0510-6201 – Digital Signal Processing

Final Project

WNNVD – Weighted Nuclear Norm for Video Denoising

https://github.com/roy-hachnochi/WNNM-Video-Denoising

Roy Hachnochi 311120851
Uri Kapustin 317598811

Date: 30.09.2021

Table of Contents

1 Summary of the chosen paper

In this project we chose to focus on the paper "Weighted Nuclear Norm Minimization with Application to Image Denoising" [*Gu et. al.*] ([1]). The paper was published in CVPR2014 and expands the idea of Nuclear Norm minimization by adding weights to the optimization problem. The paper also proposes an algorithm for solving the optimization problem for various cases of the weight vector and demonstrates a practical use for an application of image denoising. We provide a short summary of the concepts presented in the paper.

1.1 Nuclear Norm

The nuclear norm of a matrix X is defined as:

$$\|X\|_* = \sum_i |\sigma_i(X)|$$

Where $\sigma_i(X)$ are defined as the singular values of X (ordered decreasing).

1.2 Weighted Nuclear Norm

As presented in the paper, the weighted nuclear norm of a matrix X is defined as:

$$\|X\|_{w,*} = \sum_i |w_i \sigma_i(X)|$$

Where $w_i \geq 0$ is the assigned weight of the singular value σ_i.

Obviously, this is an expansion of the Nuclear Norm, and reduces to it when given $w_i = 1 \ \ \forall i$.

The weighted nuclear norm is non-convex in general, making it much harder to optimize. However, the paper shows that when w_i are non-ascending the norm remains convex, and additionally proposes methods for optimization in other, more general, cases.

1.3 Image denoising

Based on previous papers ([2], [3]) the paper proposes the following low-rank minimization problem with a weighted nuclear norm regularization:

$$\hat{X}_j = argmin_{X_j} \frac{1}{\sigma_n^2} \left\|Y_j - X_j\right\|_F^2 + \left\|X_j\right\|_{w,*}$$

We will explain the terms of the above equation:

The image denoising algorithm is a non-local self-similarity (NSS) method, which takes advantage of the fact that a natural image holds many repetitions of local patches. This fact may be utilized by

finding estimates of these local patches using methods such as Block Matching ([4]) and using the similarity of the patches to achieve a better estimate of the original patch. We denote Y_j as the $n \times p^2$ matrix in which each row is a vectorized patch of the NSS group, $\hat{X}_j$ as the $n \times p^2$ estimation of the denoised matrix, and σ_n^2 as the noise level (variance), assumed to be known.

The optimization problem thus adds the assumption that $\hat{X}_j$ is actually a low-rank matrix because it's rows (patches) are similar. The first term in the optimization problem is the data-term, which aims for the estimated patches to be somewhat similar to the noised patches. The second term is the low-rank prior, aiming to achieve similar patches by forcing the matrix to be of low rank.

For the nuclear norm problem ($w_i = 1$) the solution was previously ([3]) shown to be:

$$\sigma_i(X) = S_{\sigma_n^2}\big(\sigma_i(Y)\big) = \max\{\sigma_i(Y) - \sigma_n^2, 0\}$$

The above is called the soft thresholding function. The problem with the solution for the non-weighted version is that it decreases each singular value by the same term, not taking advantage of the prior knowledge that the largest singular values actually hold the most information on the signal and we should thus apply a softer thresholding on them. The final calculation of the weights is:

$$w_i = \frac{c\sqrt{n}}{\sqrt{\max\{\sigma_i^2(Y_j) - n\sigma_n^2, 0\}} + \varepsilon}$$

1.4 The solution

The paper divides the solution under three cases regarding the weight vector:

1) <u>The weights are in a non-ascending order</u>:

   The paper proves that for this particular case the optimization problem is still convex, and that the solution is given by:

   $$\hat{X} = US_w(\Sigma)V^T$$

   Where $Y = U\Sigma V^T$ is the SVD of Y, and $S_w(\Sigma)$ is the generalized soft-thresholding operator, given by:

   $$S_w(\Sigma) = \max\{\Sigma_{ii} - w_i, 0\}$$

2) <u>The weights are in arbitrary order</u>:

   For this case the paper proposes an iterative process for finding $\hat{X}$, also using the generalized soft-thresholding operator. We will not expand on this since this case isn't important for our work, but only for the deduction of the last case.

3) The weights are in a non-descending order:

The paper proves that for this case the iterative process from above has a fixed point, given by the same solution of the first case:

$$\hat{X} = US_w(\Sigma)V^T$$

The final case is actually the most important one, since (as stated previously) we would like to give lower weights to the larger singular values, therefore resulting in a softer thresholding for them.

1.5 The algorithm

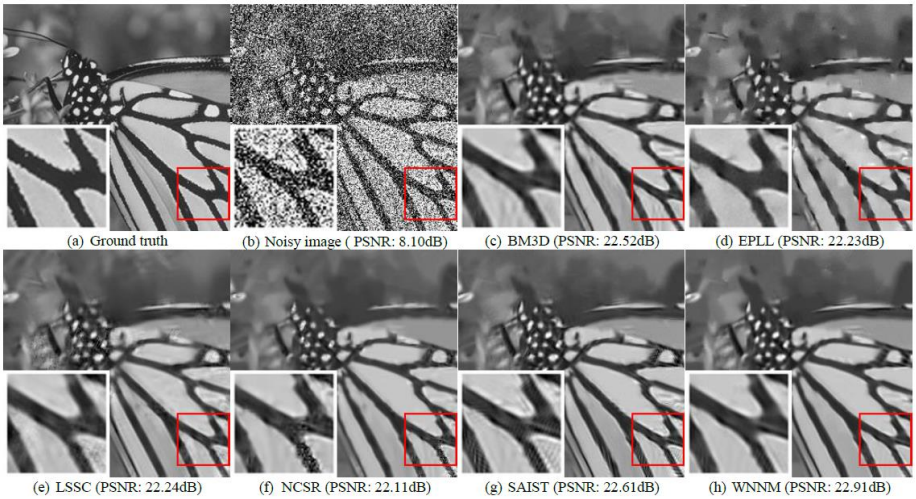
The Weighted Nuclear Norm Image Denoising algorithm (WNNID) presented in the paper follows the solution of the optimization problem in 1.4 and takes the following steps:

Algorithm 1 Image Denoising by WNNM

Input: Noisy image y
1: Initialize $\hat{x}^{(0)} = y, y^{(0)} = y$
2: **for** k=1:K **do**
3: Iterative regularization $y^{(k)} = \hat{x}^{(k-1)} + \delta(y - \hat{y}^{(k-1)})$
4: **for** each patch y_j in $y^{(k)}$ **do**
5: Find similar patch group Y_j
6: Estimate weight vector w
7: Singular value decomposition $[U, \Sigma, V] = SVD(Y_j)$
8: Get the estimation: $\hat{X}_j = US_w(\Sigma)V^T$
9: **end for**
10: Aggregate X_j to form the clean image $\hat{x}^{(k)}$
11: **end for**
Output: Clean image $\hat{x}^{(K)}$

1.6 Results

The algorithm achieved state-of-the-art results in its time, overcoming all competing algorithms for all noise levels and all test images. An example for a denoised image from the paper:



(a) Ground truth (b) Noisy image (PSNR: 8.10dB) (c) BM3D (PSNR: 22.52dB) (d) EPLL (PSNR: 22.23dB)
(e) LSSC (PSNR: 22.24dB) (f) NCSR (PSNR: 22.11dB) (g) SAIST (PSNR: 22.61dB) (h) WNNM (PSNR: 22.91dB)

2 Related work

The paper cites several other papers, the most worth mentioning ones to our opinion (which also helped us understand the context and ideas behind the innovation of the paper) are the following:

- "Image Denoising by Sparse 3-D Transform-Domain" (BM3D) ([4]):
  This algorithm was the state-of-the-art image denoising algorithm prior to our chosen paper. It is based on the same NSS principle that WNNID is based on, and also performs block matching and aggregation. The difference is the optimization problem being solved to clean each patch-group. In BM3D the groups are denoised using collaborative filtering and Wiener filtering.
  This algorithm is also used as a comparison for the WNNID algorithm. We note that the WNNID is much simpler, and also achieves better results (although it may have a higher computational cost).
  The WNNID algorithm also bases the block matching step on the block matching proposed by BM3D. This is a very important step in the algorithm, since the success may be only as good as the quality of the matched patches.
- "A non-local algorithm for image denoising" ([5]):
  Amongst other contributions, this paper presents the Nonlocal Self-Similarity (NSS) approach, which was vastly adopted in following publications, including WNNID.
- "A singular value thresholding algorithm for matrix completion" ([2]):
  This paper presented the Nuclear Norm Minimization problem as a relaxation of the NP-hard rank-minimization problem and proposed the soft-thresholding operator which was expanded in the WNNID paper.

We also mention several papers which expanded the work presented in our chosen paper:

- "Multi-Scale Weighted Nuclear Norm Image Restoration" ([6]):
  This paper adopted the idea of Weighted Nuclear Norm image denoising for the more general task of image restoration (e.g., deblurring, inpainting). They did so by proposing a half quadratic splitting (HQS) technique based on a generalized version of the problem presented in 1.3. They also added multi-scale patches (patches not only from the image itself, but also from scaled down versions of the image), relying on previous work which showed the property of recurring patches across different scales. The paper presented competitive and state-of-the-art results for the tasks of image deblurring and image inpainting.
- "Multi-channel Weighted Nuclear Norm Minimization" ([7]):
  This paper proposed an algorithm called MC-WNNM (Multi-Channel Weighted Nuclear Norm Minimization), expanding the grayscale image denoising performed by WNNM to a RGB

colored image denoising scheme. This is done by concatenating patches from the 3 channels to form the patch vector $y \in \mathbb{R}^{3p^2}$ (p being the patch size) and formulating the following optimization problem:

$$\min_X \|W(Y - X)\|_F^2 + \|X\|_{w,*}$$

Where $Y, X \in \mathbb{R}^{3p^2 \times K}$ are the grouped noise and estimated similar patches appropriately, and $W \in \mathbb{R}^{3p^2 \times 3p^2}$ is a weight matrix to balance the noise levels between channels. The paper also proposes a solution for this problem, employing the variable splitting method ([8], [9]) and solving using the alternating direction method of multipliers (ADMM) ([10]).

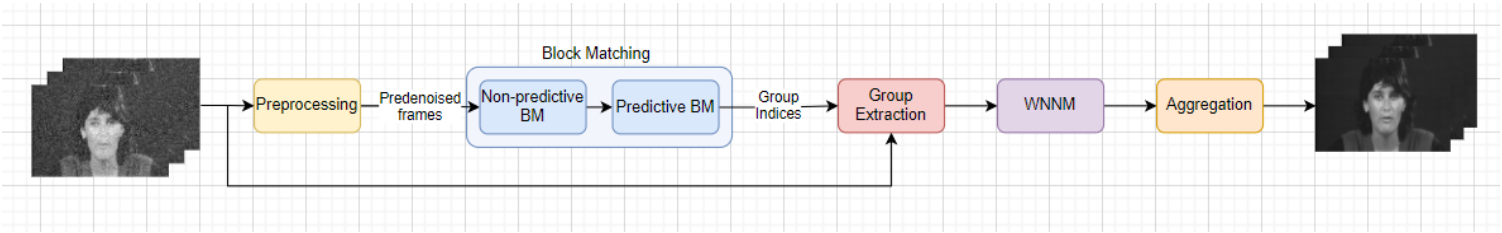
- "An Improved WNNM Algorithm for Image Denoising" ([11]):
  This paper shows that the WNNM algorithm, while achieving state-of-the-art results for white gaussian noise denoising, attains bad performance for salt & pepper denoising. The paper solves this problem simply by performing a two-stage algorithm: firstly, using WNNM to denoise the image, and then applying adaptive median filtering to process the remaining salt & pepper noise.
- Additionally, we also note that many other papers from the following years cite this paper as a state-of-the-art image denoising algorithm.

3 Our project – WNNVD

After reading the paper and the additional related papers, and seeing the impressive results achieved for the image denoising task, we initially thought to expand the idea of using WNNM for image denoising to use it for image deblurring, an idea which was not explored in the original paper. We soon found out that [6] dealt exactly with this problem and solved it impressively. We also thought about expanding the proposed algorithm to deal with RGB (multi-channel) images and to check the performance of the algorithm on more types of noise (Salt & Pepper, Poisson…).
Eventually we chose to expand the idea of image denoising to the video denoising task. We will describe our chosen method, note our innovation, show some results, and discuss further possible improvements.

3.1 Method

The obvious and simplest approach would be to apply the WNNID algorithm per-frame sequentially. Except from being time-consuming, this naive method lacks a strong assumption which may be taken in advantage when dealing with videos. This assumption is that there is a high temporal familiarity in videos, meaning that many patches are repeated between neighboring frames. For example, the background stays almost exactly the same, or take a moving ball which looks the same between neighboring frames but only slightly changes its location.
This important assumption is what drives our innovation – we improve the block matching phase of the algorithm to find not only spatially similar patches, but also search in the temporal dimension. This will result in two advantages. The first is that we now base our WNNM phase on more/better matched patches in each group. The second is that with this method we process patches from several frames at once, sparing the need to process each frame individually and thus allowing a lower computational cost.
Given a noised video $Y \in \mathbb{R}^{h \times w \times f}$, our model may be described using the following block diagram:



3.1.1 Preprocessing

As suggested in many image/video denoising algorithms based on the concept of block matching, including [4], we perform a preprocessing phase which includes a naive per-frame denoising method, e.g., Gaussian filtering, Median filtering. The denoising helps in the block matching process by making patches which were originally similar but were contaminated by noise, become slightly more similar again, thus making them easier to match. Note that the chosen denoising methods may be changed orthogonally to the WNNVD algorithm to match specific priors on the video/noise.

3.1.2 Block Matching

While this part of the algorithm is not the true innovation of the original paper, this may be considered the most crucial part since low-quality block matching will result in un-correlated groups and thus in a bad patch-restoration when applying the WNNM method. Furthermore, this is where our true innovation lies.

As explained in 3.1, we wish to take advantage of the temporal correlation between patches in addition to the original spatial correlation suggested in [1]. We therefore adopt the video block matching method proposed in [12], which may be described as follows for each of N reference patches described by the indices (R, C, T):

Non-predictive block matching:

We search for patches in the current frame which are similar to the reference patch by a brute-force search in a $m_{NP} \times m_{NP}$ window around the reference patch indices, using a stride of s_{NP} to reduce computational cost. Distances for all patches in this window are calculated, and the k most similar patches are chosen. This results in the matrix $B^{(T)} \in \mathbb{N}^{k\times 3}$ of nearest patch indices in the current frame, and the vector $d^{(T)} \in \mathbb{R}^{k\times 1}$ holding their distances.

Predictive block matching:

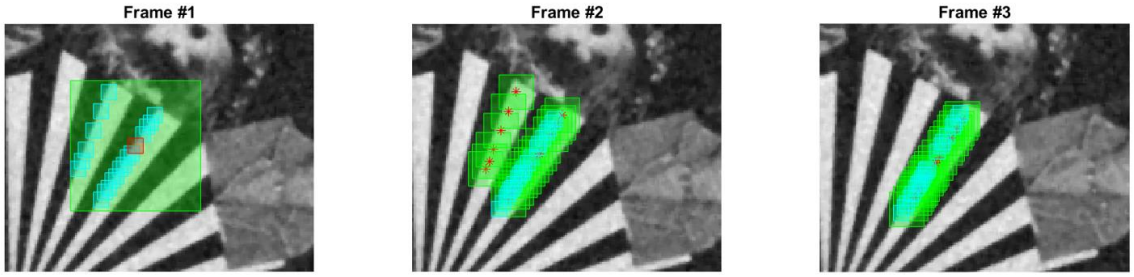
For each $t \in \{T - M, \dots, T + M\}$ (M being the temporal search window, and excluding the frame T (and corner cases of the video where $T - M < 1$ or $T + M > f$), we take the matched patches from the previous frame ($B^{(t-1)}$ if $t > T$ and $B^{(t+1)}$ if $t < T$) and open k search windows around each patch index, with a search window of size $m_P \times m_P$ and stride s_P. Note that m_P may be significantly smaller than m_{NP} since we have already found the spatially similar patches and now

should only find their movement between neighboring frames. Once again, distances for all patches in all k search windows are calculated, and the k most similar patches are chosen, resulting in $B^{(t)} \in \mathbb{N}^{k\times 3}$ and $d^{(t)} \in \mathbb{R}^{k\times 1}$.

After finding $B^{(t)}$ and $d^{(t)}$ for all $t \in \{T-M, \dots, T+M\}$, the K most similar patches are chosen for the group, resulting in the matrix $G_i \in \mathbb{N}^{K\times 3}$.

The non-predictive and predictive searches are performed for each reference patch, ultimately resulting in the matrix $G \in \mathbb{N}^{N\times K\times 3}$.

Below is an example for predictive block matching through 3 following frames:



Frame #1 is the reference frame, the red rectangle is the reference patch, the green rectangles represent the search window (notice the larger window for the non-predictive BM in frame #1, and the multiple smaller windows for the following frames), the cyan rectangles are the matched patches per frame, which are also the indices for the search windows for the following frame. Notice how even though the wheel moves a bit between frames, our predictive block matching process allows finding the translated patches.

3.1.3 Group Extraction

For each $G_i \in \mathbb{N}^{K\times 3}$ of the N groups found in 3.1.2, the patches are extracted from the video, vectorized, and concatenated to form the matrix $Y_i \in \mathbb{R}^{K\times p^2}$ (where p denotes the patch size).

3.1.4 WNNM

This step is actually identical to that of the original WNNID algorithm, since it takes a group matrix $Y_i \in \mathbb{R}^{K\times p^2}$, blind to whether the patches were extracted from the same frame or different frames.

This step performs the WNNM thresholding algorithm and returns $\hat{X}_i \in \mathbb{R}^{K \times p^2}$, by applying the generalized soft-thresholding operator as presented in 1.4.

3.1.5 Aggregation

Two matrices are formed in this step: $\tilde{X} \in \mathbb{R}^{h \times w \times f}$ and $W \in \mathbb{N}^{h \times w \times f}$. After denoising the patch group and obtaining $\hat{X}_i$, each denoised patch is then placed back in its corresponding position using G_i. Since different patches may overlap, we actually sum all patches to $\tilde{X}$, and hold a count matrix for each pixel W. After placing all denoised patches, $X \in \mathbb{R}^{h \times w \times f}$ is obtained by dividing $\tilde{X}/W$ pixel-wise. Pixels which were not aggregated in this step are simply taken from the original video Y.

3.1.6 Loop

Since the above steps are performed using a single reference frame T, and since some frames/patches may remain un-processed after these steps due to not reaching the frame or not grouping the patches, we perform steps 3.1.2-3.1.5 multiple times, each time choosing a different reference frame based on the frame with the least processed pixels. In each iteration we mark all processed pixels in the video, and use this for the following:

1. Choosing the next reference frame.
2. Checking if the algorithm has finished processing enough (parameter) pixels.
3. Skipping reference patches which were already processed entirely. This significantly aids in reducing runtime, since towards the end of the algorithm most patches have already been processed and will thus be skipped.

Note that the algorithm is guaranteed to end since for each reference frame, all its pixels are automatically marked as processed since they all appear in reference patches.

3.2 WNNVD extention

As explained in 3.1, there is a great significance to the temporal information which we wish to exploit. Indeed, we use this in the WNNVD implementation as explained in the previous section. But, as explained in [13], there is much more significance to temporal correlations than spatial ones. They proposed a different "Blocking" mechanism instead of the BM presented in 3.1.2. We tried to mimic their approach in order to gain performance improvement.

The method is first to identify trajectories along several frames from a reference patch. After that, like in the original BM, we find similar trajectories in the spanned frames of the reference trajectory. In [13] they used a "motion estimation" technique but we leveraged the predictive and non-

predictive blocks to perform such a trajectory creation. This is because implementing other ways would be out of the scope of this project.

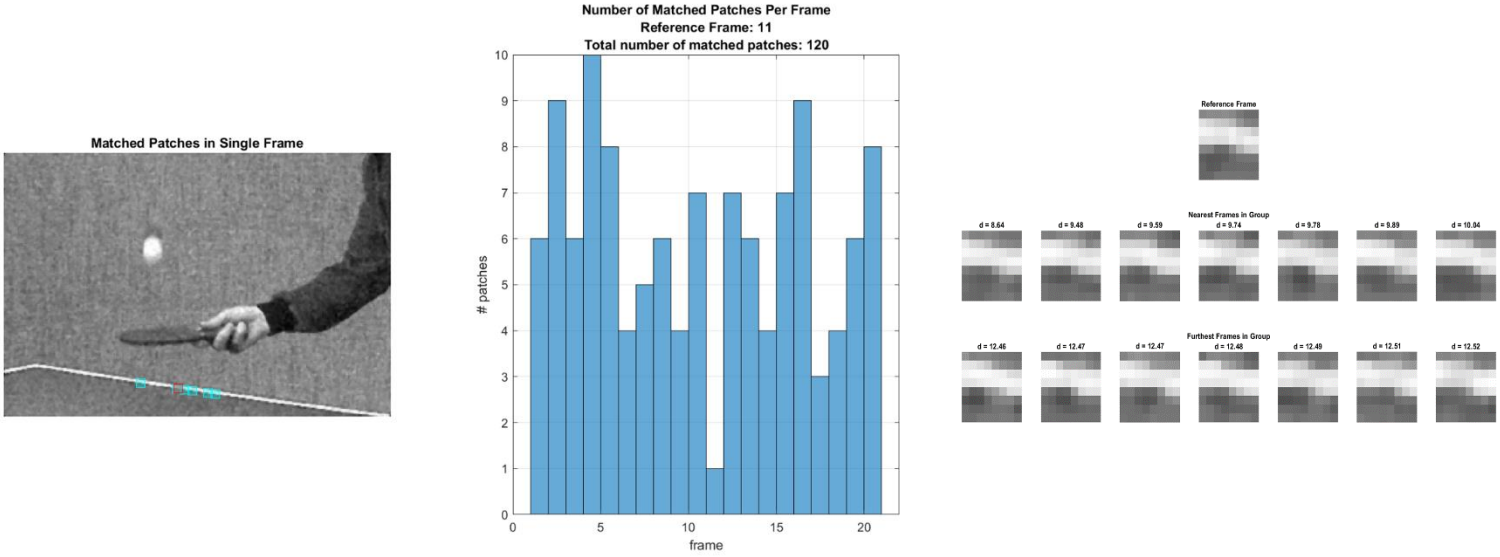
In our implementation, each trajectory was created by calling predictive search with $k = 1$ for the defined span of frames. We calculate a reference trajectory for some reference frame and then, using non-predictive search, we choose patches which will be expanded to trajectories. We then vectorize and stack entire trajectories to the matrix Y_i instead of single patches.

This method presented poor performance which we will not show. We believe this is due to the vectorization of the trajectories which causes the WNNM step to treat them as single data points, and thus not utilizing the temporal correlations inside the trajectories themselves. We instead propose to stack the trajectories as 3D matrices and find a way to apply the WNNM step on this type of data structure (more in 3.4).

3.3 Results

3.3.1 Block Matching

We first show some analysis of our block matching process:



Above we may see an example of the block matching process, performed on a temporal window of 10 frames (per direction). On the left is the reference patch (in red) and the matched patches in the specific frame (in cyan).

On the middle is a histogram showing how many patches from each frame were used to form the group. There is a fairly uniform distribution for this specific reference patch (except for the reference frame). From this we may deduce that the temporal block matching aids in finding

matching patches across the different frames in the video as expected, and thus allowing the denoising process to work on multiple frames at a time.

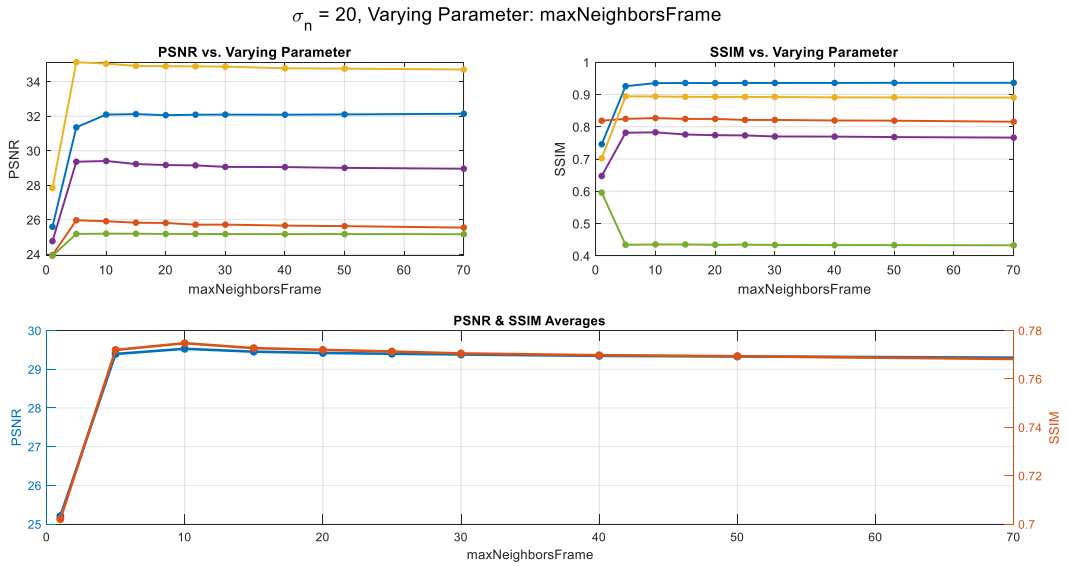
The plot on the right shows the matched patches and their distances from the reference patch (nearest patches on top row and furthest on bottom), validating the correctness of the block matching.

3.3.2 Parameter Analysis

We perform an analysis of the effect of several interesting parameters of the algorithm:

Maximum Patches Per Frame:

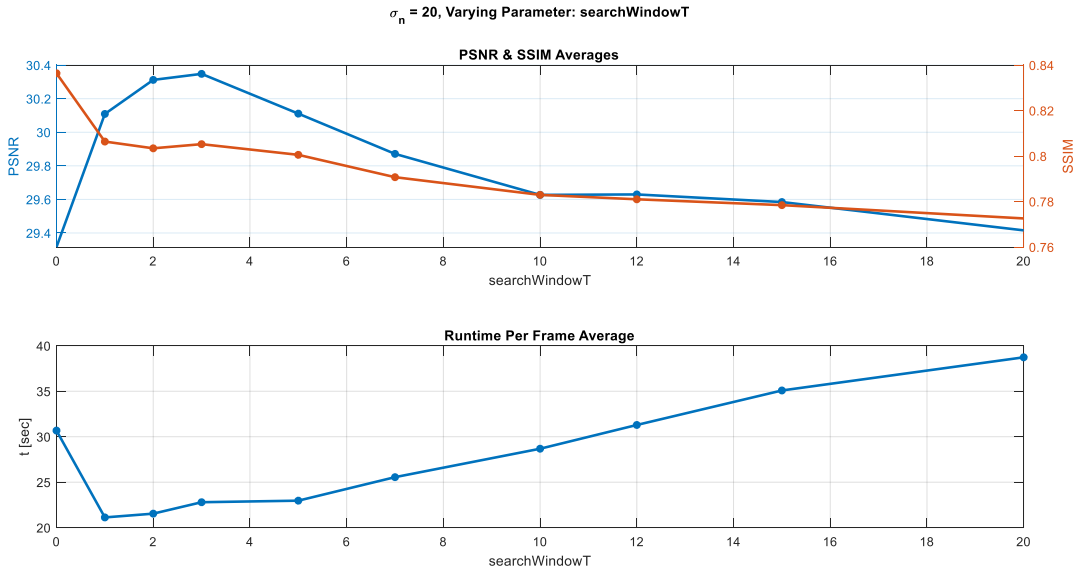
This parameter controls the maximal number of matched patches in each frame, and hence limits the ratio of spatial patches vs. temporal patches in the group. Setting this parameter to 1 reduces to an exclusively temporal search which doesn't take spatially similar patches but only looks for the most similar patch in each frame.



As expected, using an exclusively temporal search isn't as good as combining spatial and temporal patches. Interestingly though, there isn't much effect of the maximal number of patches per frame on the results after around 10 patches.

Temporal Search Window:

This parameter controls how many adjacent frames are being processed at once. In some way this parameter also affects the ratio of spatial patches vs. temporal patches, because setting it to 0 reduces to using only spatial patches, which is actually equivalent to performing WNNID on each frame separately.



As expected, using only spatial patches (WNNID per frame) is inferior to combining temporal patches, because this setting doesn't harness the temporal redundancy in the video. Additionally, it is noticeable that the time per frame of the naive WNNID solution is larger from most configurations. We deduce that using a fairly low temporal window, but not 0, is better both in computational costs and in result metrics.

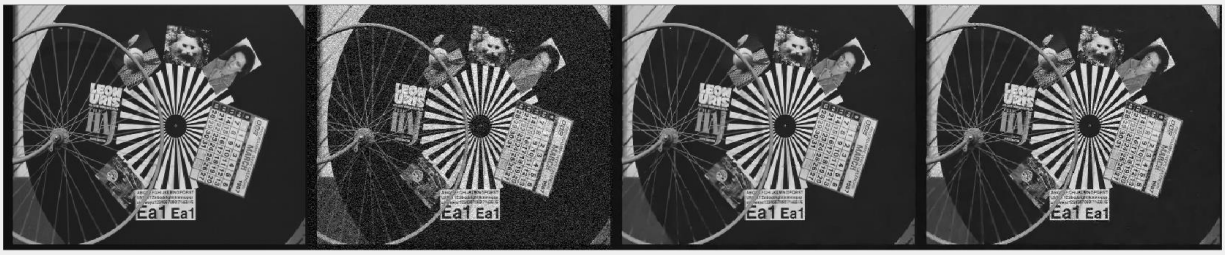
3.3.3 Comparison

Noise STD	Video	Algorithm	PSNR	SSIM	Time per frame [sec]
10	gbicycle	VBM3D	**36.5**	**0.97**	**1.12**
		WNNID	34.2	**0.97**	73.35
		WNNVD	36.1	**0.97**	62.92
	gflower	VBM3D	**29.8**	**0.95**	**0.19**
		WNNID	27.7	0.93	11.23
		WNNVD	28.8	0.94	12.03
	gmissa	VBM3D	**39.3**	**0.93**	**0.26**
		WNNID	37.8	0.92	21.12
		WNNVD	38.2	0.92	16.23
	gsalesman	VBM3D	**36.4**	**0.95**	**0.24**
		WNNID	32.9	0.91	19.24
		WNNVD	34.2	0.92	10.18
	gstennis	VBM3D	**31.8**	**0.87**	**0.21**
		WNNID	29.6	0.83	16.55
		WNNVD	29.2	0.77	12.92
	lena	WNNVD	35.8	0.92	110.4

Noise STD	Video	Algorithm	PSNR	SSIM	Time per frame [sec]
20	gbicycle	VBM3D	**33.6**	**0.95**	**1.13**
		WNNID	29	0.92	72.36
		WNNVD	32.6	**0.95**	63.79
	gflower	VBM3D	**27.5**	**0.91**	**0.19**
		WNNID	23.7	0.85	11.01
		WNNVD	26	0.85	14.79
	gmissa	VBM3D	**37.5**	**0.91**	**0.26**
		WNNID	34.1	0.88	20.52
		WNNVD	35.5	0.9	17.37
	gsalesman	VBM3D	**33.6**	**0.91**	**0.25**
		WNNID	28.8	0.81	19.12
		WNNVD	30.1	0.82	11.78
	gstennis	VBM3D	**29.6**	**0.77**	**0.21**
		WNNID	26.8	0.67	16.39
		WNNVD	26.3	0.53	15.27
	lena	WNNVD	33.1	0.88	114

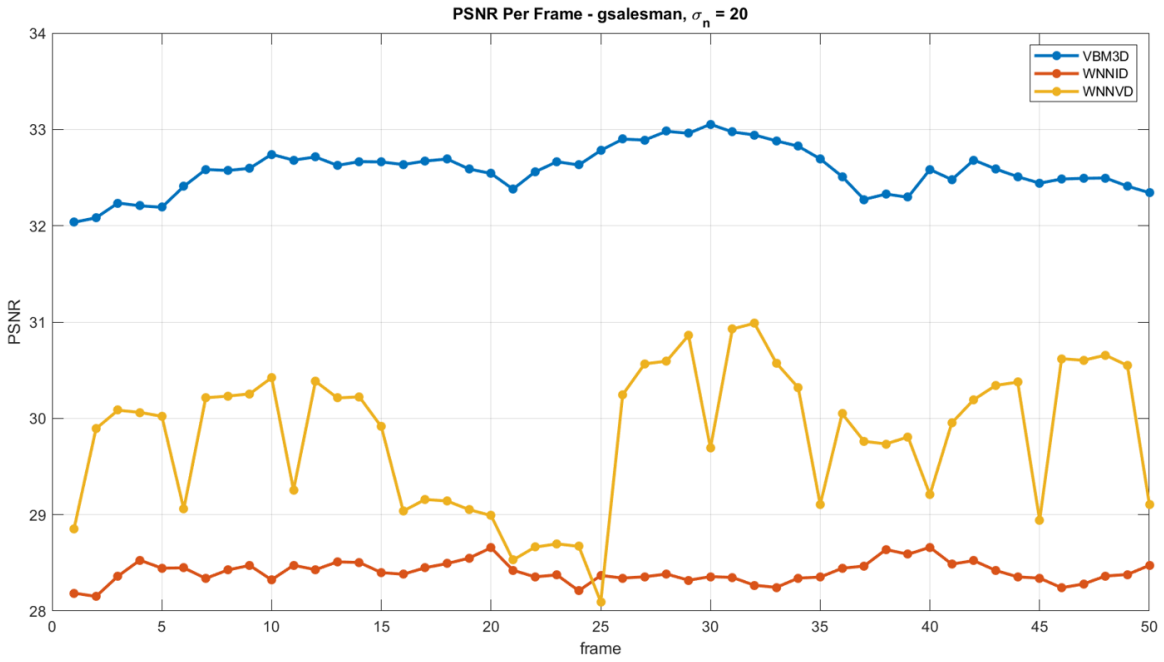
Noise STD	Video	Algorithm	PSNR	SSIM	Time per frame [sec]
30	gbicycle	VBM3D	**31.1**	**0.93**	**1.13**
		WNNID	26.5	0.9	71.66
		WNNVD	29.4	0.9	65.86
	gflower	VBM3D	**25.4**	**0.86**	**0.19**
		WNNID	21.2	0.78	11.07
		WNNVD	22.6	0.74	15.64
	gmissa	VBM3D	**35.6**	**0.91**	**0.26**
		WNNID	32.6	0.88	20.47
		WNNVD	33.3	0.88	18.24
	gsalesman	VBM3D	**30.9**	**0.85**	**0.25**
		WNNID	26.7	0.73	19.03
		WNNVD	27.3	0.72	13.27
	gstennis	VBM3D	**27.7**	**0.66**	**0.21**
		WNNID	25.3	0.57	16.22
		WNNVD	25.6	0.5	17.47
	lena	WNNVD	31.3	0.85	108.6

It is clearly noticeable that VBM3D outperforms both our WNNVD and the naive WNNID algorithms in PSNR, SSIM, and runtime. Since the scope of this project was not the optimization of the parameters and runtime, we refer to comparing our WNNVD to the WNNID algorithm, in which there is a clear improvement in all metrics. We also performed a qualitative analysis if the resulting videos and found that WNNVD resulted in much cleaner videos than WNNID, for example:



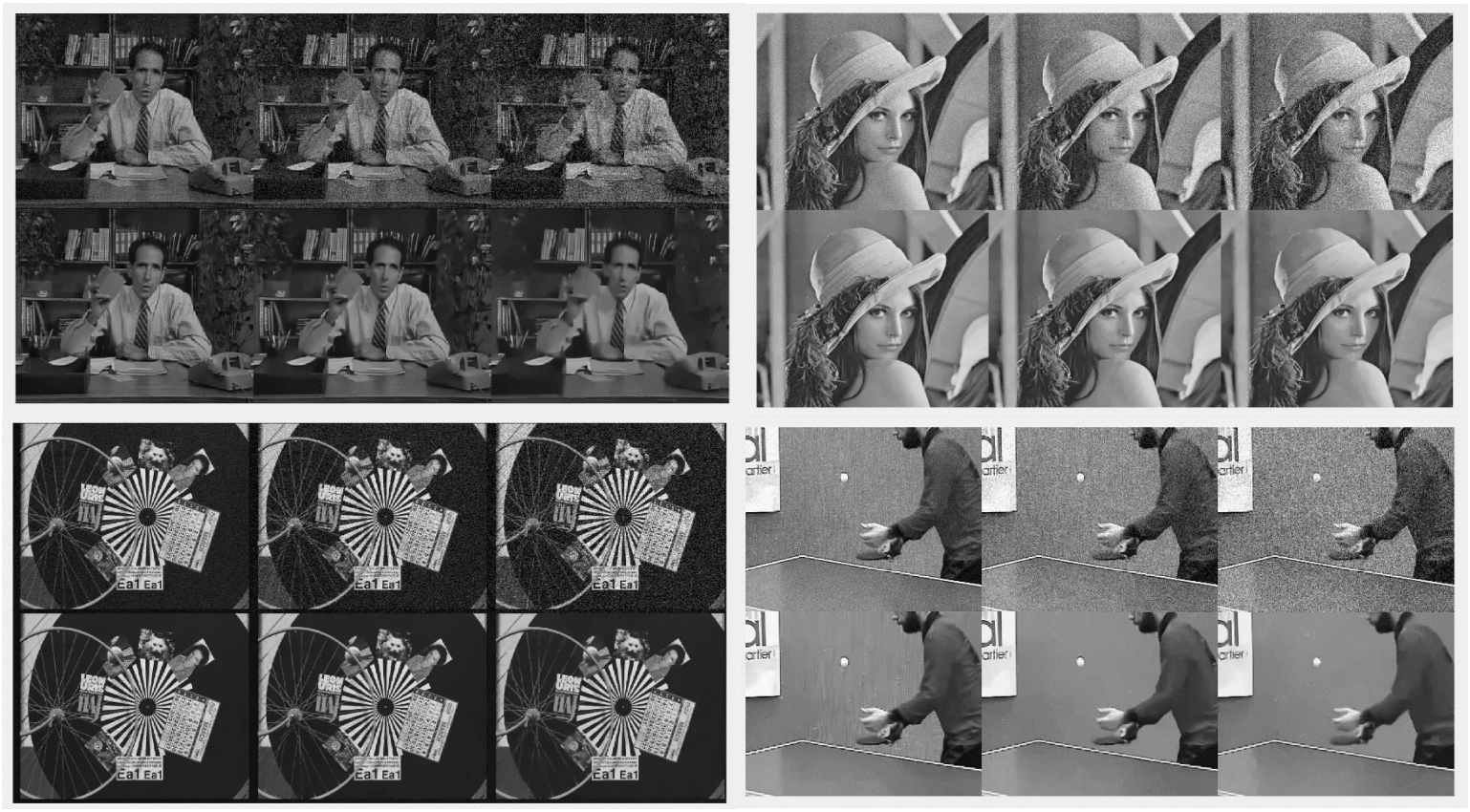
From left to right: Original, Noised (STD = 20), WNNVD denoising, WNNID denoising. The center of the wheel and the photos are clearly noised in the WNNID while WNNVD managed to denoise it successfully.

We also show an example of the PSNR per frame for one of the videos:

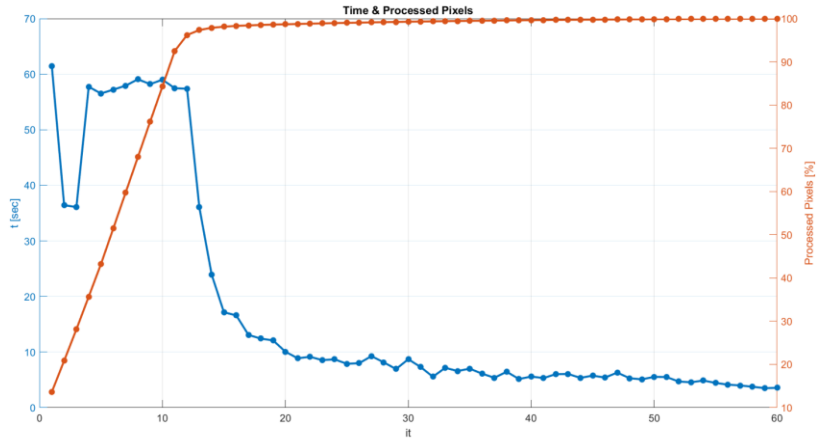


3.3.4 Examples

We show some examples of denoised frames from various videos using our WNNVD algorithm, and also show the result on an image (single frame) to verify that our algorithm is indeed an extension of the WNNID algorithm and works on images as well as videos. Each example presents denoising of the same frame for three different noise levels (10, 20, 30), the upper row contains the noised versions, and the lower row contains the denoised frames.



To support mark 3 of 3.1.6, we show an example of the percentage of processed pixels and iteration runtime throughout the algorithm:



At the beginning the middle frame is chosen as a reference frame. Then, due to our reference frame choosing mechanism, the first and last frame are chosen – and hence a lower runtime (only single sided block matching). Then is the phase where other frames are chosen, and then the drop in the runtime which is caused by the fact that most frames have already been processed and we skip the reference patches which were already processed. We may see that this mechanism reaches 95% after only 12 iterations, on a video of 60 frames.

3.4 Future work

We propose several directions for future work on the project, based on our results and insights from our work:

- As may be seen in 3.3.3, our results aren't as good as expected for most. This may be a result of poor hyperparameters, we suggest further research regarding the effect and refinement of the hyperparameters. Specifically, we suggest focusing on the lower-quality results presented in 3.3.3.
- The algorithm may be expanded to handle 3D blocks (as presented in 3.2), also taking the temporal dimension into account. This may be done, in our opinion, in two options. The first is simply expanding the block matching to handle 3D blocks instead of 2D and dispense the predictive block matching in favor of a simple exhaustive 3D search. The second is a more complex idea: for each reference patch find a trajectory and form a 3D block of matching patches across frames, then perform the block matching on these 3D trajectories, as done in [13]. The latter should also be combined with an expansion of the SVD part of the algorithm to a high-order SVD (HOSVD) ([14]) in order to harness both the spatial and the temporal redundancy simultaneously.
- An interesting expansion may be to implement a causal version of the algorithm, which in time (together with runtime improvements) may be used as an online video denoising algorithm. This expansion should not be too difficult, as the inner methods of the algorithm are kept unchanged and only a wrapper which uses a buffer of frames should be added.
- As seen in 3.3.3, the runtime of the algorithm is significantly higher than the competing VBM3D algorithm. We invested many efforts in reducing runtime especially in the block matching part, but we believe that much more could be improved in the group denoising part. Also, for a fair comparison between WNNVD and VBM3D regarding runtime, a mex (Matlab executable) should be compiled which will be much faster. In order to compile our code to an executable, there might be several required changes in the code, and a compiling script should be written.
- In the same way that WNNID may be expanded to handle colored RGB images, our algorithm may also be expanded to denoise colored videos. The naive way to implement this is to simply expand the entire algorithm to work on multi-channel frames (by simply concatenating the channels in the vectorization in the block matching and grouping phases). Another way to do so, as proposed by [6], would be to convert the frames to the YCbCr color space, apply the algorithm only on the luminance channel, and handle the chrominance channels using a simple interpolation, and then converting back to RGB. The latter is more elegant since it requires no expansion of the WNNVD algorithm itself and also should require less computational cost because we are dealing with smaller patches (single channel).

4 Conclusion

In this project, we conducted research in the field of image and video denoising. We focused our work on the WNNID algorithm, and also learned of some interesting expansions or competing algorithms. We expanded the idea presented in the original paper for image denoising to a video denoising framework, by combining several methods used in other video denoising papers and proposed the WNNVD algorithm. We conducted extensive research and analysis of our proposed algorithm and implementation, showed an improvement over the naive solution, and compared to a state-of-the-art algorithm.

References

[1] L. Z. W. Z. a. X. F. S. Gu, "Weighted nuclear norm minimization with application to image denoising," *IEEE International Conference on Computer Vision and Pattern Recognition,* 2014.

[2] E. J. C. a. Z. S. J.-F. Cai, "A singular value thresholding algorithm for matrix completion," *SIAM Journal on Optimization,* vol. 20, no. 4, p. 1956–1982, 2010.

[3] G. S. a. X. L. W. Dong, "Nonlocal image restoration with bilateral variance estimation: a low-rank approach," *TIP,* vol. 22, no. 2, pp. 700-711, 2013.

[4] A. F. V. K. a. K. E. K. Dabov, "Image denoising by sparse 3-d transform-domain collaborative filtering," *TIP,* vol. 16, no. 8, pp. 2080-2095, 2007.

[5] B. C. a. J.-M. M. A. Buades, "A non-local algorithm for image denoising," *CVPR,* 2005.

[6] N. &. M. T. Yair, "Multi-scale weighted nuclear norm," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* p. 3165–3174, 2018.

[7] L. Z. D. Z. a. X. F. Jun Xu, "Multi-channel weighted nuclear norm minimization for real," *IEEE International Conference on Computer Vision,* p. 1105–1113, 2017.

[8] R. Courant, "Variational methods for the solution of problems of equilibrium and vibrations," *Bull. Amer. Math. Soc.,* vol. 49, no. 1, pp. 1-23, 1943.

[9] J. E. a. D. P. Bertsekas, "On the Douglas–Rachford splitting method and the proximal point algorithm for maximal monotone operators," *Mathematical Programming,* vol. 55, no. 1, pp. 293-318, 1992.

[10] N. P. E. C. B. P. a. J. E. S. Boyd, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.,* vol. 3, no. 1, pp. 1-122, 2011.

[11] J. Wu and X. Lee, "An Improved WNNM Algorithm for Image Denoising," *J. Phys. Conf.,* vol. 022037, p. 1237, 2019.

[12] A. F. K. E. K. Dabov, "Video Denoising by Sparse 3D Transform-Domain Collaborative Filtering," *EUSIPCO,* vol. 15, 2007.

[13] S. R. Y. B. B. Wen, "VIDOSAT: High-dimensional Sparsifying Transform Learning for Online Video Denoising," *IEEE Transactions on Image Processing,* vol. 28, no. 4, pp. 1691-1704, 2019.

[14] L. R. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika,* vol. 31, no. 3, pp. 279-311, 1966.