Jaypee Institute of Information Technology, Noida



MINOR PROJECT   
(Project Report)   
  
BM3D – The best?

Team Members:   
Dharmesh Malav - 19803005  
Nikhil Paleti - 19803024  
Divyanshu Tiwari – 19803025

Supervisor: Dr. Niyati Aggrawal

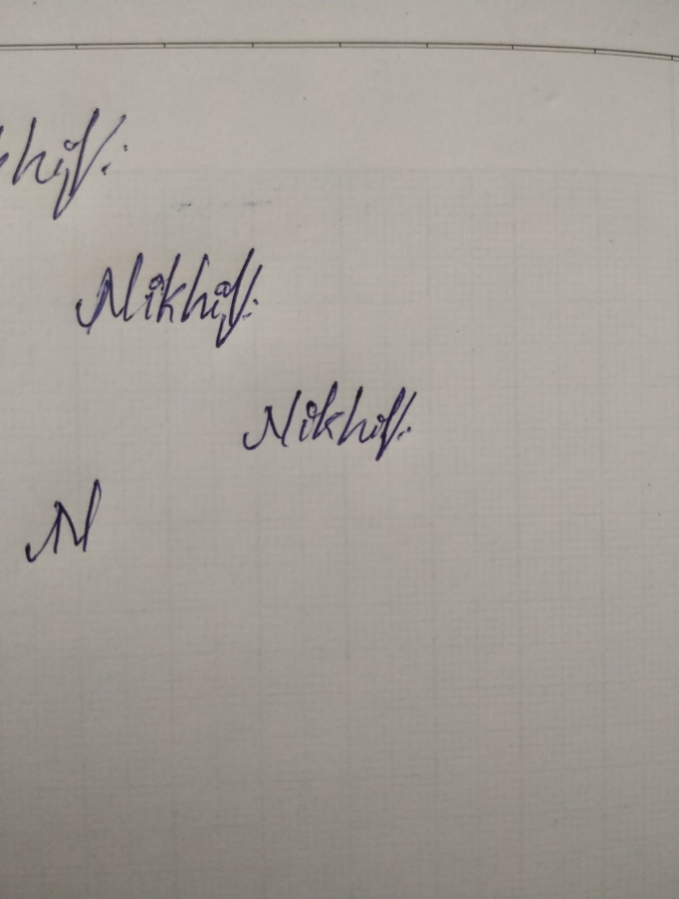
DECLARATION

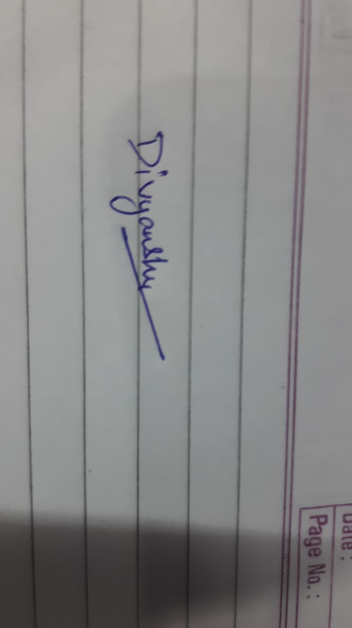
We, Nikhil, Divyanshu and Dharmesh, hereby declare the following usage of the open source code and prebuilt libraries in our minor project in 5th Semester with the consent of our supervisor. We also measured the similarity percentage of pre written source code and our source code and the same is mentioned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

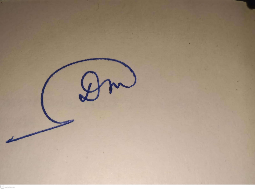
|  |  |
| --- | --- |
| Feature | Documentation/Source |
| bm3d | <https://pypi.org/project/bm3d/> |
| Npmpy | <https://numpy.org/doc/> |
| Pyplot | <https://matplotlib.org/stable/api/pyplot_summary.html> |
| Convolve2d | <https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.convolve2d.html> |
| Structural similarity | <https://scikit-image.org/docs/dev/api/skimage.metrics.html> |
| time | <https://docs.python.org/3/library/time.html> |

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



Nikhil Paleti 19803024

Divyanshu Tiwari 19803026

  
  
Dharmesh Malav 19803005

Declaration by Supervisor (To be filled by Supervisor only)

I, ........................................(Name of Supervisor) declares that I above submitted project with Titled ..................................................................... was conducted in my supervision. The project is original and neither the project was copied from External sources not it was submitted earlier in JIIT. I authenticate this project.

(Any Remarks by Supervisor)

Signature (Supervisor)

ABSTRACT

The abstract objective of the proposed project is to use/create De-Noising Algorithms to De-Noise, and compare their performance with the industry-standard BM3D De-Noising Algorithm.

The main aim of the project is to for 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.

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.

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

2. MEDIAN FILTER

The Median Filter algorithm is a fairly rudimentary yet, highly effective algorithm.

The Median Filter applies a sliding window algorithm, to move a window of size NxN, through the image. Each time the window moves, the median value of all the pixels in range of the window is computed, and then the median value is over-written onto the data of the central pixel of the window.

3. CONVOLUTION BLUR

The blur function has been coded, to blur the image.

The function a given kernel/window size of NxN, which then is “convolved” with a unit-matrix of size N.

4. PSNR

MSE =

PSNR =

5. SSIM

SSIM =

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

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

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 in clear testing.

Each downloaded image is then cropped to a 1:1 aspect ratio (if required), then reduced to 999x999 pixels in size, or lesser.   
All images used were of “.jpg” format.

WORKFLOW DIAGRAM

As the Project flow suggests, the first process executed for each of our models and experiments is the data preparation, adding noise to it creating images to be compared later to the outputs.

Then the data is divided and used in training and testing models and the results are then compared to that of BM3D.

For the complete test dataset, the images are first run through our model and then they are run through the BM3D function that generates the output image as well.

Both these images are then passed through the PSNR and SSIM functions are used to generate scores for each of these images, and finally a mean score for that particular dataset is generated and recorded.

Data Analysis

1. The images are obtained from all over the Internet, with a varying set of subjects, scenarios and more
2. The images are stored in the “dataset” folder

RESEARCH/TESTING METHODOLOGY

Data Preprocessing Techniques

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, at 1%, 10% and 30%
6. The final dataset is then split into train and test for model training. Before Pre-processing After Pre-processing

De-noising Methods

Performance Measures

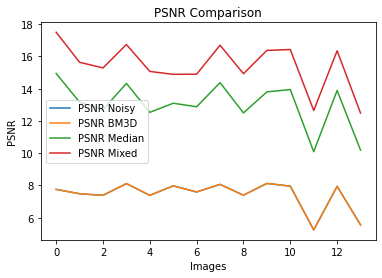
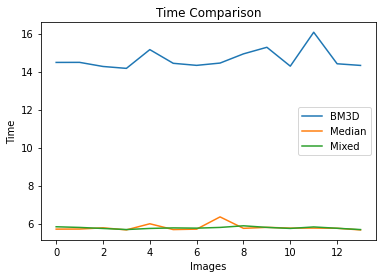
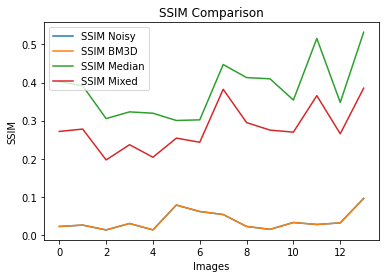
1. The final results are based on the levels on PSNR and SSIM in the final output images, and the time taken to denoise the images too.
2. A comparative report is done on the basis of the above measures, between BM3D, Median Filter and a custom hybrid algorithm.

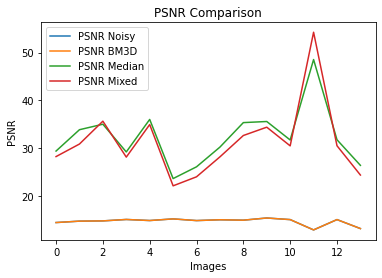
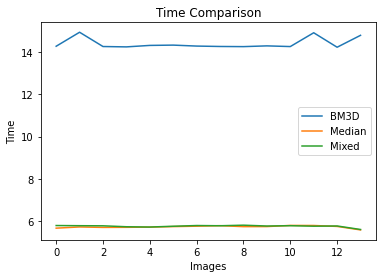
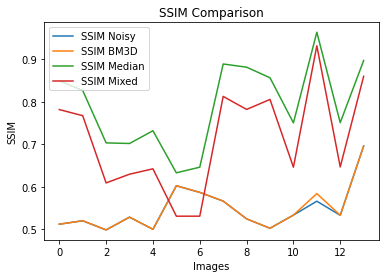
What is PSNR and SSIM

* PSNR, or “Peak Signal-to-Noise Ratio” is the normalized Mean Squared Error of a given image. PSNR returns a value between 0 and 1, 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.
* 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.   
  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.   
  Definition -
* In short, PSNR is a guide for better noise removal, and SSIM for feature preservation.

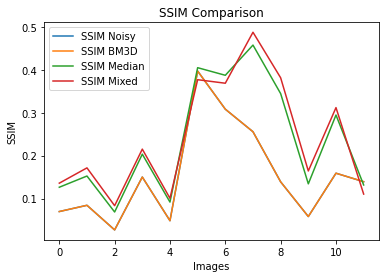
RESULTS

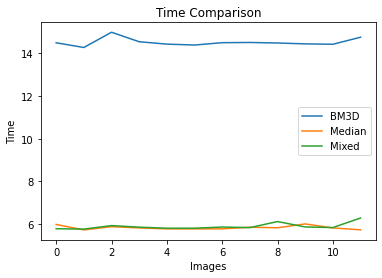
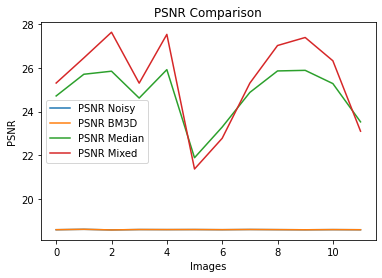
Performance of each filter, plotted for – Salt n’ Pepper Noise (Sigma – 0.3) // High Noise

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

Performance of each filter, plotted for – Gaussian Noise (Sigma – 0.3) //Low Noise





CONCLUSION

prediction

REFERENCES

1. Matplotlib Refs - <https://matplotlib.org/stable/api/pyplot_summary.html>
2. PSNR vs SSIM and implementations- <https://ieeexplore.ieee.org/document/5596999> ‘
3. Grayscale Conversion - [Image Processing 101 Chapter 1.3: Color Space Conversion (dynamsoft.com)](https://www.dynamsoft.com/blog/insights/image-processing/image-processing-101-color-space-conversion/)