Computer Imaging: Image Denoise++

Anonymous CVPR submission

Paper ID *****

1. Introduction

1.1. Definition of image denoising

Image denoising is the process of removing unwanted noise from images while preserving essential details. Noise can arise from sensor limitations, environmental interference, or compression artifacts, degrading image quality. Effective denoising is crucial in applications like medical imaging and digital photography.

1.2. Importance of image denoising

A key challenge in denoising is distinguishing noise from fine image details. Excessive smoothing can blur important features, while inadequate noise removal affects clarity. Various techniques, from traditional Gaussian filters to advanced machine learning models, aim to balance these trade-offs.

1.3. Overview of the project

This project evaluates different image denoising algorithms by optimizing their parameters and testing them on a dataset. The goal is to identify the most effective approach under specific conditions and analyze the trade-offs between noise reduction and detail preservation.

2. Methods

2.1. Dataset

This project utilizes the Kodak dataset, a widely used benchmark for image processing and compression tasks. The dataset consists of 24-bit true-color images of natural scenes, offering diverse textures, colors, and details, making it suitable for denoising evaluation. Originally released by the Eastman Kodak Company for unrestricted usage, this dataset is widely adopted as a standard benchmark for image compression and processing tasks. Its high-quality images make it suitable for testing denoising algorithms, as it provides diverse challenges in noise removal while preserving image details.

For training and parameter tuning, we used 14 images (kodim_11.png to kodim_24.png), except for the Transformer-based Restomer algorithm, which was pretrained model. The test set comprises 10 images (kodim_01.png to kodim_10.png) for final evaluation.

2.2. Denoising Algorithms

This project evaluates five denoising methods: Bilateral Filter, Non-Local Means, Total Variation Filtering, Wavelet Denoising, and Restormer. Each method is summarized below:

- **Bilateral Filter**: A non-linear, edge-preserving filter that smooths noise while maintaining edges by considering spatial closeness and intensity similarity.
- **Non-Local Means**: Reduces noise by averaging similar patches across the image, preserving textures and fine details.
- **Total Variation (TV) Filtering**: Minimizes total variation to remove noise while preserving edges, using the Chambolle algorithm.
- **Wavelet Denoising**: Uses wavelet transform to decompose the image, applying thresholding to remove noise while retaining important structures.
- **Restormer**: A transformer-based deep learning model for image restoration. The pretrained model was applied directly to test images without parameter tuning.

2.3. Parameter Tuning

We optimized denoising parameters using Kodak images #11–#24, except for Restormer, which was used as a pre-trained model. We used the algorithms from the `skimage.restoration` library.

For **Bilateral Filter**, we tested `win_size` {3, 5, 7, 11, 15, 17, 21}, `sigma_color` {0.005–0.2}, and `sigma_spatial` {1, 3, 5, 7, 10}, totaling 245 combinations. The best configuration minimized Mean Squared Error (MSE) and maximized Peak Signal-to-Noise Ratio (PSNR).

For **Non-Local Means**, we tested `h` {0.1–1.0}, `patch_size` {3, 5, 7}, `patch_distance` {3, 5, 7}, and `fast_mode` {True, False}, totaling 30 combinations. The optimal parameters were selected based on MSE and PSNR.

For **Total Variation (TV) Filtering**, we explored `weight` {0.1–2.0}, `max_num_iter` {100–500}, and `eps` {0.0001, 0.001}, totaling 50 combinations, selecting the best set based on MSE and PSNR.

For **Wavelet Denoising**, we tested `wavelet` {'haar', 'db1', 'db2', 'sym2', 'sym9'}, `method` {'BayesShrink', 'VisuShrink'}, `mode` {'soft', 'hard'}, and `rescale_sigma` {True, False}, totaling 40 combinations, optimizing based on MSE and PSNR.

Restormer was applied without tuning due to computational constraints.

2.4. Evaluation Metric

To evaluate denoising performance, we used Root Mean Squared Error (RMSE), which measures pixel-wise differences between the ground truth and denoised images. Root Mean Square Error (RMSE) is sometimes preferred over Mean Squared Error (MSE) because it provides a measure of error that is in the same unit as the target variable, making it easier to interpret and compare. Additionally, RMSE penalizes larger errors more than MSE, making it more sensitive to outliers. RMSE is computed as:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(I_{\text{original}}(i) - I_{\text{denoised}}(i))^2}$$

where N is the total number of pixels in the image, and $I_{\text{original}}(i)$ and $I_{\text{denoised}}(i)$ represent the pixel values of the original and denoised images, respectively.

For each test image, we calculated the RMSE and then averaged the values across all 10 test images to obtain the average RMSE for each algorithm. This average RMSE provides a single quantitative measure of the denoising performance, enabling direct comparison among the five methods.

A lower average RMSE indicates better performance, as it signifies that the denoised image is closer to the ground truth. While RMSE is the primary metric, visual inspection was also performed to ensure that denoising did not introduce artifacts or excessive smoothing.

3. Results

3.1. Visual Comparisons

The below images show Kodak testset's original/noised/denoised image for each denoiser.

Figure 1. Original Test Images

Figure 2. Noisy Images by adding Gaussian noise

Figure 3. Denoised Image using Bilateral filter

Figure 4. Denoised Images using Non-local means filter

Figure 5. Denoised Image using Total Variation filter

Figure 6. Denoised Images using Wavelet filter

Figure 7. Denoised Image using Restormer algorithm

3.2. Performance Metrics

The average RMSE values for the five denoising algorithms were calculated across the 10 test images. The results are as follows:

Denoiser	Average RMSE
Bilateral	0.0680
Non-local Means	0.0897
Total Variation	0.0761
Wavelet	0.0714
Restormer	18.5606

Table 1. Average RMSE values for different denoising algorithms. Lower RMSE indicates better performance.

To explain the high RMSE for Restormer (Table 1), we analyzed the RMSE for each image. Restormer had consistently higher RMSE than the other four algorithms across most test images. Notably, image 9 had the highest RMSE for all algorithms, while image 8 had the lowest RMSE for Restormer and relatively low RMSE values for the other algorithms as well. (Table 2)

	Bilateral	Non-local Means	TV Chambolle	Wavelet	Restormer
Image 1	0.0555	0.0604	0.0714	0.0742	16.8295
Image 2	0.0881	0.1821	0.1057	0.1473	32.7376
Image 3	0.0354	0.0327	0.0603	0.0613	12.0981
Image 4	0.0444	0.0565	0.0686	0.0842	15.6871
Image 5	0.0430	0.0368	0.0695	0.0431	10.0815
Image 6	0.0431	0.0362	0.0716	0.0523	10.4813
Image 7	0.0322	0.0281	0.0577	0.0495	10.3340
Image 8	0.0470	0.0350	0.0713	0.0127	7.5867
Image 9	0.2333	0.3185	0.1397	0.1239	49.5303
Image 10	0.0580	0.1112	0.0455	0.0652	20.2401

Table 2. RMSE values for denoising algorithms. Lower RMSE indicates better performance.

3.3. Visual Comparisons

Figures 3, 4, 5, 6, and 7 show example images before and after denoising for each algorithm. Visual inspection revealed that Restormer produced the cleanest images with minimal noise. Bilateral filter and Non-local means also showed good denoising but retained some noise. Total Variation Chambolle resulted in partial color shifts, while Wavelet denoising exhibited color shifts, especially in noisier images, but less so in images with less noise (e.g., image 8, the house image).

3.4. Analysis of Results

Based on the average RMSE values, the Bilateral Filter achieved the best performance, followed by Wavelet, Total Variation Chambolle, and Non-local means. Restormer had the highest average RMSE (18.5606), which initally raised concerns about calculation errors. However, upon analyzing individual images, it was confirmed that Restormer consistently produced higher RMSE values.

The analysis of specific images revealed that image #09 (Figure 8) had the highest RMSE with the Restormer model, which was attributed to the noisy version of this image being significantly degraded compared to the original.

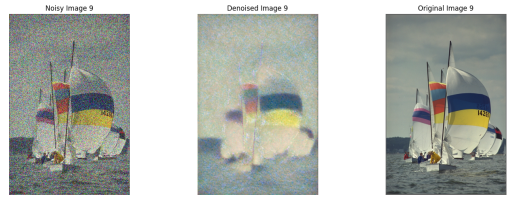

Figure 8. Noisy/Denoised/Original Image #09 using Restormer Denoiser

Conversely, image #08 (Figure 9) had the lowest RMSE values with the Restormer denoiser, likely because its noisy version was already visually similar to the original image.

Figure 9. Noisy/Denoised/Original Image #08 using Restormer Denoiser

Despite the numerical results, the visual comparisons suggested that the Restormer denoised images had the best visual quality, with minimal visible noise and high fidelity. Bilateral Filter and Non-local means images improved the images but retained some noise, while Total Variation Chambolle and Wavelet images showed noticeable color distortions.

The higher RMSE for Restormer could be due to its pre-trained nature, not optimized for the specific dataset. Deep learning models like Restormer prioritize perceptual quality over pixel accuracy, while classical algorithms are fine-tuned for the dataset, leading to lower RMSE but potentially inferior visual quality. This discrepancy highlights the limitations of RMSE as the sole evaluation metric and underscores the importance of incorporating perceptual evaluations for image quality assessment.

4. Conclusion

4.1. Best Denoiser

In terms of numerical evaluation, the Bilateral Filter achieved the best average RMSE (0.0680), followed

by Wavelet (0.0714). However, the Restormer algorithm, with an average RMSE of 18.5606, produced visually superior results. This suggests that RMSE alone may not fully capture the perceptual quality of the denoised images.

The analysis also highlighted the impact of the original noise levels on performance. Image #08, which was less noisy to begin with, had low RMSE values across all methods, while the highly degraded image #09 had the highest RMSE values.

Overall, the classical denoising algorithms provided competitive results in terms of RMSE, but the Restormer algorithm excelled in visual quality, demonstrating its potential for practical applications where perceptual fidelity is critical.

4.2. Challenges Encountered

A primary challenge was selecting the right denoising algorithm. With numerous options, it was difficult to determine which algorithm would best suit the dataset and produce optimal results. This required extensive research on different algorithms, evaluating their strengths and weaknesses, and understanding how each might behave with the noisy images in the dataset. I eventually tried all of the algorithms professor suggested in the project instructions and the pretrained Restormer algorithm.

Integrating a pretrained model from GitHub into the Colab notebook was another challenge. With no prior experience, I struggled with understanding code and ensuring compatibility, especially with version mismatches.

Hyperparameter tuning for the algorithms was also challenging. Despite extensive experimentation, finding the right balance was not straightforward.

Lastly, the evaluation of image quality was more complicated than initially anticipated. There was a puzzling discrepancy between the numerical results (RMSE) and the visual performance of the algorithms. Although Restormer had the highest RMSE, its denoised images appeared to be of better visual quality compared to the others. This made it difficult to reconcile the numerical results with subjective observations, and it remains an unexplained aspect of the project, highlighting the limitations of RMSE as a sole evaluation metric.

4.3. Key Learnings

This project provided valuable insights into the process of selecting and evaluating image denoising algorithms. I learned how to balance theoretical knowledge with practical experimentation when choosing algorithms. The importance of clear evaluation metrics, such as RMSE and visual quality, became evident as I compared the performance of different denoising methods and analyzed the results.

I also learned how to integrate pretrained models, manage dependencies, and adapt them for specific tasks. Hyperparameter tuning taught me the value of patience and methodical experimentation to optimize algorithms.

Through the process of hyperparameter tuning, I gained hands-on experience in optimizing algorithms for specific tasks. I learned how important it is to be patient and methodical in tuning parameters, as this directly impacts the performance of traditional image processing algorithms.

The project also highlighted the limitations of RMSE as a sole evaluation metric. I learned that while RMSE is useful for measuring pixel-wise accuracy, it does not always align with human visual perception, particularly for deep learning-based models like Restormer. This experience emphasized the need to consider both numerical metrics and subjective evaluations when assessing image quality.

Finally, this project reinforced the importance of systematic experimentation and problem-solving. It taught me how to approach challenges methodically, whether it was integrating a pretrained model, tuning parameters, or handling dataset issues. This experience has significantly enhanced my critical thinking and problem-solving abilities, skills that will be valuable for future projects.

4.4. Future Work

If given more time, I would refine the hyperparameter tuning process, experimenting with different configurations and using multiple evaluation metrics. I also plan to explore additional metrics beyond RMSE and PSNR to determine the most effective denoising algorithm.

References

[1] Arize AI, "Root Mean Square Error (RMSE): What You Need to Know," Arize AI Blog, 2022. [Online]. Available: https://arize.com/blog-course/root-mean-square-error-rmse-what-you-need-to-know/#:~:text=Root%20Mean%20Square%20Error%20(RMSE)%20is%20sometimes%20preferred%20over%20Mean,it%20more%20sensitive%20to%20outliers. [Accessed: Feb. 3, 2025].

[2] Wikipedia, "Total variation denoising," Wikipedia, 2025. [Online]. Available: https://en.wikipedia.org/wiki/Total_variation_denoising. [Accessed: Feb. 3, 2025].

[3] GeeksforGeeks, "Noise Removing Technique in Computer Vision," GeeksforGeeks, 2021. [Online]. Available: https://www.geeksforgeeks.org/noise-removing-technique-in-computer-vision/. [Accessed: Feb. 3, 2025].

[4] Scikit-Image, "Denoising Images," Scikit-Image Examples, 2025. [Online]. Available: https://scikit-image.org/docs/stable/auto_examples/filters/plot_denoise.html. [Accessed: Feb. 3, 2025].

[5] Scikit-Image, "Restoration Module," Scikit-Image Documentation, 2025. [Online]. Available: https://scikit-image.org/docs/stable/api/skimage.restoration.html. [Accessed: Feb. 3, 2025].

[6] Wikipedia, "Bilateral filter," Wikipedia, 2025. [Online]. Available: https://en.wikipedia.org/wiki/Bilateral_filter. [Accessed: Feb. 3, 2025].

[7] Scikit-Image, "Non-Local Means Denoising," Scikit-Image Documentation, 2025. [Online]. Available: https://scikit-image.org/docs/stable/api/skimage.restoration.html#skimage.restoration.denoise_nl_means. [Accessed: Feb. 3, 2025].

[8] PyDocs, "Wavelet Denoising in Skimage," PyDocs, 2025. [Online]. Available: https://pydocs.github.io/p/skimage/0.17.2/api/skimage.restoration._denoise.denoise_wavelet.html. [Accessed: Feb. 3, 2025].

[9] Scikit-Image, "Wavelet Denoising," Scikit-Image Documentation, 2025. [Online]. Available: https://scikit-image.org/docs/stable/api/skimage.restoration.html#skimage.restoration.denoise_wavelet. [Accessed: Feb. 3, 2025].

[10] PyWavelets, "Wavelet Reference," PyWavelets Documentation, 2025. [Online]. Available: https://pywavelets.readthedocs.io/en/latest/ref/wavelets.html. [Accessed: Feb. 3, 2025].

[11] PyWavelets, "Installation Guide," PyWavelets Documentation, 2025. [Online]. Available: https://pywavelets.readthedocs.io/en/latest/install.html. [Accessed: Feb. 3, 2025].

[12] SWZ30, "Restormer: Efficient Transformer for Image Restoration," GitHub Repository, 2025. [Online]. Available: https://github.com/swz30/Restormer. [Accessed: Feb. 3, 2025].

[13] Google Colab, "Restormer Implementation," Google Colab Notebook, 2025. [Online]. Available: https://colab.research.google.com/drive/1C2818h7KnjNv4R1sabe14_AYL7lWhmu6?usp=sharing. [Accessed: Feb. 3, 2025].

[14] Kodak, "Kodak Image Dataset," R0K, 2006. [Online]. Available: https://r0k.us/graphics/kodak/. [Accessed: Feb. 3, 2025].