JYX
JYVÄSKYLÄN YLIOPISTO
UNIVERSITY OF JYVÄSKYLÄ

This is a self-archived version of an original article. This version may differ from the original in pagination and typographic details.

Author(s): Wolfmayr, Monika

Title: Parameter Optimization for Low-Rank Matrix Recovery in Hyperspectral Imaging

Year: 2023

Version: Published version

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland.

Rights: CC BY 4.0

Rights url: https://creativecommons.org/licenses/by/4.0/

Please cite the original version:

Wolfmayr, M. (2023). Parameter Optimization for Low-Rank Matrix Recovery in Hyperspectral Imaging. Applied Sciences, 13(16), Article 9373. https://doi.org/10.3390/app13169373

applied sciences

MDPI

Article

Parameter Optimization for Low-Rank Matrix Recovery in Hyperspectral Imaging

Monika Wolfmayr [1,2]

[1] Institute of Information Technology, Jamk University of Applied Sciences, 40101 Jyväskylä, Finland; monika.wolfmayr@jamk.fi
[2] Faculty of Information Technology, University of Jyväskylä, 40014 Jyväskylä, Finland

Abstract: An approach to parameter optimization for the low-rank matrix recovery method in hyperspectral imaging is discussed. We formulate an optimization problem with respect to the initial parameters of the low-rank matrix recovery method. The performance for different parameter settings is compared in terms of computational times and memory. The results are evaluated by computing the peak signal-to-noise ratio as a quantitative measure. The potential improvement in the performance of the noise reduction method is discussed when optimizing the choice of the initial values. The optimization method is tested on standard and openly available hyperspectral data sets, including Indian Pines, Pavia Centre, and Pavia University.

Keywords: noise reduction; nonlinear optimization; low-rank modeling; hyperspectral imaging; signal-to-noise ratio improvement

1. Introduction

In hyperspectral imaging (HSI), spectral signatures of objects are recorded for each image pixel. The spectral signature of an object is the reflectance variation or function with respect to the wavelength. It is important for characterizing materials and their properties. In HSI, different types of noise appear due to environmental or instrumental influences. The noises include Gaussian noise [1], impulse noise [2], dead pixels or lines [3], and stripes [4]. This has been recently discussed, for instance, in [5] and in the overview articles [6,7]. HSI combines spatial and spectral information in a hyperspectral data cube, as displayed in Figure 1. Naturally, the amount of generated data is huge, and an efficient and reliable approach to noise reduction takes advantage of the internal dependencies between the wavebands. HSI precision and reliability are essential for many applications, including digitalization and robotization in Earth and space exploration. The applications include agricultural monitoring [8], atmospheric science [9], geology [10], and space exploration [11]. Efficient and reliable noise reduction techniques are essential for image processing in practice regarding real-time decision-making and automation. In [12], various noise reduction techniques have been compared for hyperspectral image data in asteroid imaging, including also low-rank matrix recovery (LRMR).

LRMR is a low-rank modeling approach [13] and has been discussed among other advanced image processing methods in more detail in [6,7]. We use here LRMR together with the Go Decomposition (GoDec) numerical optimization algorithm as presented in [14] in the inner iteration steps of the approach for solving the restoration model. Parameter optimization in advanced image processing can provide important indirect information for control and real-time decision-making. The main idea here is to optimize the choice of the algorithm's initial values in order to improve the performance of the noise reduction. We analyze the parameter choices, including applying nonlinear optimization methods as derived in [15]. The open, available hyperspectral data sets Indian Pines [16], Pavia Centre [17], and Pavia University [17] are used for the computational tests.

check for updates

Citation: Wolfmayr, M. Parameter Optimization for Low-Rank Matrix Recovery in Hyperspectral Imaging. *Appl. Sci.* **2023**, *13*, 9373. https://doi.org/10.3390/app13169373

Academic Editors: Dimitris Mourtzis and Paul Geladi

Received: 14 May 2023
Revised: 4 August 2023
Accepted: 16 August 2023
Published: 18 August 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

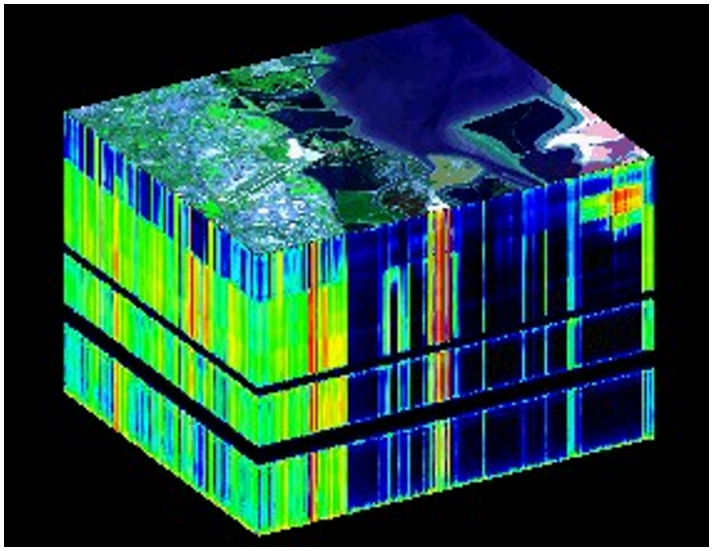

Figure 1. An example of a hyperspectral data cube (courtesy of NASA/JPL-Caltech).

This work is part of the coADDVA—ADDing VAlue by Computing in Manufacturing project funded by the Regional Council of Central Finland/Council of Tampere Region and European Regional Development Fund. It supports the project's goals to improve the efficiency of robotics by developing optimal control methods leading to flexible imaging and automation in image processing.

The article is organized as follows: In Section 2, the methods are discussed, including LRMR and the optimization applied to its initial parameter settings. The used data sets are presented. Section 3 presents the computational results. Section 4 discusses the results according to the existing literature and possible future work. Finally, Section 5 presents the main contributions of this work.

2. Methods and Materials

2.1. Methods

The first LRMR model was proposed in [18] as a robust principle component analysis approach. It was further developed in [13] for hyperspectral image restoration by combining nonlinear optimization in the inner iteration loop in order to solve the restoration model. In this work, we apply optimization with respect to the initial parameters in terms of an outer iteration loop.

In the following, we present the LRMR model. Given the real matrix D of size $m \times n$ containing the observed data and assuming corruption by the sparse error matrix S and random Gaussian noise modeled by the matrix N, the goal is to recover the low-rank matrix L with $D = L + S + N$, which are all real-number matrices of the same size: $m \times n$. The minimization problem

$$\min_{L,S} \|D - L - S\|_F^2 \text{ s.t. } \operatorname{rank}(L) \leq r, \operatorname{card}(S) \leq p \tag{1}$$

is solved, with r denoting the upper bound for the rank of L and p for the cardinality of S, which is related to the estimation of noise corruption. The norm $\|\cdot\|_F$ denotes the Frobenius-norm. The formulation (1) of LRMR can be found in [13,14]. Redundancies between the wavebands yield the low-rank property required for LRMR. LRMR modeling is then applied together with the GoDec algorithm [14] in order to solve the subproblems exploiting the low-rank property of HSI. The subproblems are created by taking subcubes centered at a pixel in the spatial dimension. Thus, if the whole data cube is of size $m \times n \times w$, where w denotes the number of spectral reflectance bands, then the subcubes are of size $b \times b \times w$, with $b < m$ and $b < n$. We define the entries of the subcube's spectral band by the spectral band. The entries of each subcube band–by-band are then organized in lexicographical order to obtain two-dimensional matrices of size $b^2 \times w$. The subcubes

are then processed iteratively, providing local image restoration. We denote further the iteration stepsize by s. More details on the specific LRMR model can be found in [13] and on GoDec in [14].

The main focus of this work is the detailed investigation of the LRMR method with respect to the starting values for its main variable parameters, including rank r and blocksize of the subcubes b, estimation parameter for the percentage of noise corruption p, and stepsize s of the inner iteration. In the following, we denote by r, p, b, and s the initial settings. We apply nonlinear optimization in order to determine the best parameter values for the parameter settings with respect to the peak signal-to-noise ratio (PSNR). We choose the Nelder–Mead simplex algorithm as presented in [15] for the nonlinear optimization. The PSNR is computed by

$$\mathrm{PSNR} = 10 * \log_{10} \frac{\max(C)^2 * m * n * w}{\|C - \tilde{C}\|^2}, \tag{2}$$

where C and $\tilde{C}$ denote the original and denoised data cube, respectively. The sizes of the spatial and spectral dimensions are denoted by m, n, and w. The performance of LRMR is analyzed in terms of computational efficiency and memory with regard to different parameter choices. The computational time taken by the algorithm in MATLAB and Python is compared in order to study the performance of LRMR and differences in the implementation between the two programming languages. The goal is to investigate possible difficulties with parameter optimization.

2.2. Materials

We present the data sets used in this work.

2.2.1. Indian Pines Data Set

The Indian Pines data set covers the Indian Pines test site, which is located in northwestern Indiana. It shows mainly agriculture and forest. The data set consists of 224 spectral reflectance bands and is 145 $\times$ 145 pixels. The wavelength range lies between 0.4 and 2.5 $\times$ 10^{-6} m. The Indian Pines data set is openly available in [16].

Figure 2 shows the scene for six different spectral reflectance bands: 1, 50, 80, 130, 180, and 220. Different bands show different layers of materials' spectral signatures.

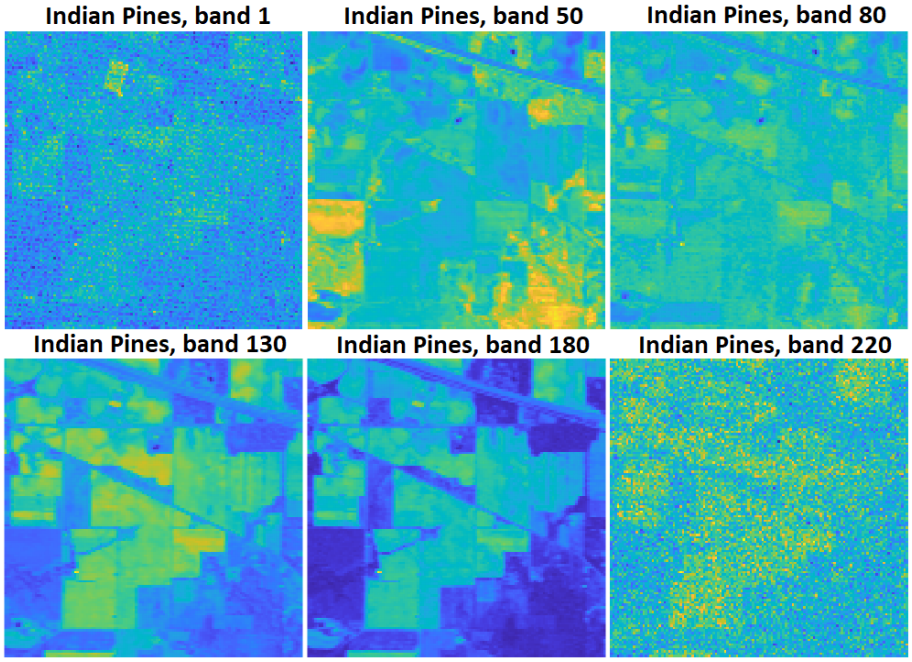

Figure 2. The Indian Pines data set displayed for six different spectral reflectance bands: 1, 50, 80, 130, 180, and 220.

2.2.2. Pavia Centre Data Set

The data set contains scenes from the center of Pavia in northern Italy. It consists of 102 spectral reflectance bands and is 1096 × 1096 pixels. The set is openly available in [17].

Figure 3 shows the scene for three different spectral reflectance bands: 1, 50, and 102.

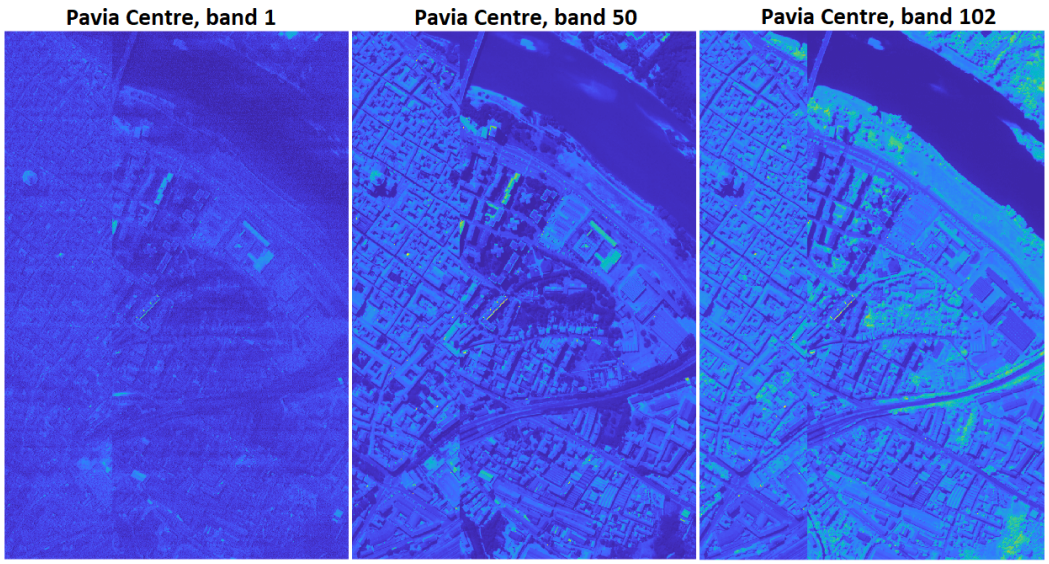

Figure 3. The Pavia Centre data set displayed for spectral reflectance bands 1, 50, and 102.

2.2.3. Pavia University Data Set

The data set contains scenes from the university in Pavia in northern Italy. It consists of 103 spectral reflectance bands and is 610 × 610 pixels. The set is openly available in [17].

Figure 4 shows the scene for three different spectral reflectance bands: 1, 50, and 103.

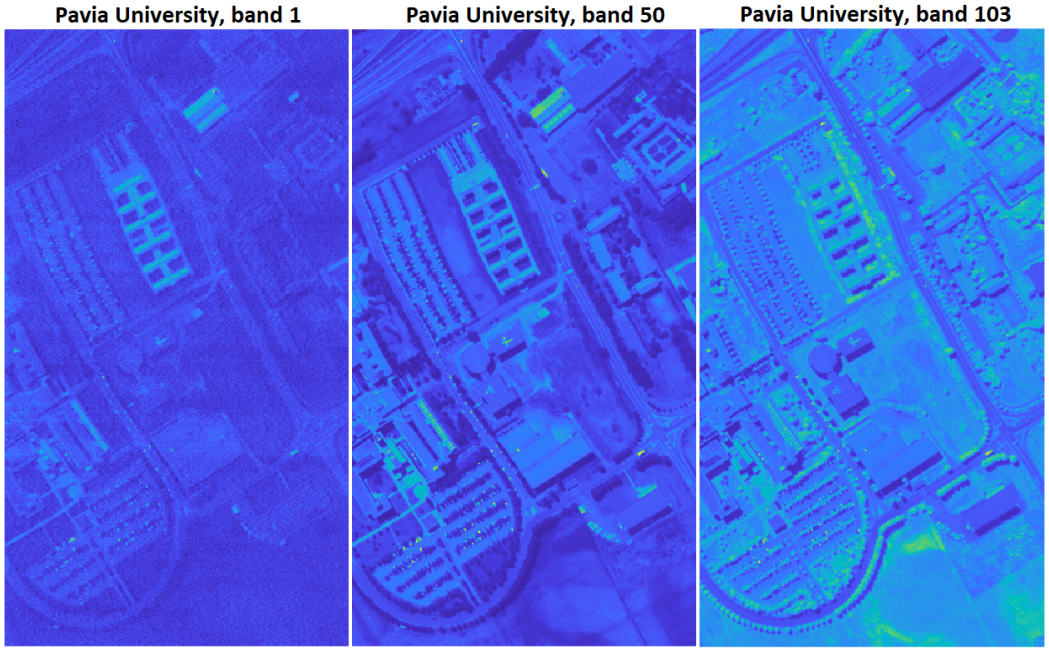

Figure 4. The Pavia University data set displayed for spectral reflectance bands 1, 50, and 103.

3. Results

We analyze how different parameter settings affect the PSNR and computational times. We are able to show that different parameter values and their combinations have an effect on the PSNRs and computational times of the LRMR method. The initial parameter values

are optimized with respect to PSNR and CPU time. The parameter value combinations are studied in detail, and the results are presented visually.

We study the initial parameter value combinations for three integer-valued parameters—r, s, and b—and one real-valued parameter p. We apply nonlinear optimization with respect to the real-valued noise estimation parameter p, which results in an improvement in PSNR. The integer parameters are analyzed on a series of test sets. The optimized values are chosen according to their best PSNR performance. For all analyzed settings and resulting combinations, we show CPU times in MATLAB and Python. The computational experiments are performed on a laptop with an Intel(R) Core(TM) i5-8250U CPU @ 1.60 GHz 1.80 GHz processor and 16.0 GB RAM.

Figure 5 shows the performance of the method for a noise-corrupted waveband of the Indian Pines data set [16] for the parameter settings $r = 7$, $p = 0.15$, $b = 20$, and $s = 8$.

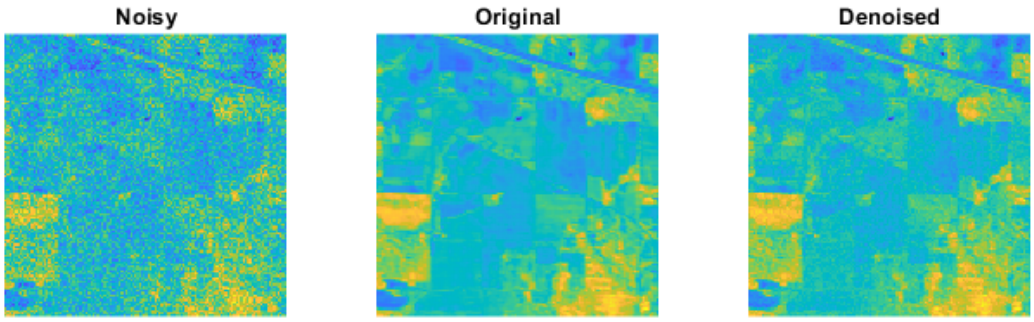

Figure 5. The noise removal performance of LRMR on a noise-corrupted spectral band for the Indian Pines data set. The image is restored efficiently.

3.1. Parameter Analysis for r

The rank parameter r describes the upper bound for the rank of the low-rank matrix describing the noise-free signal of the data cube. The value should be as small as possible while not underestimating the noise intensity of the image data. Small values provide shorter computational times, while larger values provide higher PSNR values. We vary the starting value for the rank parameter r between 1 and 20. The other parameters are kept at $p = 0.15$, $b = 20$, and $s = 8$. Figure 6 shows the PSNR values for different values of r. In general, higher values of r provide larger PSNR values. The analysis shows that a value of $r \geq 4$ is sufficient. The largest gradient step is already taken from $r = 1$ to $r = 2$. The computational times in MATLAB are shown in Figure 7. A larger rank value r results in a larger computational time. The computational tests were computed on the Indian Pines data set.

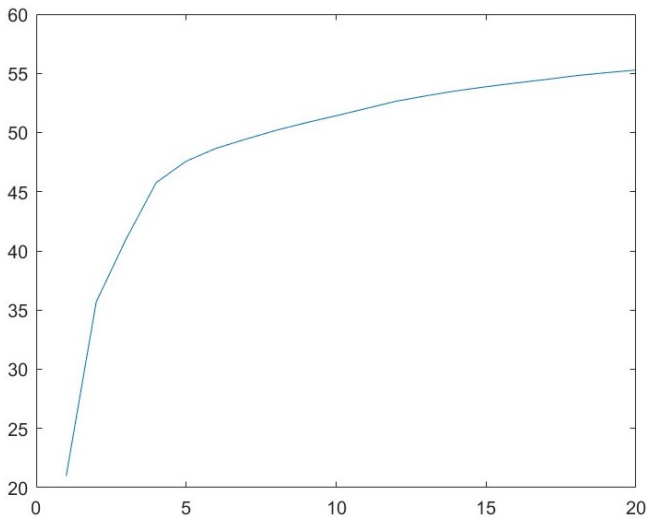

Figure 6. PSNR plot for different values of r.

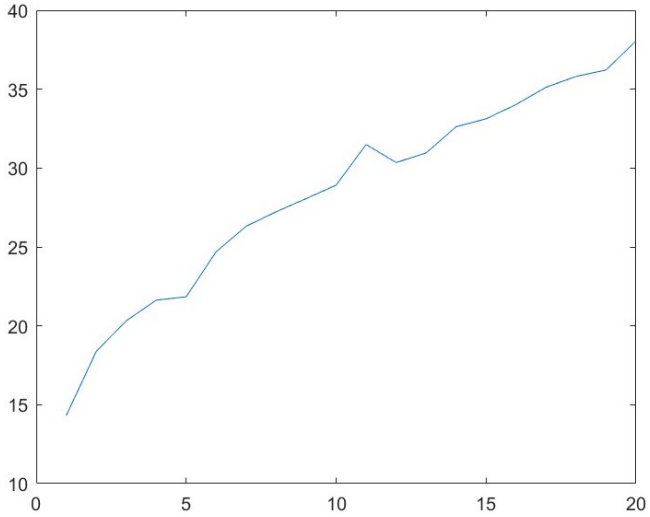

Figure 7. CPU time plot for different values of r.

In the following, we present the PSNR, the gradient of the PSNR denoted as ∇PSNR, and computational times in seconds in MATLAB and Python, denoted as t_{Matlab} and t_{Python}, respectively, in Table 1. Lower rank values are better in terms of computational effort.

Table 1. The PSNRs, gradients of PSNR, and the CPU times in seconds for computing the denoised cubes for different values of r in MATLAB t_{Matlab} and Python t_{Python}.

r	PSNR	∇PSNR	t_{Matlab}	t_{Python}
1	20.99	14.70	14.31	95.15
2	35.69	10.02	18.37	85.57
3	41.02	5.04	20.32	108.98
4	45.77	3.28	21.63	107.42
5	47.58	1.45	21.84	109.01
6	48.67	0.94	24.70	108.22
7	49.45	0.76	26.32	115.23
8	50.19	0.69	27.23	112.48
9	50.83	0.61	28.06	114.21
10	51.42	0.60	28.92	116.79
11	52.03	0.61	31.49	117.49
12	52.65	0.53	30.35	99.18
13	53.10	0.44	30.96	100.97
14	53.52	0.38	32.62	103.88
15	53.87	0.33	33.12	97.06
16	54.19	0.31	34.03	111.33
17	54.49	0.31	35.12	113.62
18	54.81	0.28	35.81	104.27
19	55.06	0.24	36.22	121.06
20	55.28	0.22	38.06	125.18

The computational effort is significantly more efficient in MATLAB. The analysis for Indian Pines suggests choosing $r = 5$. This parameter choice provides a sufficient balance between PSNR and computational time.

3.2. Parameter Analysis for s

The parameter s describes the iteration stepsize for processing the subcubes of LRMR's local image restoration. We vary the parameter value for the stepsize s between 1 and 20. The other parameter values are kept at $r = 7$, $p = 0.15$, and $b = 20$. Figure 8 shows the PSNR values for different values of s. The trend shows that larger values for the stepsize s yield smaller PSNR values. Hence, smaller values for s are preferred, since we target a larger

PSNR. The computational times in MATLAB are shown in Figure 9. The computational times are smaller for larger stepsizes s. However, the CPU times decrease significantly for stepsizes larger than 4. The computational tests were computed on the Indian Pines data set.

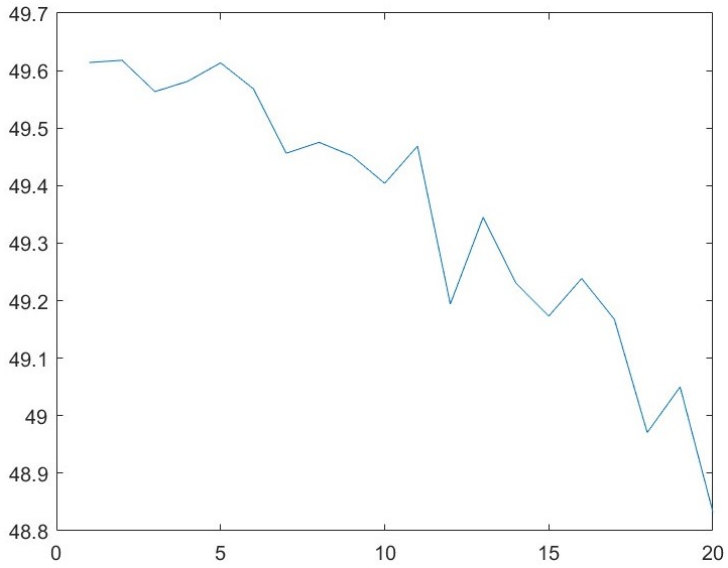

Figure 8. PSNR plot for different values of s.

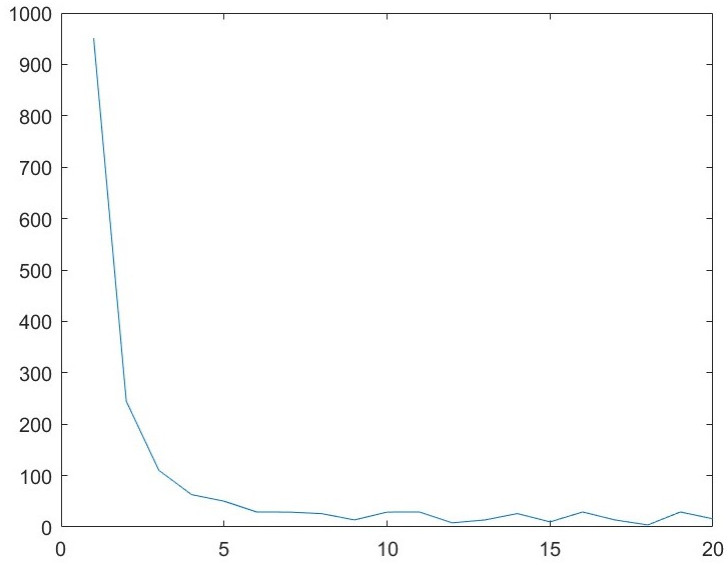

Figure 9. CPU time plot for different values of s.

In the following, we present the computational times in seconds in MATLAB and Python in Table 2. Also, the gradient values for the computational times in MATLAB ∇t_{Matlab} are presented. The result is that a balance between smaller stepsizes that produce better PSNRs and larger stepsizes that are better in terms of computational time and effort has to be achieved. However, note that the difference in PSNR values is not significant, as Figure 8 shows. Figure 9 and the gradient values for the CPU times in Table 2 show clearly that a stepsize $s \geq 4$ provides significant improvement regarding computational times.

Table 2. The PSNR, the CPU times in seconds, and the gradient of CPU times for computing the denoised cubes for different values of s in MATLAB t_{Matlab} as well as the CPU times in Python t_{Python}.

s	PSNR	t_{Matlab}	∇t_{Matlab}	t_{Python}
1	49.61	951.11	−706.38	3793.27
2	49.62	244.72	−420.41	1007.87
3	49.56	110.29	−90.82	481.01
4	49.58	63.09	−30.02	301.01
5	49.61	50.24	−16.93	193.88
6	49.57	29.22	−10.62	178.32
7	49.46	29.00	−1.63	114.31
8	49.48	25.96	−7.63	100.41
9	49.45	13.74	1.54	139.66
10	49.40	29.04	7.71	112.07
11	49.47	29.16	−10.48	111.66
12	49.19	8.08	−7.71	137.42
13	49.34	13.74	9.01	53.73
14	49.23	26.11	−1.90	100.47
15	49.17	9.94	1.53	38.31
16	49.24	29.18	1.85	111.84
17	49.17	13.63	−12.60	52.35
18	48.97	3.98	7.81	178.98
19	49.05	29.25	5.95	114.47
20	48.83	15.88	−13.36	60.94

As for the rank value r, the computational effort is significantly better in MATLAB than in Python.

3.3. Parameter Analysis for r and s

In this parameter analysis, we study how the parameter choices for r and s influence each other. Figure 10 shows the PSNRs for different rank r and stepsize s values, whereas the other parameter values b and p are set as $b = 20$ and $p = 0.15$. The point of intersection is marked and corresponds to $r = 7$ and to values for $s \in [1, 20]$. Again, $r = 7$ is shown to be the optimized parameter choice, whereas values for s show no significant difference in PSNRs. The tests were computed on the Indian Pines data set.

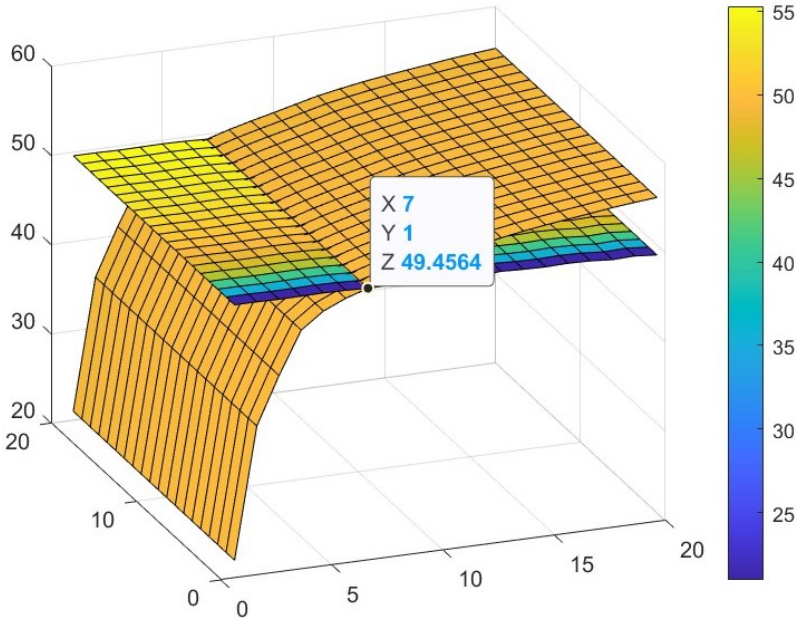

Figure 10. PSNR surface plots for different r and s intersects.

3.4. Parameter Analysis for b

The blocksize of the subcubes for local restoration of LRMR is denoted by b. The parameter choice is significant, since naturally, the size of the subcubes provides accuracy and computational effort to the restoration. We vary the starting value for the blocksize parameter b between 15 and 25. Smaller values for b have not provided successful computations and errors in PSNR. The other parameters are kept as $r = 7$, $p = 0.15$, and $s = 8$. Figure 11 shows the PSNR values for different values of b. The PSNR values are linearly decreasing with respect to increasing blocksize. Figure 12 shows the computational times in seconds in MATLAB. Apart from the large leap at the end and the smaller leap at the beginning, the computational time regarding the change in blocksize seems not to be affected significantly, although a weak trend towards increasing PSNR with increasing blocksize seems to be visible. We present the computational results for the Indian Pines data set.

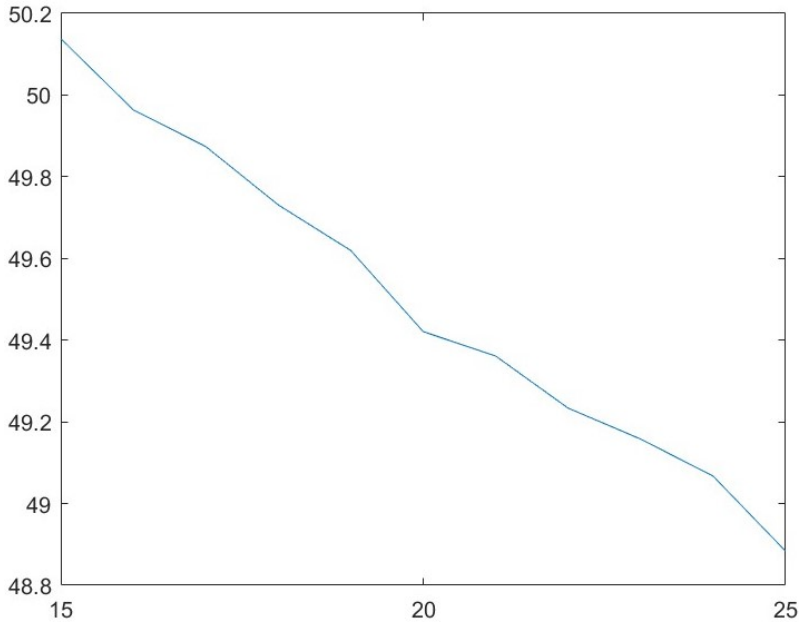

Figure 11. PSNR plot for different values of b.

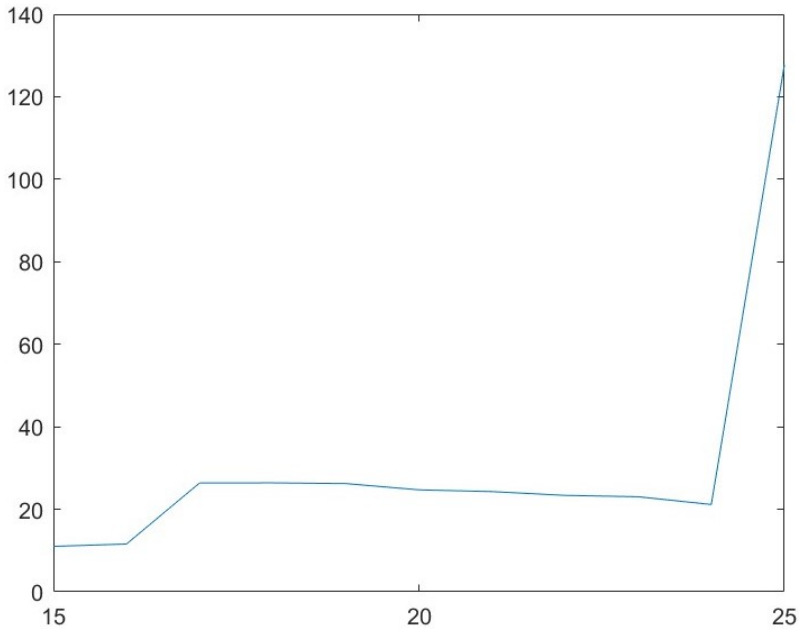

Figure 12. CPU time plot for different values of b.

In the following, we present the computational times in seconds in MATLAB and Python in Table 3.

Table 3. The CPU times in seconds for computing the denoised cubes for different values of b in MATLAB t_{Matlab} and Python t_{Python}.

b	PSNR	t_{Matlab}	t_{Python}
15	50.14	11.01	48.13
16	49.96	11.58	83.56
17	49.87	26.39	102.33
18	49.73	26.41	124.62
19	49.62	26.23	122.86
20	49.42	24.71	127.33
21	49.36	24.29	97.29
22	49.23	23.37	92.63
23	49.16	23.04	102.33
24	49.07	21.16	179.84
25	48.88	127.72	257.69

Again, the computational effort is better in MATLAB R2022a.

3.5. Parameter Analysis for p

The parameter p describes an initial estimation of the amount of noise corruption in the data cube. It is the only real-valued parameter of LRMR's initial parameters. Hence, nonlinear optimization can be applied to determine the optimal initial value for p. We present results for the nonlinear optimization method: the direct search method (i.e., the Nelder–Mead simplex algorithm [15]). We applied the method on the different test sets and varied the starting value for p. The other parameters were set as $r = 7$, $b = 20$, and $s = 8$. We computed the optimization results of this section in MATLAB. In Figure 13, the negative PSNR values of the iteration steps of the nonlinear minimization method, the Nelder–Mead simplex algorithm, computed with respect to p are presented. We present the results for the Indian Pines data set. The starting value $p = 0.15$ was chosen.

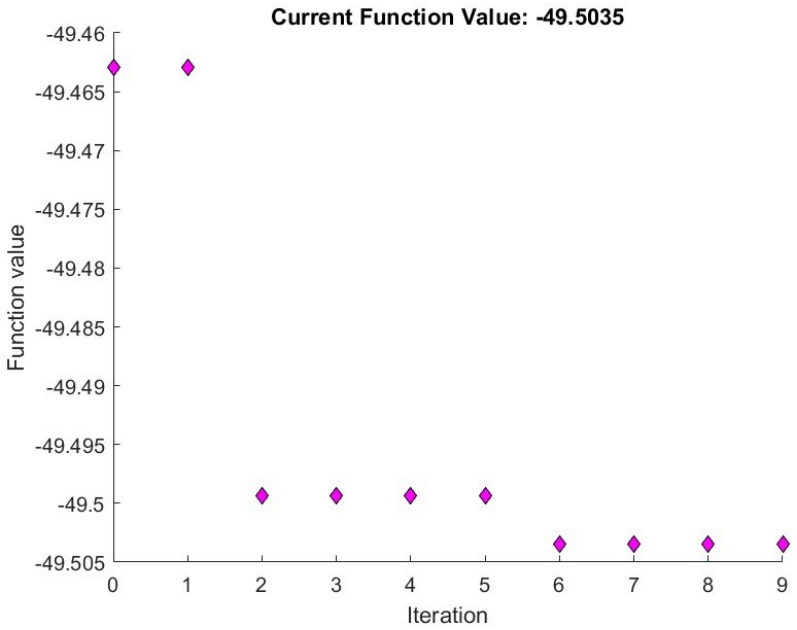

Figure 13. The function values of the iteration steps of the nonlinear optimization Nelder–Mead simplex algorithm with respect to p for the Indian Pines data set and starting value $p = 0.15$.

The results in Figure 13 show convergence towards a local minimum. The value yields an improvement in PSNR. The computational tests show proper performance applying the Nelder–Mead simplex algorithm, including convergence and computational effort. However, the method is searching for a local optimum and, hence, has its limitations. The starting value is crucial. We include a table of final minimization function values and values for p for different starting values (Table 4). The stopping criteria were chosen as follows: the error in the function value is smaller than 10^{-15}, the error in the optimization parameter value is smaller than 10^{-9}, the maximum of function evaluations in one iteration step equals 20, and the overall maximum number of outer iteration steps equals 20.

We propose the choice of $p = 0.15$ for further computations on the data sets used in this work. It is numerically stable and provides sufficient PSNR values. Higher starting values than $1e + 1$ have not provided sufficient convergence.

As a second test, we computed the PSNR and computational times for different values of $p \in [0, 0.2]$ to test different starting values. The Indian Pines test set was used again for a qualitative parameter analysis. Figure 14 shows the PSNR values for different p values. The figure suggests choosing a smaller starting value for p, which coincides with Table 4. Figure 15 shows the CPU times for different p values. The CPU times seem not to be significantly affected in this range of p.

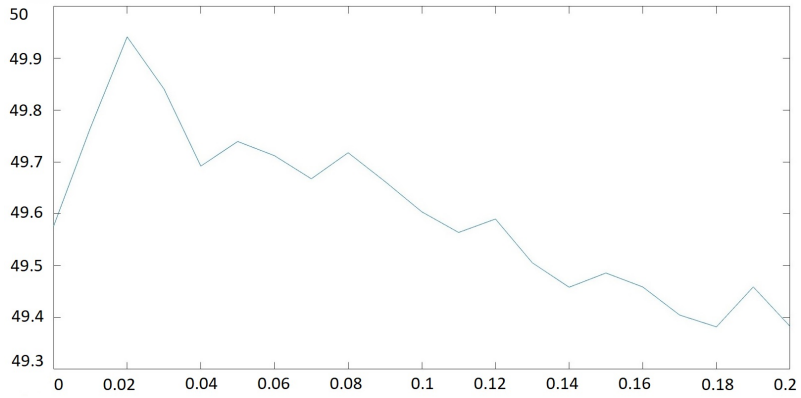

Figure 14. PSNR plot for analyzing different starting values of p.

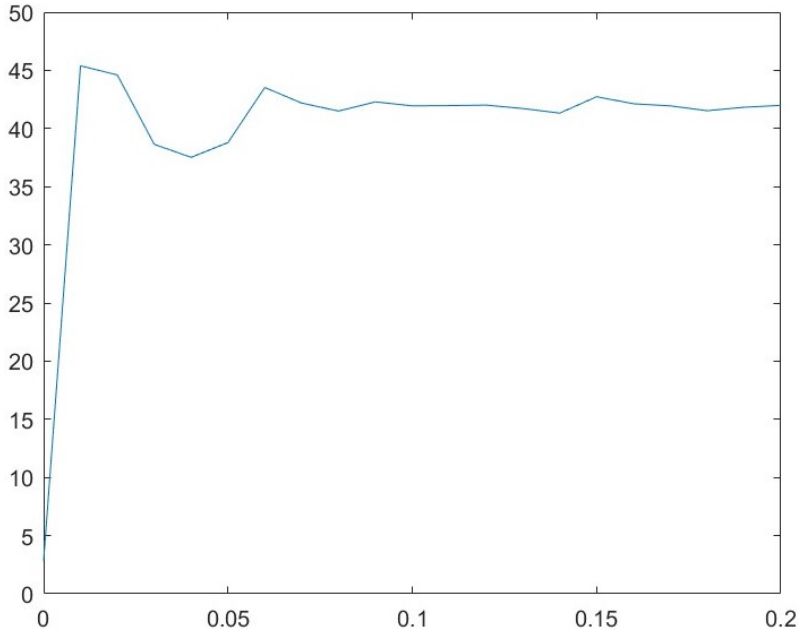

Figure 15. CPU time plot for different values of p.

Table 4. The final function values for the minimization problem (the negative PSNR values) and the corresponding parameter value p_{final} for different starting values p_0. Clearly, local minima are approached. Based on the Indian Pines data set.

p_0	−PSNR	p_{final}
0.015	−49.990256	0.014977
0.05	−49.887782	0.047813
0.15	−49.503483	0.143438
0.5	−49.476512	0.596875
1.	−49.616882	0.975000
1.5	−49.750781	1.818750

3.6. Optimized Parameter Choice

Combining the results of the previous subsections, we present the resulting images for the optimized values. We propose the following settings: $r = 7$, $s = 8$, $b = 20$, and $p = 0.15$.

3.6.1. Indian Pines Data Set

The result of applying LRMR with the optimized parameter setting is shown in Figure 16.

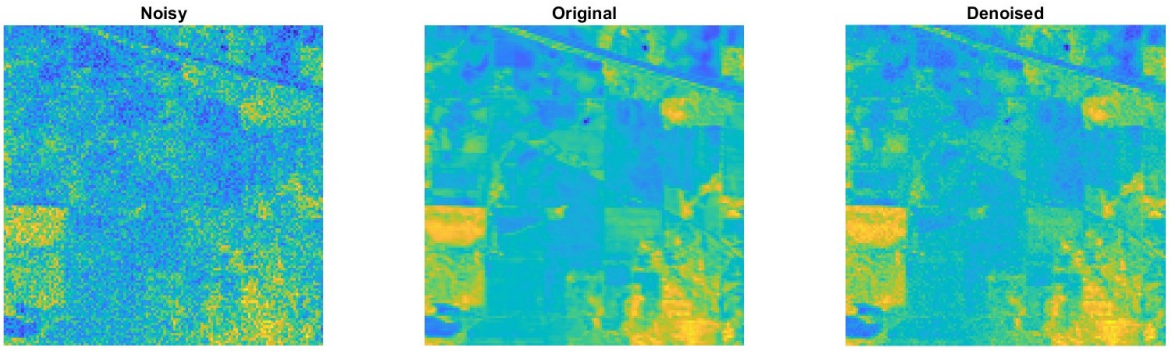

Figure 16. LRMR result computed on the Indian Pines data set for the optimized parameter settings.

3.6.2. Pavia Centre Data Set

The result of applying LRMR with the optimized parameter setting is shown in Figure 17.

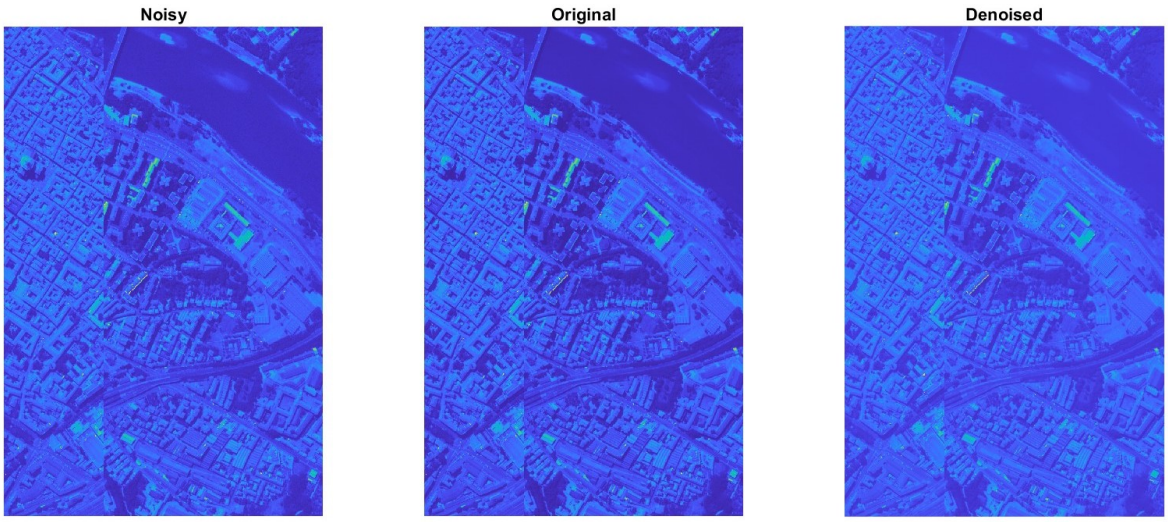

Figure 17. LRMR result computed on the Pavia Centre data set for the optimized parameter settings.

3.6.3. Pavia University Data Set

The result of applying LRMR with the optimized parameter setting is shown in Figure 18.

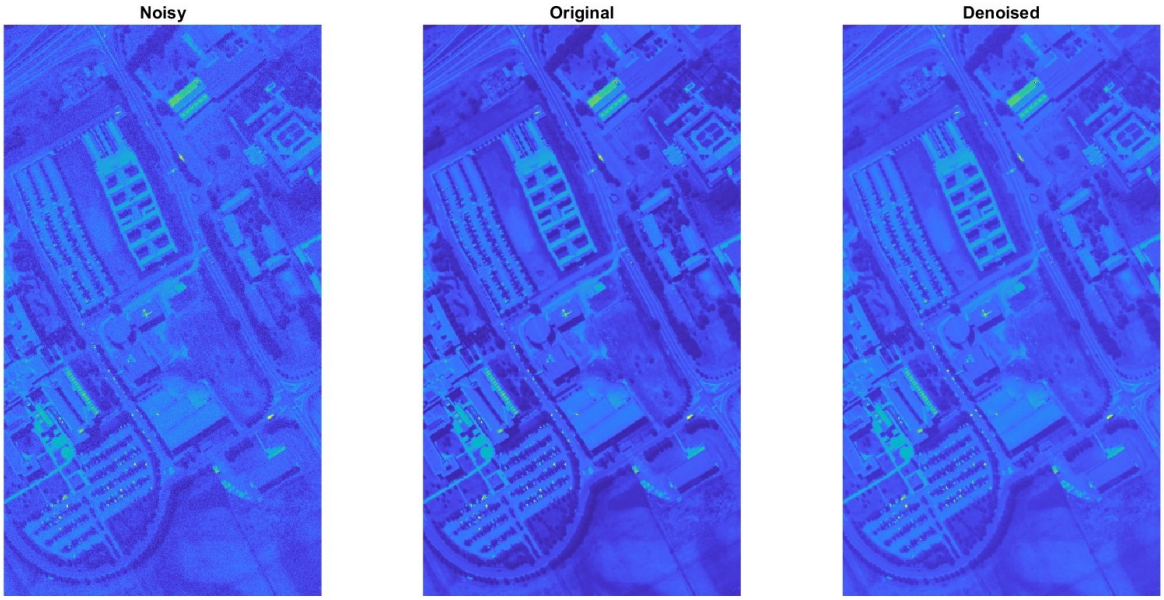

Figure 18. LRMR result computed on the Pavia University data set for the optimized parameter settings.

4. Discussion

We note that parameter tests on other different hyperspectral image data sets may provide different parameter settings. Hence, we advise researchers to perform a parameter study for the initial values in the LRMR method setup for a new test set. A problem-specific parameter configuration is suggested, and it was a goal of this work to highlight and provide a sample analysis as a model for other tests.

We computed the nonlinear optimization and the main analysis results in MATLAB. A comparison between the computational costs of MATLAB and Python was presented for the integer-valued parameters. The overall behavior of the algorithms is the same in MATLAB and Python. However, the computational times are significantly larger in Python. The reason is the efficient storing and processing of matrix–vector operations in MATLAB. The analysis of computational times is important for applications where the data budget is limited or where the data have to be processed in real-time (therefore, for real-time decision making). It was important to show the similar behavior in relative terms for both implementations—MATLAB and Python—and to discuss the implementation issues that may occur. We observed stable behavior for each implementation, and the relative behavior was the same.

Regarding future work, other optimization methods can be applied in order to select the optimized initial values for p. We have applied the Broyden–Fletcher–Goldfarb–Shanno (BFGS) quasi-Newton method [19–22] as an alternative to nonlinear optimization. However, the method always stopped after two iteration steps. Alternatively, the use of a machine learning approach might provide additional parameter optimization results and a more efficient optimization approach to choosing p. This would require access to or the creation of a large data set in order to secure efficient training and testing of the machine learning algorithm. However, this is beyond the scope of this article and is ongoing work of the author.

5. Conclusions

We address parameter analysis and optimization yield improvement of hyperspectral images with regard to the PSNR. We present a detailed analysis with respect to the initial parameters of the LRMR method for different data test sets; however, we focus on the Indian Pines data set. The approach of this work studying parameter optimization for LRMR's initial parameters is new and has not been presented in this way earlier. The analysis provides new insights into the flexibility and boundaries of the method.

Funding: This research was funded by the Regional Council of Central Finland/Council of Tampere Region and European Regional Development Fund as part of the coADDVA—ADDing VAlue by Computing in Manufacturing projects of Jamk University of Applied Sciences.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All data sets are openly available: Indian Pines [16], Pavia Centre [17], and Pavia University [17].

Conflicts of Interest: The author declares no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

HSI	hyperspectral imaging
LRMR	low-rank matrix recovery
GoDec	Go Decomposition
PSNR	peak signal-to-noise ratio
BFGS	Broyden–Fletcher–Goldfarb–Shanno

References

1. Mandelbrot, B.B. A fast fractional Gaussian noise generator. *Water Resour. Res.* **1971**, *7*, 543–553. [CrossRef]
2. Majumdar, A.; Ansari, N.; Aggarwal, H.; Biyani, P. Impulse denoising for hyper-spectral images: A blind compressed sensing approach. *Signal Process.* **2016**, *119*, 136–141. [CrossRef]
3. Shen, H.; Zhang, L. A MAP-based algorithm for destriping and inpainting of remotely sensed images. *IEEE Trans. Geosci. Remote Sens.* **2009**, *47*, 1492–1502. [CrossRef]
4. Rogass, C.; Mielke, C.; Scheffler, D.; Boesche, N.K.; Lausch, A.; Lubitz, C.; Guanter, L. Reduction of uncorrelated striping noise—Applications for hyperspectral pushbroom acquisitions. *Remote Sens.* **2014**, *6*, 11082–11106. [CrossRef]
5. Sun, L.; Cao, Q.; YChen, Y.; Zheng, Y.; Wu, Z. Mixed Noise Removal for Hyperspectral Images Based on Global Tensor Low-Rankness and Nonlocal SVD-Aided Group Sparsity. *IEEE Trans. Geosci. Remote Sens.* **2023**, *61*, 1–17. [CrossRef]
6. Ghamisi, P.; Yokoya, N.; Li, J.; Liao, W.; Liu, S.; Plaza, J.; Rasti, B.; Plaza, A. Advances in hyperspectral image and signal processing: A comprehensive overview of the state of the art. *IEEE Geosci. Remote Sens. Mag.* **2017**, *5*, 37–78. [CrossRef]
7. Rasti, B.; Scheunders, P.; Ghamisi, P.; Licciardi, G.; Chanussot, J. Noise reduction in hyperspectral imagery: Overview and application. *Remote Sens.* **2018**, *10*, 482. [CrossRef]
8. Singh, S.; Suresh B. Role of hyperspectral imaging for precision agriculture monitoring. *ADBU J. Eng. Technol.* **2022**, *11*, 1–5.
9. Calin, M.A.; Calin, A.C.; Nicolae, D.N. Application of airborne and spaceborne hyperspectral imaging techniques for atmospheric research: Past, present, and future. *Appl. Spectrosc. Rev.* **2021**, *56*, 289–323. [CrossRef]
10. Peyghambari, S.; Zhang, Y. Hyperspectral remote sensing in lithological mapping, mineral exploration, and environmental geology: An updated review. *J. Appl. Remote Sens.* **2021**, *15*, 031501. [CrossRef]
11. Qian, S.E. Hyperspectral satellites, evolution, and development history. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2021**, *14*, 7032–7056. [CrossRef]
12. Wolfmayr, M.; Pölönen, I.; Lind, L.; Kašpárek, T.; Penttilä, A.; Kohout, T. Noise reduction in asteroid imaging using a miniaturized spectral imager. *SPIE Sensors Syst. Next-Gener. Satell. XXV* **2021**, *11858*, 121–133.
13. Zhang, H.; He, W.; Zhang, L.; Shen, H.; Yuan, Q. Hyperspectral image restoration using low-rank matrix recovery. *IEEE Trans. Geosci. Remote Sens.* **2013**, *52*, 4729–4743. [CrossRef]
14. Zhou, T.; Tao, D. Godec: Randomized low-rank & sparse matrix decomposition in noisy case. In Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, WA, USA, 28 June–2 July 2011; pp. 33–40.
15. Lagarias, J.C.; Reeds, J.A.; Wright, M.H.; Wright, P.E. Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions. *SIAM J. Optim.* **1998**, *9*, 112–147. [CrossRef]

16. Purdue University Research Repository. Indian Pines Data Set. Available online: https://purr.purdue.edu/publications/1947/1 (accessed on 12 December 2022).
17. Pavia Centre and University Data Sets; Gamba, P. Telecommunications and Remote Sensing Laboratory, Pavia University. Available online https://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes#Pavia_Centre_and_University (accessed on 12 December 2022).
18. Wright, J.; Ganesh, A.; Rao, S.; Peng, Y.; Ma, Y. Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 7–10 December 2009; p. 22.
19. Broyden, C.G. The Convergence of a Class of Double-Rank Minimization Algorithms. *IMA J. Appl. Math.* **1970**, *6*, 76–90. [CrossRef]
20. Fletcher, R. A New Approach to Variable Metric Algorithms. *Comput. J.* **1970**, *13*, 317–322. [CrossRef]
21. Goldfarb, D. A Family of Variable Metric Updates Derived by Variational Means. *Math. Comput.* **1970**, *24*, 23–26. [CrossRef]
22. Shanno, D.F. Conditioning of Quasi-Newton Methods for Function Minimization. *Math. Comput.* **1970**, *24*, 647–656. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.