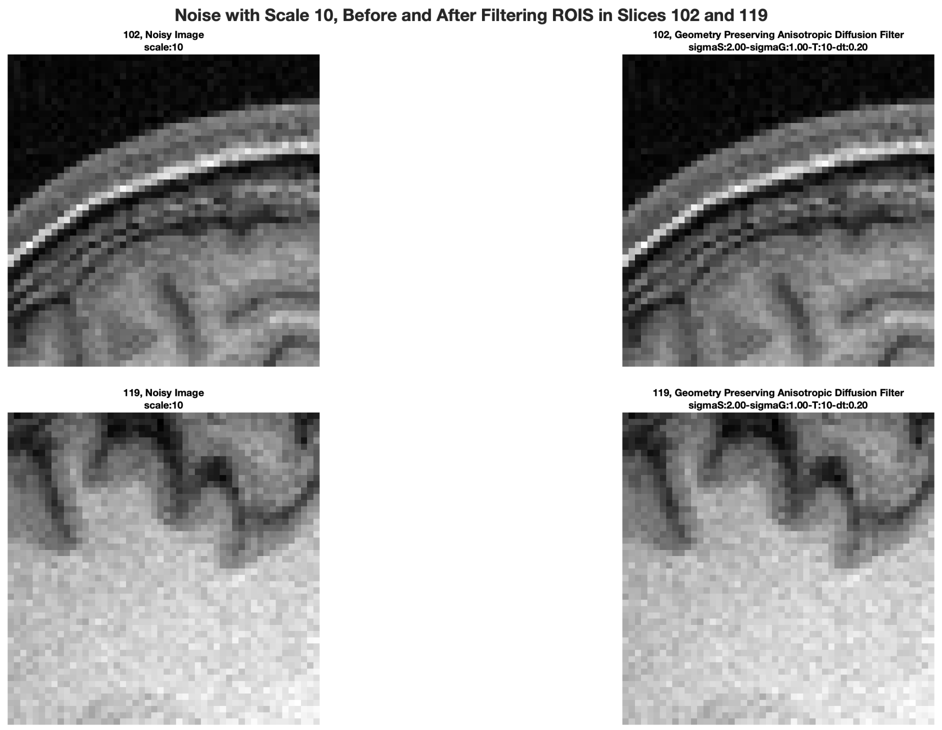
Noise with Scale 10, Before and After Filtering ROIS in Slices 102 and 119
102, Noisy Image
scale:10
102, Geometry Preserving Anisotropic Diffusion Filter
sigmaS:2.00-sigmaG:1.00-T:10-dt:0.20
119, Noisy Image
scale:10
119, Geometry Preserving Anisotropic Diffusion Filter
sigmaS:2.00-sigmaG:1.00-T:10-dt:0.20

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 GPAD filter on the right. The second row shows the noisy ROI of slice 119 on the left, along with its filtered versions using the same filter.

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

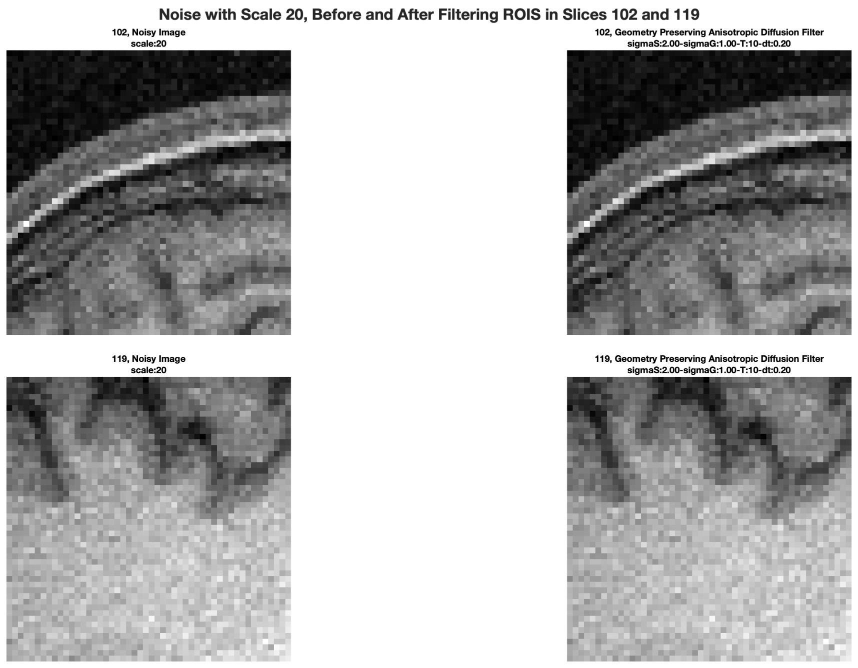
Noise with Scale 20, Before and After Filtering ROIS in Slices 102 and 119
102, Noisy Image scale:20
102, Geometry Preserving Anisotropic Diffusion Filter sigmaS:2.00-sigmaG:1.00-T:10-dt:0.20
119, Noisy Image scale:20
119, Geometry Preserving Anisotropic Diffusion Filter sigmaS:2.00-sigmaG:1.00-T:10-dt:0.20

Fig. 3.: noisy ROIS at level 20, and its smoothed ROIS using GPAD filter.

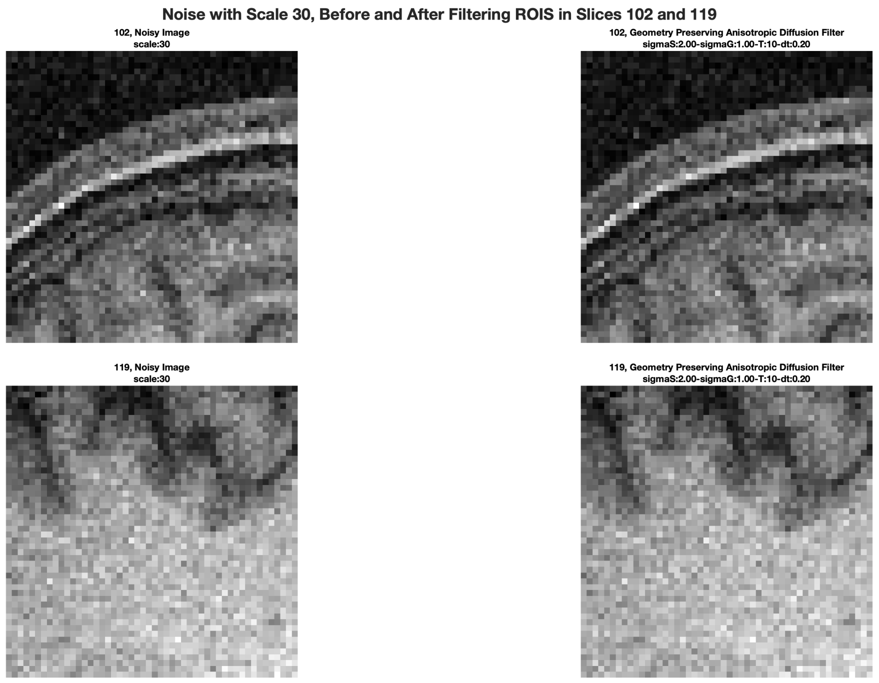
Noise with Scale 30, Before and After Filtering ROIS in Slices 102 and 119
102, Noisy Image scale:30
102, Geometry Preserving Anisotropic Diffusion Filter sigmaS:2.00-sigmaG:1.00-T:10-dt:0.20
119, Noisy Image scale:30
119, Geometry Preserving Anisotropic Diffusion Filter sigmaS:2.00-sigmaG:1.00-T:10-dt:0.20

Fig. 4.: noisy ROIS at level 30, and its smoothed ROIS using GPAD filter.

- 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 have more complex structures that test the filters' ability to preserve edges without introducing artifacts, especially at higher noise levels.

Examining these different images of slices 102 and 119, we observe:
- That the filter successfully maintains clear and well-defined edges, even at higher noise levels like 30 while preventing significant blurring of edges.
- The filter smooths regions with uniform intensity while keeping the essential structural features intact.

7) Quality Performance Analysis

5.1) Voxel Squared Error Difference
We compute the voxel squared error difference between the original brain image and the denoised image by the GPAD filter (**getVoxelSquaredErrorDifference**) for each level of noise (10,20 and 30). Then, we plot the voxel squared error difference for each level of noise and slice 102 and 119 (**plotVoxelSquaredErrorDifference**).
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 GPAD filter under various noise levels (**Fig. 5**).

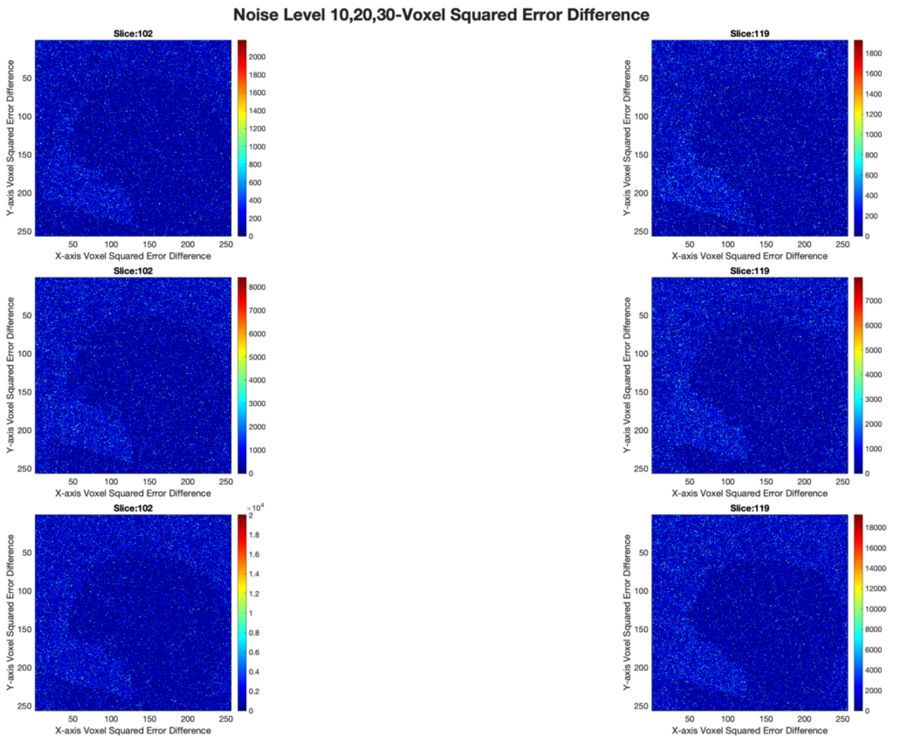

Fig. 5: The first row is the voxel squared error difference for slice 102 and 119. Subsequent rows have similar content for noise level 20 and 30.

Overall, these plots are predominantly blue with some lighter dots, indicating that the GPAD filter maintains a generally low squared error difference per voxel, demonstrating its effective noise suppression, especially at lower noise levels. The color bar range increases significantly as noise levels increase, revealing more pronounced voxel error differences:

- For **slice 102**, the maximum voxel squared error difference starts at approximately **2000** for noise level 10 and rises to nearly **20,000** at noise level 30.
- For **slice 119**, the maximum error at noise level 10 is about **1800**, escalating to around **18,000** at noise level 20.

These observations indicate the challenge faced by the GPAD filter as noise levels rise, highlighting its limitations when handling high-noise images.

Despite these differences, the voxel squared error difference range being similar for both slices suggests that the GPAD filter handles various edge types consistently. This indicates that it preserves structural detail effectively.
To improve performance, further parameter tuning or the incorporation of additional filtering steps might be necessary.

5.2) Mean Square Error

We compute the Mean Square errors for the three noise levels and the five filters (**getRMSE**) and plot it (**plotRMSE**) (Fig.6).
This plot that shows:

- 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 the lowest noise level (10), the bilateral, Perona-malik and GPAD filters have the lowest MSE (**~0**) followed by the Gaussian and median filters.
- The **GPAD** filter MSE increases significantly with the level of noise; finishing with the worst MSE (**37**) among all the filters; indicating that it is not handling the noise efficiently with its current parameter configuration and may need more tuning (T,dt, σ_g, σ_s).
- By comparison, the **bilateral filte**r has a low MSE (**0**) at noise level 10 and its MSE increases the less among all the filters (**MSE:33 at noise level 30**); suggesting it is better at handling increased level of noise and the smoothed image quality is closest to the original image even at higher noise levels.
- The **Gaussian Filter** maintains a moderate MSE increase and appears to be competitive, especially at higher noise levels, being at the same MSE than the bilateral filter at noise level 30 (**MSE:33**).
- The **median filter** initially performs worse than the Gaussian filter (**MSE: 22 at noise level 10**) but still shows a less severe increase in MSE (**34**) at noise level 30 compared to the Perona-Malik and GPAD filters
- **The Perona-Malik filter** shows the lowest MSE at noise level 10 (**0**), demonstrating its effectiveness in edge preservation and noise reduction at lower noise levels. However, its MSE increases rapidly like the GPAD filter (**MSE:36 at noise level 30**). Again, 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 (α, κ, T).

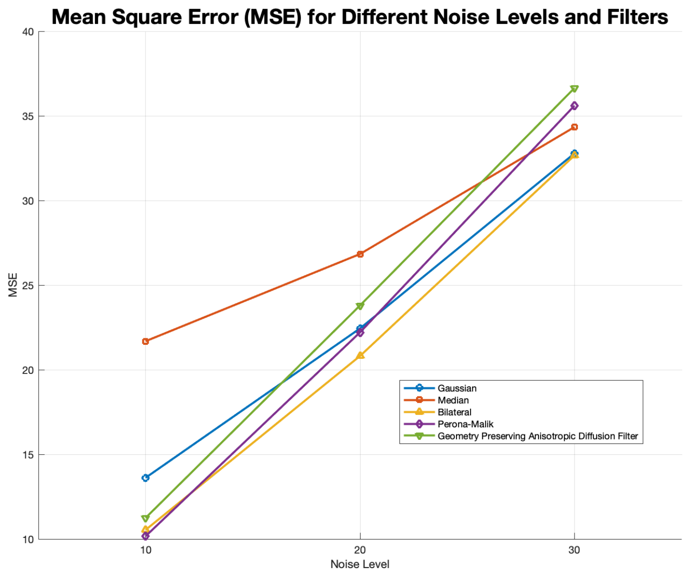

Fig. 6: MSE for Different Levels of Noise and filters: Gaussian, Median, Bilateral, Perona-Malik and GPAD Filter.

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 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.