Jaypee Institute of Information Technology, Noida

MINOR PROJECT
(Project Report)

Comparison of Various Image De-noising Algorithms

Team Members:

Dharmesh Malav - 19803005

Nikhil Paleti - 19803024

Divyanshu Tiwari – 19803025

Supervisor: Dr. Niyati Aggrawal

DECLARATION

We, Nikhil, Divyanshu and Dharmesh, hereby declare the following usage of the open source code and prebuilt libraries in our minor project in 5th Semester with the consent of our supervisor. We also measured the similarity percentage of pre written source code and our source code and the same is mentioned below. This measurement is true with best of our knowledge and abilities.

1. List of pre build libraries
   Numpy, Matplotlib, Scipy, Skimage, Time (?)

2. List of pre build features in libraries or in source code.

Feature	Documentation/Source
bm3d	https://pypi.org/project/bm3d/
numpy	https://numpy.org/doc/
pyplot	https://matplotlib.org/stable/api/pyplot_summary.html
convolve2d	https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.convolve2d.html
structural similarity	https://scikit-image.org/docs/dev/api/skimage.metrics.html
time	https://docs.python.org/3/library/time.html

3. Percentage of pre written source code and source written by us.

   Estimate – 30%

Nikhil Paleti 19803024 Nikhil

Divyanshu Tiwari 19803025 Divyanshu

Dharmesh Malav 19803005 [illegible]

Project Link – https://github.com/NikhilPaleti/Denoise-Comparison-Report

INTRODUCTION

Image de-noising can be described as the problem of mapping from a noisy image to a noise-free image. The best currently available de-noising methods approximate this mapping with cleverly engineered algorithms.

In this work we attempt to apply multiple de-noising algorithms like Median Filter and Blur. While this has been done before, our approach is to optimize the existing algorithms to de-noise effectively and efficiently.

The abstract objective of the project is to compare de-noising algorithms to de-noise a given "noisy" image, with the minimum loss of information from the original image.

The objective of the resulting model is to improve the results and provide better measuring scores than the popular and widely-used, "Golden Standard" for De-Noising Images - BM3D.

PROBLEM STATEMENT

Photography is increasingly becoming a huge part of people's daily lives, thanks to smartphones, and photography has become a new way to communicate because of Social Media like Facebook, Instagram, Snapchat and more, which facilitate communication through photos.

And similarly, photography does come with challenges, one of them being the noise that is captured by images, that doesn't really exist, due to the limitations in the hardware of the camera modules. Noise is also "boosted" at night time photography, as a trade-off to capture more light.

So it is quite critical to develop, and figure out an appropriate algorithm to reduce noise after capturing an image, to provide clear, yet bright images under all conditions. However, we must simultaneously make sure the algorithm is quick, so as to not tax the system resources all the time, and to be able to allow users to snap as many images as they need.

LITERATURE REVIEW

1. BM3D
   Block-matching and 3D filtering algorithm (BM3D) is a popular algorithm which is predominantly used for image de-noising. This algorithm has a high capacity to achieve better noise removal results as compared with other existing algorithms at the time.
   Nevertheless, there is still much room for improvement in this algorithm to achieve more attractive results.

   BM3D consists of 2 main “steps” underneath:
   - Hard Thresholding/Block-Matching, which scans the image for repeating, similar “blocks” of pixels, which fall under a certain threshold.
   - Weiner Filtering. The blocks of images which are matched, are run through the Weiner Filter Algorithm, to de-noise. The Weiner Filter is a linear mapping, estimation de-noising function, which aims to reduce MSE of the given image.

2. MEDIAN FILTER
   The Median Filter algorithm is a fairly rudimentary yet, highly effective algorithm. The core principle of the Median Filter Algorithm is to find the “median” value of a given array/matrix, and replace the elements with the median value. This effectively allows Median Filter to work as a “Band Pass Filter”, cutting off low and high signals.

3. CONVOLUTION BLUR
   The blur function has been coded, to blur the image and reduce noise.
   Blur is a common utility to reduce blur of an image, since it effectively acts as a Low-Pass Filter, cutting-off extremely dark inputs (pixels).
   However, undoubtedly, this will result in a clear loss of clarity of the image, since it will of course, blur the image.

   In our given function, the given kernel/window size of NxN from the image is “convolved” with a unit-matrix of size N, to “blur” it and also reduce noise.

4. PSNR
   PSNR, or “Peak Signal-to-Noise Ratio” is the normalized Mean Squared Error of a given image. PSNR returns a value between 0 and 100, where a higher value represents a better signal integrity of the image, indicating lesser noise.
   Mean Squared Error, is the average of the square of the differences of values in each pixel of an image, compared to the processed version.
   Definition - PSNR is the expression for the ratio between the maximum possible

value (power) of a signal and the power of distorting noise that affects the quality of its representation

MSE = $\frac{1}{mn}\Sigma\Sigma[\mathrm{y}-\mathrm{x}]^2$

PSNR = $10 * log10\left(\frac{225^2}{MSE}\right)$

4. SSIM

   SSIM, or "Structural Similarity Index Measure", is a full reference metric, which measures the quality of an image, by making a comprehensive comparison of the entire image, encompassing attributes like Luminance, Contrast and of course, structure. SSIM returns a value from 0 to 1, higher the better.

   However, unlike PSNR, in case of SSIM, each pixel is not processed individually and groups of pixels are processed/computed at once, to compare the "essence" of an image.

   SSIM = $\left[l(x,y)^{\alpha}.c(x,y)^{\beta}.s(x,y)^{\gamma}\right]$

5. SALT n' PEPPER NOISE

   Salt n' Pepper Noise, is an occurrence in digital photography, when there might be a "dead", "stuck" or a blocked pixel in the camera sensor, or highly unstable power supply to the Silicon. It is also common on old Film Style cameras, because of the physical layout of the crystal structure.

   Salt n' Pepper noise, as the name suggests, is a distribution of Salt (1s) and Pepper (0s), on the dish (image).

6. GAUSSIAN NOISE

   Gaussian Noise is one of the most common forms of noise encountered during photography and even other applications like audio, digital circuitry and more. Gaussian Noise is the noise that is produced at source of capturing images or audio, in form of disrupted power supply or caused by poor lighting in case of photography too.

   Gaussian Noise is a simple, randomly and evenly distributed noise, within bounds of a specified "sigma" value, which describes the probability distribution, as to how much noise is meant to be added.

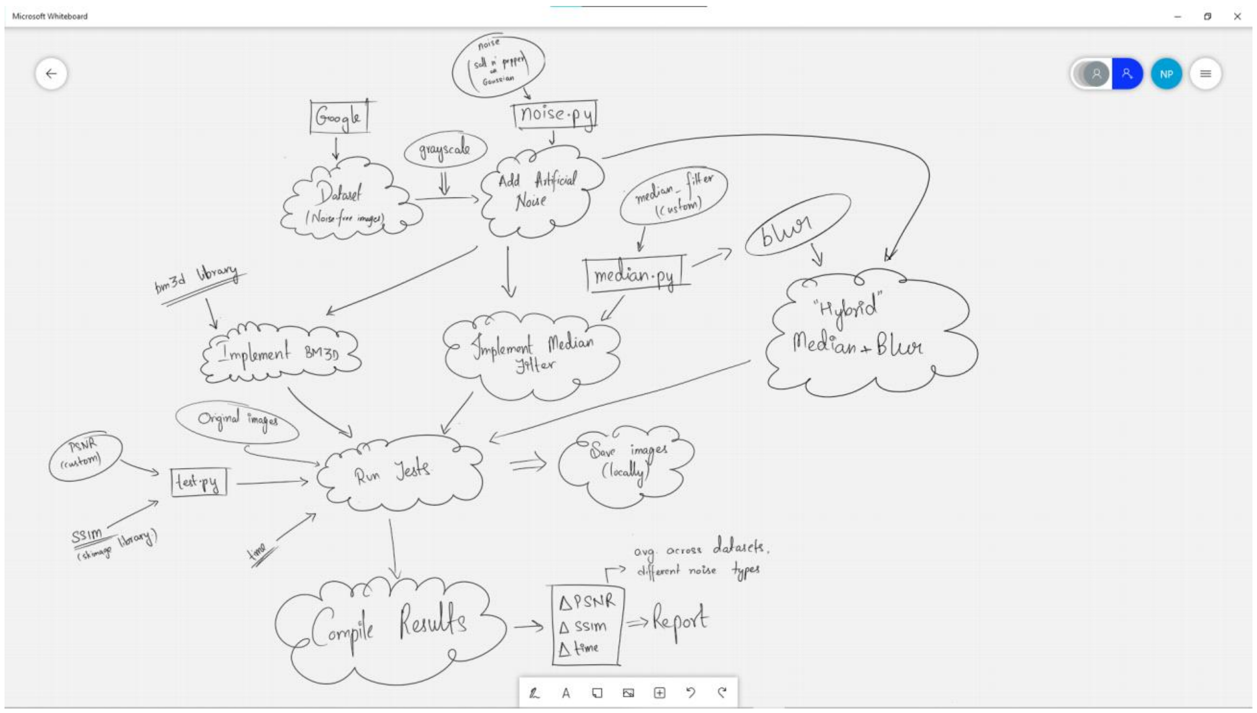
EXPERIMENTAL DESIGN

The process of the project can be broken down into the following steps:

1. Data Processing
    - Crop images to 999x999 (or lesser)
    - Import images to Python Code, in arrays
    - Convert images to grayscale
    - Artificially add noise

2. Create Median Filter and Blur function for de-noising evaluation.
3. A BM3D function is also developed for the same purpose.
4. Finally their results are compared.

WORKFLOW DIAGRAM

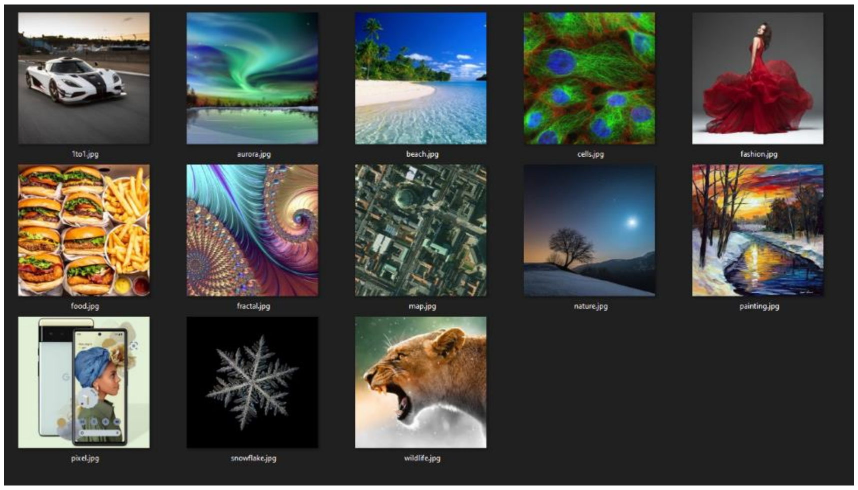


DATASET

The “dataset” is a comprehensive list of manually picked images from the web, which are of very high resolution, and have minimal original noise, to aide with proper testing.

A wide range of subjects were chosen, varying from wildlife to fashion to cells, with varying overall structures, be it geometric structured objects, objects with organic origin and more, so as to also confirm the consistency of each algorithm, across a wide variety of subjects.

Each downloaded image is then cropped to a 1:1 aspect ratio (if required), then reduced to 999x999 pixels in size, or lesser (if required). This was done to further improve consistency between tests/results, and also to be able to have a sizeable comparison *(since all algorithms would be incredibly fast with smaller resolutions)*.
All images used were of “.jpg” format.



RESEARCH/TESTING METHODOLOGY

Data Preprocessing

1. The images sizes are fixed to ~999x999 and saved locally
2. The images are imported to the python file and stored in lists.
3. The images are converted from RGB to Grayscale.
4. The reference data is formed by adding noise at various sigma values, using various noise-adding techniques, to original images.
5. Now, for the reference data, we introduce noise at different “rates” (sigma values), with different filters, as described above
6. Before Pre-processing After Pre-processing

Artificial Noise-Adding Techniques

- Salt n' Pepper Noise: In our implementation, a function was assigned the task to assign 1s and 0s to on random, within range of assigned sigma value.
- Gaussian Noise: Our implementation of the Gaussian Noise has been as the definition suggest, a randomly, evenly distributed noise, within specified sigma values.

De-noising Methods

- BM3D: The highly popular, Golden Standard of image de-noising, was used as a reference, as the target to improve upon.
  An Open-Source BM3D Library was utilized to apply BM3D Filtering.
- Median Filter: Our implementation of the Median Filter applies a sliding window algorithm, to move a window of size NxN, through the image. The median value of all the pixels in range of the window is computed, and then is over-written onto the data of the central pixel of the window.
- "Hybrid Filter": An interesting, custom application using and combining multiple algorithms, to achieve results better than BM3D, but with trade-offs compared to our Median Filter.
  Our "Hybrid" Algorithm is a simple blend of Median Filter and Convolution Blur, however, with lower "Kernel" sizes, to reduce performance impact.

Performance Measures

1. The final results are based on the levels on PSNR and SSIM in the final output images, and the time taken to de-noise the images too.
2. A comparative report is done on the basis of the above measures, between BM3D, Median Filter and a custom hybrid algorithm.

- PSNR, as the name suggests, measures the Peak Signal to Noise Ratio (Higher the better), and gives a mathematical idea of how much of the signal has been degraded.
  Our implementation is applying the mathematics of the PSNR formula into a function
- SSIM meanwhile, gives a "score" between 0 and 1, as to how similar the given 2nd image is, compared to the primary image. Higher the better in this case too.
  Our implementation is using the skimage library to implement the measure.

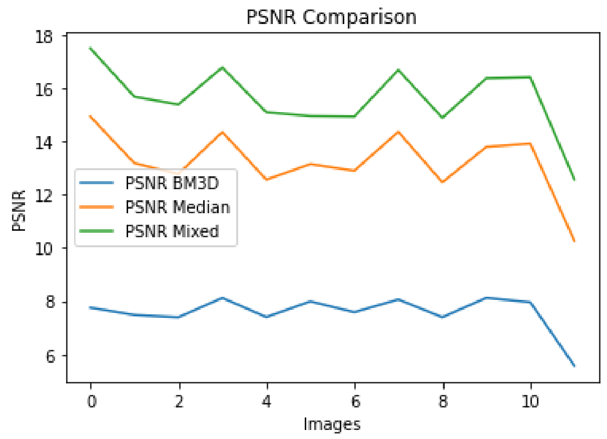
In short, PSNR is a guide for better noise removal, and SSIM for feature preservation.

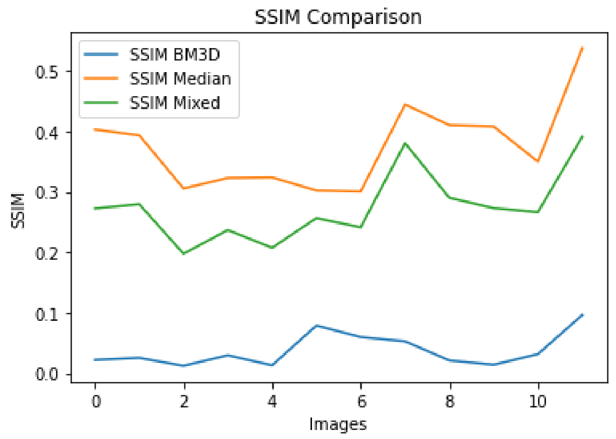
- Time, of course, lower the better, and is simply measured by measuring time before and after executing the code, and then the difference of the two is the time taken.

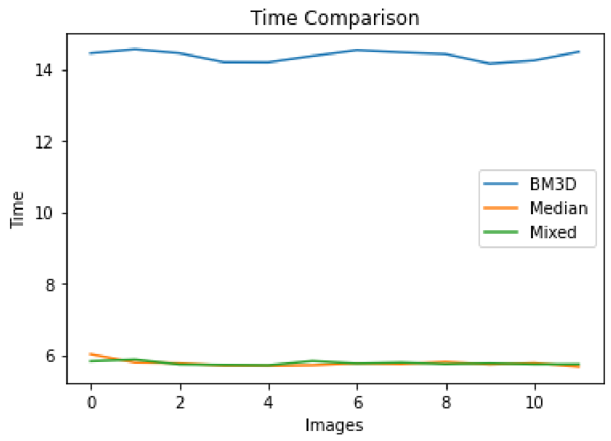
-

RESULTS

Performance of each filter, plotted for – Salt n' Pepper Noise (Sigma – 0.3) // High Noise

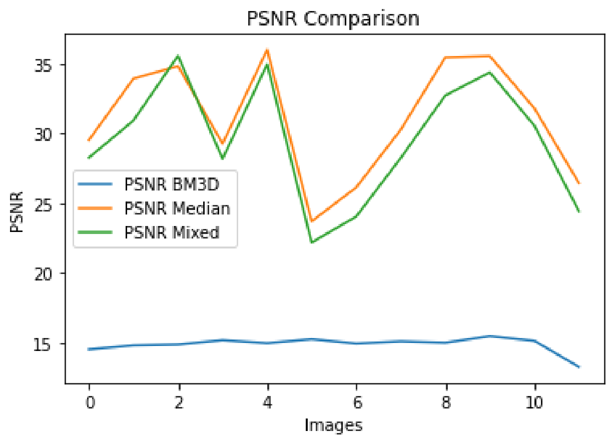


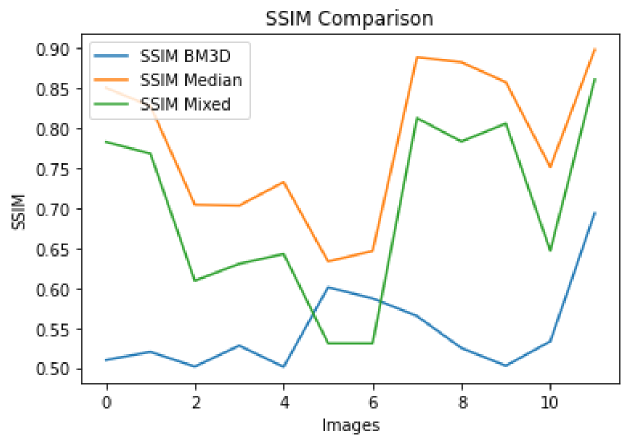


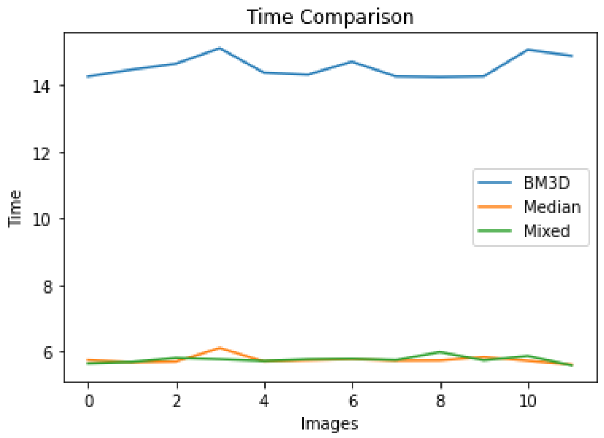


BM3D PSNR	Median PSNR	Mixed PSNR	BM3D SSIM	Median SSIM	Mixed SSIM
7.7403	14.8416	17.3852	0.0221	0.4038	0.2731
7.497	13.2242	15.7364	0.0259	0.3932	0.2798
7.3819	12.729	15.3323	0.0124	0.304	0.1956
8.0978	14.3093	16.7385	0.0298	0.3209	0.2342
7.3878	12.5022	15.0161	0.013	0.3223	0.2078
7.9739	13.1194	14.9322	0.0779	0.3016	0.2559
7.5802	12.8346	14.8471	0.0615	0.3005	0.2404
8.0589	14.3151	16.6331	0.0531	0.4442	0.3802
7.3857	12.4634	14.8814	0.0219	0.4092	0.2896
8.1116	13.7541	16.3363	0.0147	0.4065	0.2716
7.9509	13.8895	16.3589	0.0312	0.3496	0.2644
5.5666	10.2743	12.5758	0.0971	0.5393	0.3946
AVERAGE	AVERAGE	AVERAGE	AVERAGE	AVERAGE	AVERAGE
7.56105	13.1880583	15.564442	0.038383333	0.374591667	0.27393333

Performance of each filter, plotted for – Salt n' Pepper Noise (Sigma – 0.05) //Low Noise



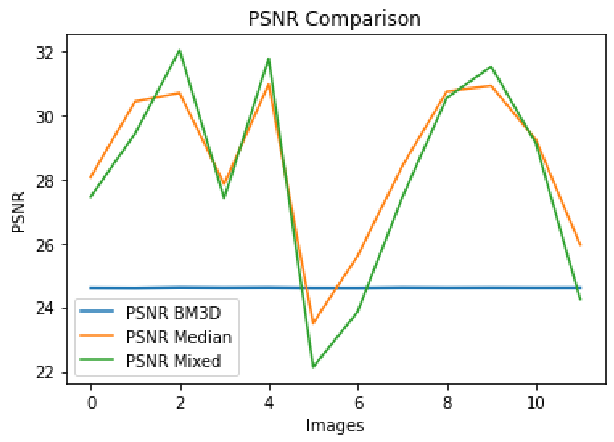


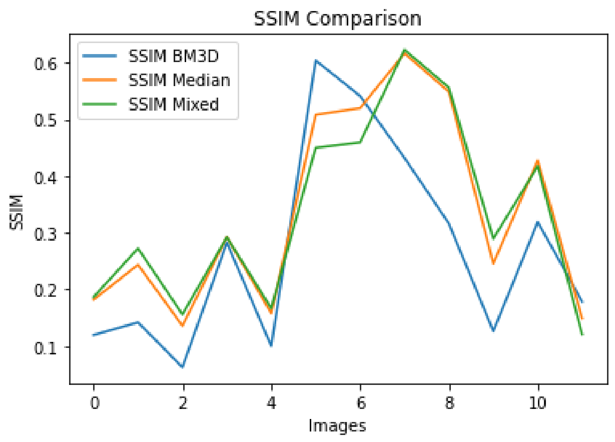


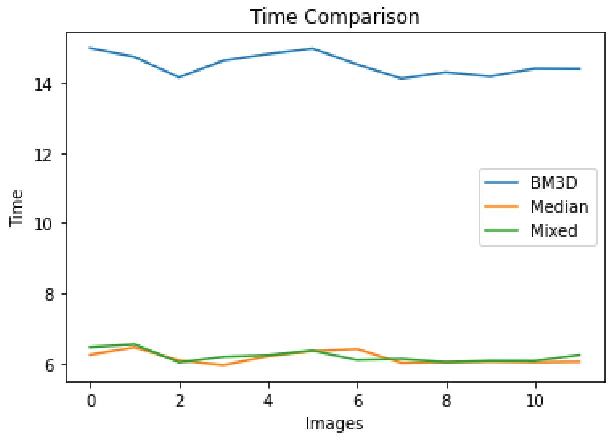
BM3D PSNR	Median PSNR	Mixed PSNR	BM3D SSIM	Median SSIM	Mixed SSIM
11.7325	27.2237	27.3147	0.5102	0.8492	0.7817
11.893	29.77	29.5682	0.5193	0.8264	0.7675
11.9227	30.1675	33.0163	0.499	0.7033	0.6086
12.3052	27.2996	27.414	0.5283	0.7019	0.6296
12.0203	30.889	32.7322	0.5036	0.7318	0.6424
12.3574	22.6293	21.7675	0.6013	0.6328	0.5307
12.0185	24.4788	23.4769	0.5859	0.6466	0.5303
12.254	27.8505	27.383	0.5648	0.8883	0.8118
12.0228	30.5663	31.0656	0.5255	0.8822	0.7828
12.5487	31.2284	32.6027	0.5022	0.8565	0.8052
12.2603	29.0163	29.4824	0.5313	0.7505	0.6465
10.311	24.4432	23.762	0.696	0.8973	0.8604
AVERAGE	AVERAGE	AVERAGE	AVERAGE	AVERAGE	AVERAGE
11.9705333	27.96355	28.298792	0.547283333	0.780566667	0.69979167

We can observe excellent performance of both our "custom" algorithms with varying levels of Salt n' Pepper – Much higher PSNR and SSIM, twice as fast.

Performance of each filter, plotted for – Gaussian Noise (Sigma – 0.15)



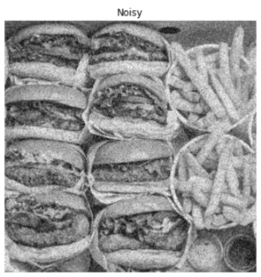


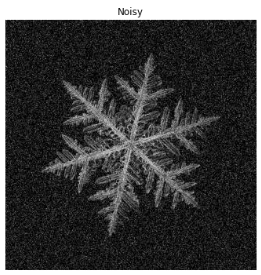


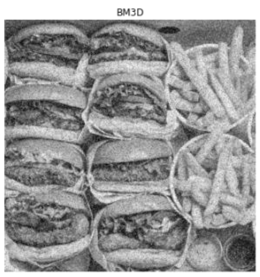
BM3D PSNR	Median PSNR	Mixed PSNR	BM3D SSIM	Median SSIM	Mixed SSIM
24.608	28.0637	27.453	0.1197	0.1825	0.1869
24.6072	30.4375	29.4257	0.1418	0.2428	0.2708
24.6083	30.7062	32.0333	0.0627	0.1358	0.1552
24.6113	27.8502	27.4029	0.2819	0.2915	0.2917
24.5976	30.9461	31.7309	0.1002	0.1575	0.166
24.6063	23.5007	22.1245	0.604	0.5073	0.4489
24.6234	25.6231	23.8663	0.5416	0.5199	0.4589
24.6143	28.4096	27.4255	0.4313	0.6156	0.6208
24.6068	30.7341	30.5366	0.3169	0.5489	0.5564
24.616	30.9082	31.5244	0.1267	0.2453	0.2896
24.6076	29.2337	29.1401	0.3192	0.4283	0.4187
24.6051	25.9626	24.2437	0.1784	0.149	0.1208
AVERAGE	AVERAGE	AVERAGE	AVERAGE	AVERAGE	AVERAGE
24.609325	28.5313083	28.075575	0.2687	0.33536667	0.3320583

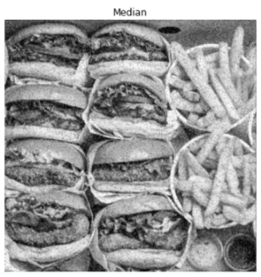
In case of Gaussian Noise however, as expected, BM3D performs better, coming much closer to our algorithms

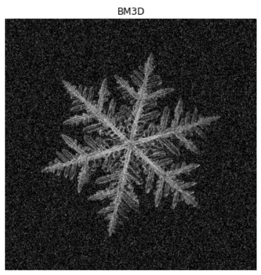
A look at the some of the images as processed:

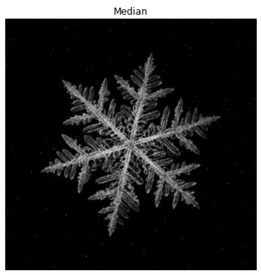


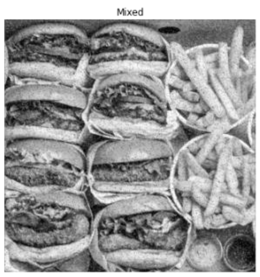


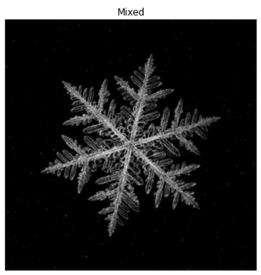












CONCLUSION

Our conclusions from the results are as follows:

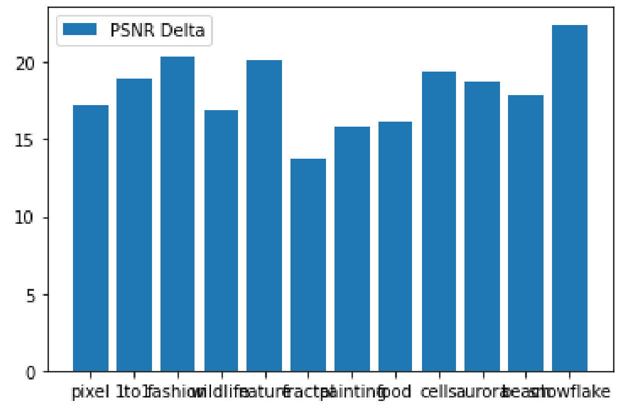
BM3D managed highly underwhelming results, in contrast to either of our algorithms, be it under Low or High Noise with Salt n' Pepper or with Gaussian Noise, even though the Gaussian Noise performance was much closer.

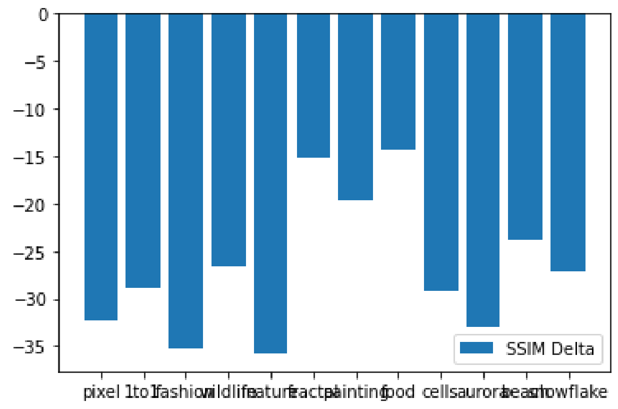
Either of our algorithms, Median Filter or "Hybrid", managed to be able to execute in just 6-seconds on average, across all the images, all the noise types, which was more than twice (2X) as fast as BM3D, which took upto 14-seconds for the results.
But BM3D, as seen in above results, managed to be significantly more consistent, returning "flatter" graphs and performing much closer to our algorithms in case of Gaussian Noise

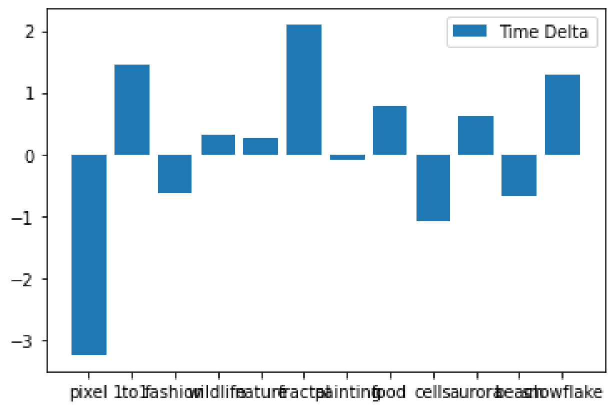
As interesting as it would've been, the meaningful conclusions left to be drawn, are between our own algorithms, the Hybrid and the Median Filter.

Below are the graphs, representing percentage improvement or deterioration of our "Hybrid Algorithm", compared to plain Median Filter, per-image, for Salt n' Pepper Noise at Sigma = 0.1.

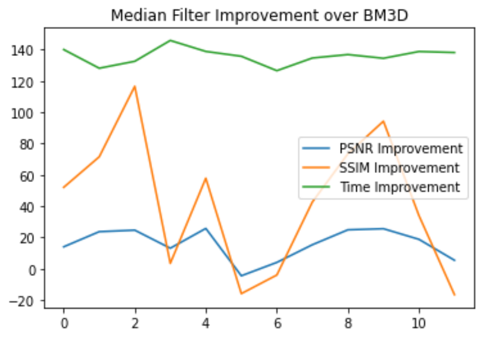
As seen, the Hybrid Algorithm manages better PSNR results, by 15% on average, while also posting worse SSIM results, of an average around 25%, which was the expected, hypothesized "trade-off" which had to be made, using a blur function.

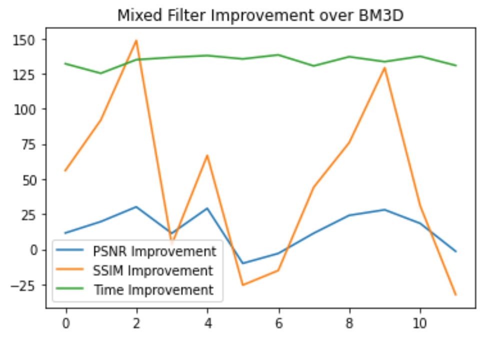






A brief look into BM3D's "impressing" Gaussian-Noise Performance. Graphs of improvement or deterioration of BM3D with Gaussian Noise, compared to our filters





REFERENCES

1. A. Horé and D. Ziou, "Image Quality Metrics: PSNR vs. SSIM," 2010 20th International Conference on Pattern Recognition, 2010, pp. 2366-2369, doi: 10.1109/ICPR.2010.579.
2. Xin, J., & Esser, J. E. Filtering and convolutions. *iCamp-UCI Interdisciplinary Computational and Applied Mathematics Program, Math 77A Lecture*.
3. Huang, T., Yang, G. J. T. G. Y., & Tang, G. (1979). A fast two-dimensional median filtering algorithm. *IEEE transactions on acoustics, speech, and signal processing*, *27*(1), 13-18.
4. Lebrun, M. (2012). An analysis and implementation of the BM3D image denoising method. *Image Processing On Line*, *2012*, 175-213.3
5. https://webpages.tuni.fi/foi/GCF-BM3D/
6. Grayscale Conversion - Image Processing 101 Chapter 1.3: Color Space Conversion (dynamsoft.com)
7. https://en.wikipedia.org/wiki/Median_filter