Implementation of BM3D Algorithm

Additional Project for "Mathematical Image Processing", SS 2021

Jiahui Li, Zixiang Zhou, Peng Gao
peng.gao@uni-heidelberg.de

October 12, 2023

Abstract

Denoising is a fundamental task in image processing with wide applications for enhancing image qualities. As a recent algorithm, BM3D is considered as an efficient method for image denoising. In this project, we try to implement the BM3D denoising method and compare it to the few algorithms(mean filtering, median filtering, L1-TV and L2-TV). We apply these algorithms to denoise gray-scale and color images with different types of noise (Gaussian and salt-and-pepper) as well as different level of noise. Experiments show that BM3D outperforms other methods, and achieves competitive results. Finally, we analyse the BM3D properties and development in modern image denoising techniques empirically.

1 Introduction

Noises are inevitable artifacts in the process of imaging equipment shooting images or videos. Image denoising as a pre-processing step of image analysis has always been the basic task of image processing and computer vision. There have been tremendous methods for image denoising in the past years, which are mainly divided into three categories: Filtering-Based Methods, Model-Based Methods and Learning-Based Methods. BM3D[1] is the state-of-the-art method in Filtering-Based Methods.

Block-matching and 3D filtering (BM3D) is a 3-D block-matching algorithm used primarily for noise reduction in images. There are two steps in BM3D: a hard-thresholding and a Wiener filter stage, both involving the following parts: grouping, collaborative filtering, and aggregation. This algorithm depends on an augmented representation in the transformation site[2].

BM3D is group-based and is implemented by stacking similar 2D images into 3D arrays. In natural images, there are strong similarities between small image blocks in different spatial positions. This is characteristic of blocks belonging to uniform areas, edges, textures, smooth intensity gradients, etc. Due to the similarity between these grouping blocks, the 3D transformation produces sparse groups. It is well known that sparsity is critical for denoising. Therefore, proper grouping is crucial for BM3D. When modeling natural images, the existence of mutually similar blocks can be considered as a very realistic assumption, which strongly inspires the use of grouping and collaborative filtering for image denoising algorithms. The algorithm is explained in detail in article[1].

In this project, we implement the grayscale BM3D method and extend it to color image denoising. The implementation of the model is based on article [3]. For the image damaged by Gaussian white noise and salt-and-pepper noise, according to the two key steps of BM3D, we design modules with different functions in each step to process the noise image, and modify corresponding parameters according to different noise levels to achieve more effective denoising.

2 Algorithm Description

2.1 Algorithm of BM3D

In this algorithm, hard threshold is first used to estimate the denoised image during the collaborative filtering. The second step is based on the original noise image and the basic estimation obtained in the first step.

This general procedure is implemented in two different forms to compose a two-step algorithm. This algorithm is illustrated in Fig. 1 and proceeds as follows.

Step 1) Basic estimate.

a) *Grouping.* Find blocks that are similar to the

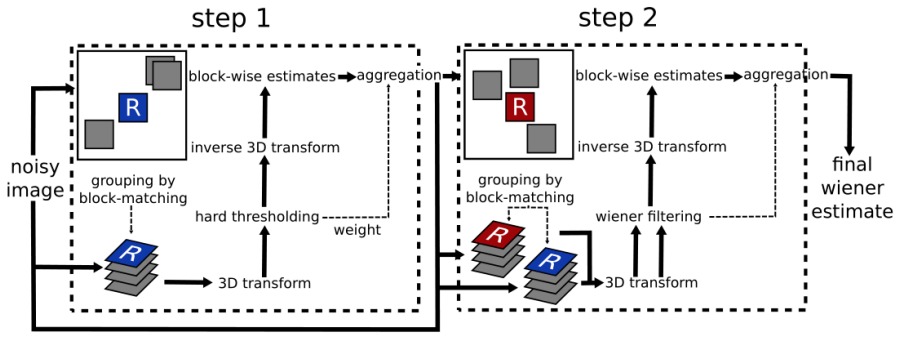

Figure 1: Scheme of the BM3D algorithm[3].

currently processed one and then stack them together in a 3-D array (group).

b) *Collaborative hard-thresholding.* Apply a 3-D transform to the formed group, attenuate the noise by hard-thresholding of the transform coefficients, invert the 3-D transform to produce estimates of all grouped blocks, and return the estimates of the blocks to their original positions.

c) *Aggregation.* Compute the basic estimate of the true-image by weighted averaging all of the obtained estimates that are overlapping.

Step 2) Final estimate: Using the basic estimate, perform improved grouping and collaborative Wiener filtering.

a) *Grouping.* Use BM within the basic estimate to find the locations of the blocks similar to the currently processed one. Using these locations, form two groups (3-D arrays), one from the noisy image and one from the basic estimate.

b) *Collaborative hard-thresholding.* Apply a 3-D transform on both groups. Perform Wiener filtering on the noisy one. Produce estimates of all grouped blocks by applying the inverse 3-D transform on the filtered coefficients and return the estimates of the blocks to their original positions.

c) *Aggregation.* Compute a final estimate of the true-image by aggregating all of the obtained local estimates using a weighted average.

2.2 Observation Model and Notation

We consider a noisy image $z : X \to \mathbb{R}$ of the form

$$u(x) = y(x) + \eta(x), \quad x \in X \tag{1}$$

where x is a 2-D spatial coordinate that belongs to the image domain $X \subset \mathbb{Z}^2$, y is the true image, and η is i.i.d. zero-mean Gaussian noise with variance $\sigma^2, \eta \sim \mathcal{N}\left(0, \sigma^2\right)$. We denote by P the reference current block whose size is $k \times k$ extracted from u. In order to distinguish between parameters used in the first and in the second step, we respectively use the superscripts "hard" (hard-thresholding) and "wiener" (Wiener filtering). For example, k^{hard} is the block size used in Step 1. Analogously, we denote the basic estimate with u^{basic} and the final estimate with u^{final}.

The following subsections present in detail the steps of the proposed denoising method.

A. Steps 1a and 1b: Grouping and Collaborative Hard-Thresholding.

The original noisy image u is searched in a P-centered $n^{\text{hard}} \times n^{\text{hard}}$ neighborhood for blocks Q similar to the reference block P. The set of similar blocks is simply defined by

$$\mathcal{P}(P) = \left\{ Q : d(P, Q) \leq \tau^{\text{hard}} \right\} \tag{2}$$

Only blocks whose distance with respect to the reference one is smaller than a fixed threshold τ^{hard} are considered similar and grouped. If the true-image y would be available, the block distance could be calculated as

$$d(P, Q) = \frac{\|\gamma'(P) - \gamma'(Q)\|_2^2}{\left(k^{\text{hard}}\right)^2} \tag{3}$$

where γ' is a hard-thresholding operator with threshold $\lambda_{2D}^{\text{hard}}\, \sigma$.

The collaborative filtering of $\mathcal{P}(P)$ is realized by hard-thresholding in 3-D transform domain. The adopted normalized 3-D linear transform, denoted τ_{3D}^{hard}, is expected to take advantage of the two types of correlation and attain good sparsity for the true signal

group. This allows for effective noise attenuation by hard-thresholding, followed by inverse transform that yields a 3-D array of block-wise estimates

$$\mathbb{P}^{\text{hard}}(P) = \tau_{3D}^{\text{hard}\ -1}\left(\gamma\left(\tau_{3D}^{\text{hard}}\left(\mathbb{P}(P)\right)\right)\right) \qquad (4)$$

where γ is a hard-threshold operator with threshold $\lambda_{3\text{D}}\sigma$. For practical purposes, the 3D transform τ_{3D}^{hard} of the 3D group $\mathcal{P}(P)$ is made up of two transforms: a 2D transform denoted by $\tau_{2D}^{\text{hard}}{}_{3p}$ applied on each patch of $\mathcal{P}(P)$, and a 1D transform denoted by τ_{1D}^{hard} applied along the third dimension of the 3D group.

B. Steps 2a and 2b: Grouping and Collaborative Wiener Filtering.

Given the basic estimate u^{basic} of the true image obtained in Step 1c, the denoising can be improved by performing grouping within this basic estimate and collaborative empirical Wienerfiltering.

The block-matching is only processed on the basic estimate. When a set of similar blocks

$$\mathcal{P}^{\text{basic}}(P) = \left\{Q : d(P,Q) \leq \tau^{\text{wien}}\right\} \qquad (5)$$

has been obtained two 3D groups are formed:

- $\mathbb{P}^{\text{basic}}(P)$ by stacking up patches together from the basic estimation u^{basic} and;
- $\mathbb{P}(P)$ by stacking up patches in the same order together from the original noisy image u.

Once again, for optimization a maximum number of N^{wien} patches is kept in the two 3D groups. They have been chosen exactly as described in the first denoising step.

We define the empirical Wiener shrinkage coefficients from the energy of the 3-D transform coefficients of the basic estimate group as

$$\omega_P(\xi) = \frac{\left|\tau_{3D}^{\text{wien}}\left(\mathbb{P}^{\text{basic}}(P)\right)(\xi)\right|^2}{\left|\tau_{3D}^{\text{wien}}\left(\mathbb{P}^{\text{basic}}(P)\right)(\xi)\right|^2 + \sigma^2} \qquad (6)$$

The Wiener collaborative filtering is realized as the element-by-element multiplication of the 3D transform of the noisy image with the Wiener coefficients ω_P. Through this sub-step an estimate of the 3D group is obtained as

$$\mathbb{P}^{\text{wien}}(P) = \tau_{3D}^{\text{wien}\ -1}\left(\omega_P \cdot \tau_{3D}^{\text{wien}}\left(\mathbb{P}(P)\right)\right). \qquad (7)$$

C. Steps 1c and 2c: Global Estimate by Aggregation.

To compute the basic and the final estimates of the true-image in Steps 1 c and 2 c, respectively, we aggregate the corresponding estimates $\mathbb{P}^{\text{hard}}$ and $\mathbb{P}^{\text{wien}}$. This aggregation is performed by a weighted averaging at those pixel positions where there are overlapping block-wise estimates.

1) Aggregation Weights: In general, block estimates are statistically correlated, biased, and have different variances for each pixel. However, it is quite difficult to take into account all these effects. A satisfactory aggregation weight selection is a weight inversely proportional to the total sample variance of the corresponding block direction estimate. That is, noisy block estimators should be given a smaller weight. If the additive noise in the groups u^{hard} and u^{wien} is independent, the total sample variance in the corresponding groups of estimates (4) and (7) is respectively equal to N_P^{hard} and $\|\omega_P\|_2^2$, where N_P^{hard} is the number of retained (non-zero) coefficients in the 3D block after hard-thresholding $\gamma\left(\tau_{3D}^{\text{hard}}\left(\mathbb{P}(P)\right)\right)$ and ω_P are the Wiener filter coefficients (6).

Based on this, in Step 1 c, we assign the weight

$$w_P^{\text{hard}} = \begin{cases} \left(N_P^{\text{hard}}\right)^{-1} & \text{if } N_P^{\text{hard}} \geq 1 \\ 1 & \text{otherwise} \end{cases} \qquad (8)$$

for the group of estimates $\mathbb{P}^{\text{hard}}$ Similarly, in Step 2 c, we assign the weight

$$w_P^{\text{wien}} = \|\omega_P\|_2^{-2} \qquad (9)$$

for the group of estimates $\mathbb{P}^{\text{wien}}$.

2) Aggregation by Weighted Average: The global basic estimate u^{basic} is computed by a weighted average of the block-wise estimates $\mathbb{P}^{\text{hard}}$ obtained in Step 1b, using the weights w_P^{hard} defined in (8), 1.e.,

$$u^{\text{basic}}(x) = \frac{\sum_P w_P^{\text{hard}} \sum_{Q\in\mathcal{P}(P)} \chi_Q(x) u_{Q,P}^{\text{hard}}(x)}{\sum_P w_P^{\text{hard}} \sum_{Q\in\mathcal{P}(P)} \chi_Q(x)} \qquad (10)$$

where $\chi_Q(x) = 1$ if and only if $x \in Q, 0$ otherwise.

The global final estimate u^{final} is computed by (10), where w_P^{hard} and $u_{Q,P}^{\text{hard}}$ are replaced respectively by w_P^{wien} and $u_{Q,P}^{\text{wien}}$.

3 Program Implementation

We implement our BM3D programs for both gray image and colorful image, which are based on the algo-

rithm described above. All numerical transitions of images are completed with the help of **openCV** packages via python.

3.1 Gray Image Version

For gray images, our program commonly follows the two steps of BM3D model. In both two steps, in order to prevent search windows and patches from out of image range, we use functions *get_search_window()* and *check_boundary()* which compute coordinates and control boundaries.

In the first step, we simply use loops to deal with each patch after initializing relative image arrays with appropriate sizes. During searching in each neighbourhood of present patch, *get_similar_patches()* is called to find the most similar N^{hard} patches. It computes the Discrete Cosine Transform (DCT) of 2D array of local image for threshold filtering and return positions, number and DCT values of these best similar patches. Then we do collaborative filtering of the 3D image achieved in the previous step with *filtering_3D()*, which requires an inverse DCT of similar patches to fill up the basically denoised image arrays. Finally we do aggregation through multiplying by Kasier weights.

In the second step, patches matching is applied again on the basic estimate, which returns DCT values of best similar patches of basic estimate image and noisy image with same positions. In the end, the final denoised image is obtained after collaborative filtering and aggregation with Wiener coefficients. The run time of gray image version program is about 2 minutes 20 seconds with CPU Intel(R) Core(TM) i5-8257U, 4 cores, clock rate of 1.4 GHz (3.9GHz in turbo mode).

3.2 Colorful Image Version

The program of colorful image version is developed on the gray image version. We choose YUV transform achieved by inside attribute *cv.COLOR_RGB2YUV* of openCV and apply it to all 3 RGB channels of image at the beginning. In the first step, we reuse the function *get_similar_patches()*, while only pass image data of Y channel to get information of best similar patches. Then we use the position of these best similar patches computed via Y channel in order to achieve DCT values of similar patches in U and V channel. Next, collaborative filtering and aggregation are applied to Y,U,V channels separately. Finally, we repeat the above operations in the second step of gray image program and return the denoised image by the inverse transformation *cv.COLOR_YUV2RGB* in openCV. The total run time is around 5 minutes which is about 2.2 times of that in gray image version.

3.3 Extra Tools Called for Comparison

For comparison, we use mean and median filtering methods encountered in the tutorial as well as the L2-TV and L1-TV variational models from the lecture that we solved by the Chambolle-Pock algorithm. For gray image, we import *proximal* to solve L2-TV and L1-TV variational models. For color image, we import *denoise_tv_chambolle* from *skimage.restoration* to solve L1-TV models; we also call the latest version packages of *BM3D* model from Tampere with love and find that it can output result in seconds.

4 Image Denoising Experiments

In the experiment, white Gaussian noise and salt-and-pepper noise are added to the noiseless image. We compare our BM3D model with some denoising methods, including mean filtering, median filtering, L1-TV and L2-TV. Denoising results (in peak signal to noise ratio (PSNR)) for different noise levels of these methods as well as results of our method are shown in Table 1,2 and 3. By comparing the PSNR of different denoised images, BM3D has the best performance in most experiments. Here we show how to compute PSNR:

The Root Mean Square Error (RMSE) between the reference image (noiseless) u_R and the denoised image u_D is computed as:

$$RMSE = \sqrt{\frac{\sum_{x \in X} \left(u_R(x) - u_D(x)\right)^2}{|X|}}. \tag{11}$$

The Peak Signal to Noise Ratio (PSNR) is evaluated in decibels (dB):

$$PSNR = 20 \log_{10} \left(\frac{255}{RMSE}\right) \tag{12}$$

4.1 Grey Level Images

For grayscale images, all denoising results for different noise levels of these methods are shown in Table 1, 2.

BM3D achieves good results for salt-and-pepper noise images with higher noise levels, while L1-TV performs better for images with lower levels. Fig. 2 shows

the results for the highest noise levels. For Gaussian noise, BM3D achieves the best results. Fig. 3 shows the results obtained by each method when $\sigma = 40$.

Table 1: Denoising results(PSNR) of grayscale images with salt and pepper noise

Method	SP amount		
	0.02	0.1	0.5
Median	22.10	21.98	20.14
Mean	21.25	20.74	17.48
L1-TV	**27.29**	**26.45**	22.39
L2-TV	21.55	20.92	17.59
BM3D_Step1	23.96	22.70	13.32
BM3D_Step2	25.14	24.72	**26.04**

Table 2: Denoising results(PSNR) of grayscale images with Gaussian noise

Method	Gaussian σ		
	10	40	100
Median	23.79	22.00	18.23
Mean	22.88	22.10	19.49
L1-TV	24.81	22.18	19.88
L2-TV	28.27	22.37	13.42
BM3D_Step1	26.23	23.33	19.25
BM3D_Step2	**32.23**	**26.23**	**22.86**

4.2 Color Images

We also apply our model to denoise color images. In our experiment, we use images with Gaussian noise ($\sigma = 40$) and salt-and-pepper noise(SP amount 0.1) as experimental objects respectively. Compared with other methods, official BM3D_3.0.9 achieved the best results on color images and our BM3D achieved the second-best results. We show the PSNR of these results in Fig. 4, 5 and Table 3.

Table 3: Color image denoising results (PSNR)

Method	SP amount	Gaussian σ
	0.1	40
Median	21.68	20.88
Mean	20.27	20.58
L1-TV	22.17	23.87
BM3D_3.0.9	**23.32**	**25.82**
BM3D_own	22.68	23.35

5 Empirical Analysis

We are curious about the reason why BM3D performs effectively in image denoising.

In fact, Lebrun[3] studies the influence of the parameters in BM3D on the effect of denoising, giving the advice on final chosen values for all parameters for different levels of Gaussian noise. In spite of the fact that theoretically the weighting would be significant, practically it is not, as shown in the study on the influence of parameters. However, it is still possible to make it more efficient by using a weighting contingent on the standard deviation of the 3D group estimated in the second step. The ideal Wiener filter study proves that it would be possible to get even better results by taking a more judicious first step, which is even good for the second step treatment. Any other denoising result could be seen as a basic estimate.

The goal of natural image denoising is to estimate a clean version of a given noisy image, utilizing prior knowledge on the statistics of natural images. However, it seems that image denoising algorithms are starting to converge and recent algorithms improve over previous ones by only fractional dB values. It is thus important to understand how much more can we still improve natural image denoising algorithms and what are the inherent limits imposed by the actual statistics of the data. (Levin et al., 2011[4]) take a non-parametric approach and derives a simple statistical measure which provides a lower bound on the optimal Bayesian minimum mean square error (MMSE). This imposes a limit on the best possible results of denoising algorithms which utilize a fixed support around a denoised pixel and a generic natural image prior. The empirical errors of all denoising algorithms are larger than our lower bound. Nonetheless, for most cases the PSNR values of BM3D are within 0.1dB of the optimal ones, suggesting that the BM3D results are quite close to optimality, for small patch sizes.

(Milanfar[5]) presents a practical and accessible framework used to arrive at new insights and methods. In particular, several novel optimality properties of algorithms in wide use such as BM3D. Noticed that specifying the kernel function or the corresponding weights is essentially equivalent to estimating a particular type of empirical prior from the given data, he makes a statistical analysis of BM3D with the filters performance.

While BM3D is a well-engineered algorithm, could we also automatically learn an image denoising procedure purely from training examples consisting of pairs of noisy and noise-free patches for data-driven models?

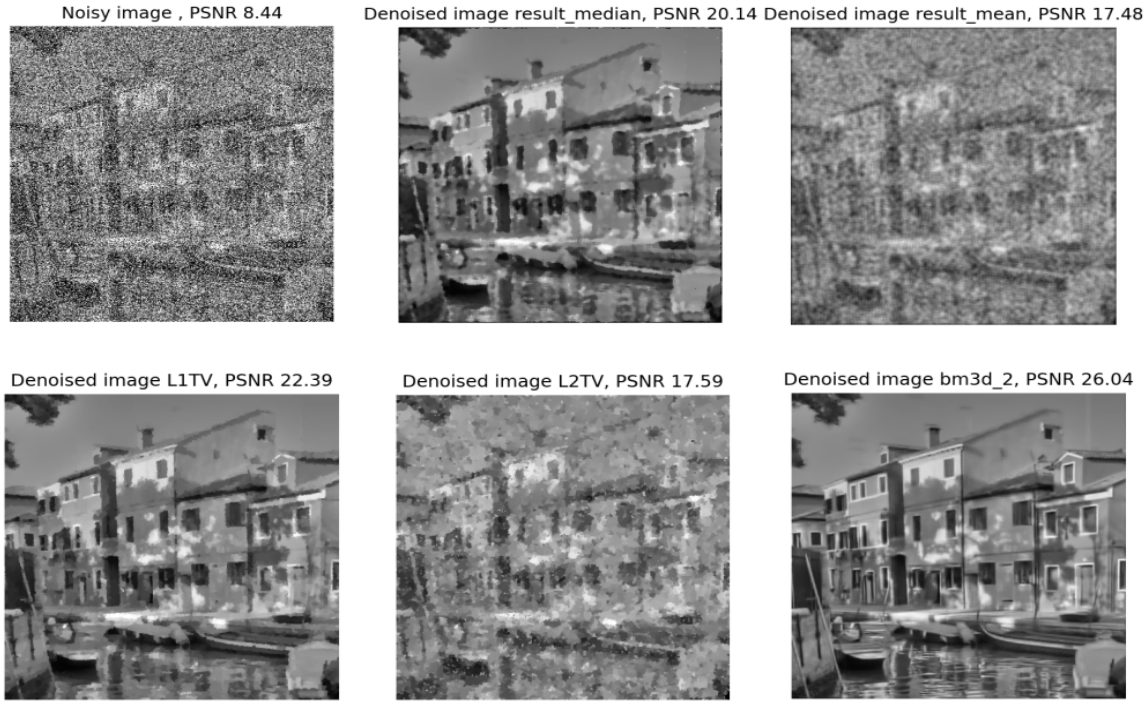

Figure 2: The initial grayscale salt-and-pepper noisy image with amount 0.5 and the images denoised by different methods.

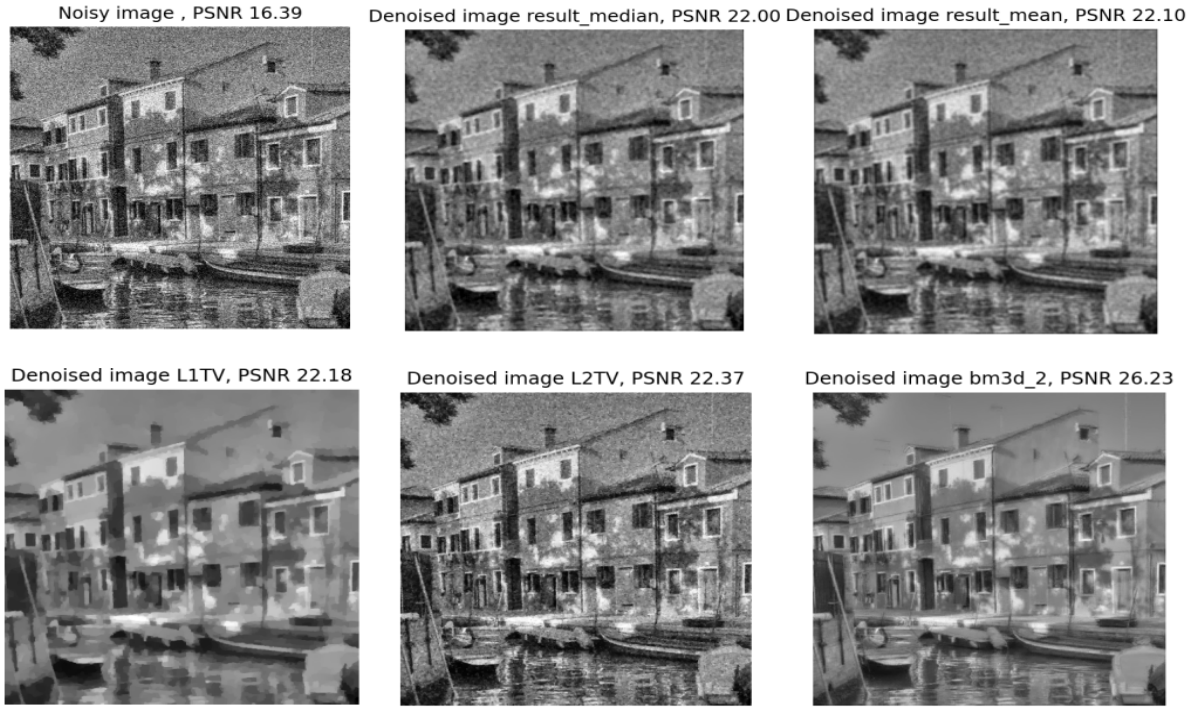

Figure 3: The initial grayscale Gaussian noisy image with $\sigma = 40$ and the images denoised by different methods.

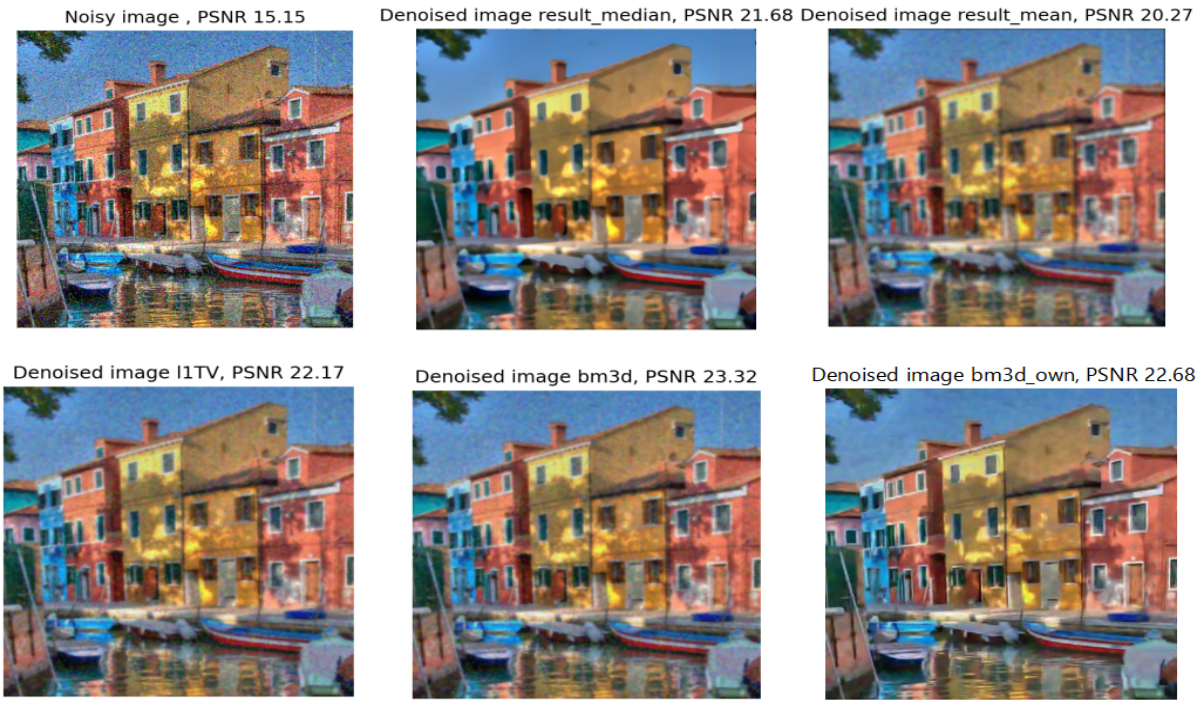

Figure 4: The initial color salt-and-pepper noisy image with amount 0.1 and the images denoised by different methods.

Figure 5: The initial color Gaussian noisy image with $\sigma = 40$ and the images denoised by different methods.

(Burger et al., 2012[6]) shows that it is indeed possible to achieve state-of-the-art denoising performance with a plain multi layer perceptron (MLP) that maps noisy patches onto noise-free ones.

With the development of deep learning, new algorithms such as Denoising Convolutional Neural Network(DnCNN)[7] have achieved better results. Many researchers are trying to improve the calculation speed of BM3D[8] as well as many researchers use BM3D as part of the hybrid model to get better performance than the original one. (Jancsary et al., 2012[9])introduced a powerful non-parametric image restoration framework based on Regression Tree Fields with BM3D as the filterbank and improved on the best published denoising method across a range of noise levels. As a state-of-the-art image denoiser, BM3D has been used for subroutine of the Plug-and-Play(PnP) framework with optimization models and achieved good results[10].

6 Conclusion

In this report, we study, implement and compare BM3D for image denoising. First, we describe the basic frameworks of BM3D algorithm. Then, we present how we implement BM3D with coding. Next, we show our results by comparing our own implementation of BM3D with mean filtering, median filtering, TV- denoising. Finally, we empirical analyzed BM3D in image denoising.

We encountered various difficulties while going through this project. In the process of constructing program of BM3D model, computation overflow is always a big trouble. Because of insufficient computing power, we also spent a lot of time searching for good parameters in TV-denoising.

Some possible open problems in the process of implementation came into our minds.
1. Why is PSNR so important? Is there a better measurement than PSNR? For instance, sometimes the denoised image analyzed by human mind significantly worse than others but the PSNR is higher especially in the mean value filter.
2. Is it possible to analyze denoising Neural Network and BM3D from a higher level perspective, and give a more convincing interpretability from mathematics or form an general framework?

Technology is iterating, and brilliant research may fade away, but they will always become a milestone in history.

References

[1] Dabov, Kostadin ; Foi, Alessandro ; Katkovnik, Vladimir ; Egiazarian, Karen: Image denoising by sparse 3-D transform-domain collaborative filtering. In: *IEEE Transactions on image processing* 16 (2007), Nr. 8, S. 2080–2095 1

[2] Maggioni, Matteo ; Katkovnik, Vladimir ; Egiazarian, Karen ; Foi, Alessandro: Nonlocal transform-domain filter for volumetric data denoising and reconstruction. In: *IEEE transactions on image processing* 22 (2012), Nr. 1, S. 119–133 1

[3] Lebrun, Marc: An analysis and implementation of the BM3D image denoising method. In: *Image Processing On Line* 2012 (2012), S. 175–213 1, 2, 5

[4] Levin, Anat ; Nadler, Boaz: Natural image denoising: Optimality and inherent bounds. In: *CVPR 2011* IEEE, 2011, S. 2833–2840 5

[5] Milanfar, Peyman: A Tour of Modern Image Filtering: New Insights and Methods, Both Practical and Theoretical. In: *IEEE Signal Processing Magazine* 2013 (2013), S. 106–128 5

[6] Burger, Harold C. ; Schuler, Christian J. ; Harmeling, Stefan: Image denoising: Can plain neural networks compete with BM3D? In: *2012 IEEE conference on computer vision and pattern recognition* IEEE, 2012, S. 2392–2399 8

[7] Zhang, Kai ; Zuo, Wangmeng ; Chen, Yunjin ; Meng, Deyu ; Zhang, Lei: Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. In: *IEEE transactions on image processing* 26 (2017), Nr. 7, S. 3142–3155 8

[8] Mäkinen, Ymir ; Azzari, Lucio ; Foi, Alessandro: Collaborative filtering of correlated noise: Exact transform-domain variance for improved shrinkage and patch matching. In: *IEEE Transactions on Image Processing* 29 (2020), S. 8339–8354 8

[9] Jancsary, Jeremy ; Nowozin, Sebastian ; Rother, Carsten: Loss-specific training of non-parametric image restoration models: A new state of the art. In: *European Conference on Computer Vision* Springer, 2012, S. 112–125 8

[10] Ryu, Ernest ; Liu, Jialin ; Wang, Sicheng ; Chen, Xiaohan ; Wang, Zhangyang ; Yin, Wotao: Plug-and-play methods provably converge with properly trained denoisers. In: *International Conference on Machine Learning* PMLR, 2019, S. 5546–5557 8