VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
INSTITUTE OF COMPUTER SCIENCE
DEPARTMENT OF COMPUTATIONAL AND DATA MODELING

Master Thesis

Hyperspectral Image Denoising

Done by:
Ignas Stočkus signature

Supervisor:
dr. Valdas Rapševičius

Vilnius
2020

Contents

Keywords and notations **3**

Abstract **4**

Santrauka **5**

Introduction **6**

1 Related Work and Theoretical Background **7**
1.1 Hyperspectral images . . . 7
1.1.1 Hyperspectral imaging technology . . . 7
1.1.2 Hyperspectral images data cubes . . . 7
1.2 Evaluating Noise in Hyperspectral Images . . . 9
1.2.1 Noise origins . . . 9
1.2.2 Spatial and spectral correlation coefficients . . . 10
1.2.3 Linear regression based noise estimation algorithms . . . 11
1.2.4 Image quality . . . 13
1.3 Denoising hyperspectral images . . . 13
1.3.1 Median filter . . . 13
1.3.2 Non-local means filtering . . . 14
1.3.3 Wavelet transformation based image decomposition . . . 15
1.3.4 Wavelet denoising . . . 17
1.3.5 BM3D . . . 19

2 Experimental Results **23**
2.1 Hyperspectral Image Noise Evaluation . . . 23
2.1.1 Experimental methods and data . . . 23
2.1.2 Correlation coefficients . . . 24
2.1.3 Median filtering results . . . 25
2.1.4 LMLSD and SSDC algorithms . . . 27
2.1.5 Noise evaluation method comparisons . . . 29
2.1.6 2D and 3D Wavelet based filtering . . . 30
2.1.7 Block Matching 3D Filtering . . . 33
2.1.8 Image quality analysis . . . 35
2.1.9 Filtering method comparison . . . 37

Conclusions and Recommendations **39**

Future research work **40**

References **41**

Keywords and notations

HSI - Hyperspectral image
LM - Local mean
LSD - Local noise standard deviation
LMLSD - Local mean local noise standard deviation
SSDC - Spatial spectral direct
BM3D - Block Matching 3D
PSNR - Peak-Signal-To-Noise Ratio

Abstract

In this research paper we have analyzed hyperspectral images, hyperspectral imaging technologies and the noise origins in hyperspectral images. We have analyzed and implemented multiple HSI noise evaluation methods: correlation coefficient based (R_1 and R_2) and linear regression based (LMLSD and SSDC). To compare different noise evaluation methods we have implemented a simple median filter as described in section 1.3.1. For the analysis we have used 5 different noisy hyperspectral crop images captured from an airborne drone. We have concluded that in all noise evaluation methods the noise estimate parameter values were lower after filtering than in the original image, therefore, concluding that our designed noise evaluation methodology can be used to represent noise levels in HSI images. Furthermore, after analyzing the different noise evaluation methods we have concluded that the correlation coefficient based estimate methods and the linear regression based methods all give different results on how effective the filtering is. This was because the initial noise in the images was unknown, so we could not tell which of the noise evaluation methods provided the most accurate representation of the noise. However, we have still concluded that the linear regression based algorithm - LMLSD does not provide accurate results in estimating the overall HSI data cube noise. The main reasons for the inaccurate results were identified as due to the noise not being fully homogeneous. However, the other implemented methods - SSDC, R_1 correlation coefficient and pixel correlation R_2 provided with similar results on determinig which are the most noisy bands and two of them were used to evaluate advanced filtering algorithms. More advanced filtering algorithms were implemented and compared. The best results were given by the BM3D algorithm which outperformed wavelet 2D, 3D and the basic median filter in all noise estimate parameters. More importantly it did not change the original signal LM values and the initial signal remained intact after filtering (full results in table 7). This was not demonstrated by any other method and this method is our recommendation for denoising HSI images. Direct 2D and 3D Wavelet Transformation based filtering was, also, implemented with different threshold functions and different wavelet forms. It was found that there are minimum dependencies on the selected wavelet form but big dependencies on thresholding functions. The best results were achieved with the λ threshold parameter and α threshold function (equations 1.23, 1.22). 3D DWT based filtering performed worse compared to 2D due to images not having homogeneous noise in all bands and, therefore, impacting the thresholding parameter accuracy. As expected, median filtered provided good SSDC LSD results but it had the biggest changes in signal LM values, changing the original useful signal information. Finally, Peak-Signal-To-Noise Ratio (PSNR) parameter (equation 1.12) was calculated to asess image quality for different filtering technologies. The results showed that the worst performing method was the wavelet 3D. Other methods provided simillar results with BM3D having the highest PSNR values in $\lambda\epsilon[550; 900]$ inverval and lowest in $\lambda\epsilon[400; 550]$ and $\lambda\epsilon[900; 1000]$. This means biggest filtering was done the lowest and highest bands, and least filtering was done in mid-range. Therefore, concluding that the BM3D targeted the noisiest channels for filtering leaving the least noisy intact.

Santrauka

Hiperspektrinių vaizdų triukšmo tyrimas

Šiame darbe analizavome hiperspektrinius vaizdus, hiperspektrinias vaizdavimo technologijas ir triukšmo kilmę hiperspektriniuose vaizduose. Mes ištyrėme ir įgyvendinome kelis HSI triukšmo vertinimo metodus: koreliacijos koeficientų suradimo metodikas (R_1 ir R_2) ir tiesinės regresijos metodikas (LMLSD ir SSDC). Analizė atlikta su skirtingais hipersprektriniais vaizdais fotografuotais iš drono. Išnagrinėjus skirtingas triukšmo įvertinimo metodikas, padarėme išvadą, kad tiesinės regresijos pagrindu sukurtas algoritmas LMLSD - neparodė gerų rezultatų įvertinant bendrą HSI triukšmą. Vis dėl to likusieji sukurti metodai: SSDC, koreliacijos koeficientų metodikos R_1 ir R_2 parodė adekvačius rezultatus ir jie gali būti naudojami įvertinant HSI triukšmus. Šie įvertinimo metodai toliau buvo naudoti sukuriant ir lyginant skirtingas triukšmų šalinimo metodikas: BM3D, medianos filtro bei 2D ir 3D wavelet transformacijomis paremtomis metodikomis. Iš šių visų metodų geriausiai pasirodė BM3D algoritmas, kuris sugebėjo ne tik labiausiai pašalinti triukšmus iš vaizdų bet ir mažiausiai iškraipė originalų signalą. Jis yra pateikiamas kaip mūsų rekomenduojamas metodas HSI triukšmų šalinimui įgyvendinti.

Introduction

One of the challenges of today's agrarian culture is to be able to identify diseases of cultivated plants early in their lifecycle [30]. Most often this is done by agronomists who manually examine the crops seeking visual indicators that provide information about a possible disease. However, there are several issues with this verification method:

1. Large crop areas severely slow down this process as manual examination by the agronomists becomes nearly impossible.
2. The signs of the disease must be clearly visually visible. Visual signs usually only come up when the disease is already dominant in the crop and not in the first phase.
3. The diseases begin in very small local areas, so even in a full field manual check the agronomist might not find anything.

The ability to detect the disease early in its phase would allow farmers to take the necessary measures to halt further progress which would reduce the financial damage suffered by farmers and reduce the amount of pesticides used in the treatment of illnesses and, therefore, reduce environmental pollution [23]. Because of these reasons there is interest in society and in the scientists community on improving how to detect early signs of crop diseases. One of the suggested options is hyperspectral photography [25]. Hyperspectral images are obtained by flying with drones above the crop (rape, wheat, rye etc.) and filming them with portable hyperspectral cameras that capture electromagnetic waves from 400 to 1000 nm in length. Specialized software then combines the obtained individual images and creates three-dimensional images (data cubes). These images are used for solving segmentation (classification) problems by classifying healthy crops and crops with certain diseases. By comparing hyperspectral images which contain the same plants we can tell which of them are disease free and which of them have a distinct plant disease. However, before starting the classification these images must be preprocessed. As these captured images are made by a flying drone in nature without any additional crop preparation or controlled ambient conditions they have a high noise level. This prevents further investigation and modeling of the resulting image. The noise levels can be very high on these images, therefore, modelling results without any noise cancelling will be not be correct. Hence, one of the first and biggest challenges becomes the identification and removal of noise in hyperspectral images [28]. Therefore, in this work, we will examine how to detect noise in hyperspectral images and how to distinguish the noisy parts of the image from the noise free, and evaluate the magnitude of the noise. We will, also, try to apply denoising algorithms and compare how effective they were. We will try implement already existing industry standard denoising methods and try determine which of them could be used to evaluate and remove noise in hyperspectral crop images.

Aim of the research – to investigate the noise reduction methods in hyperspectral images.
Research tasks:

1. Create a noise in hyperspectral images evaluation methodology.
2. Implement hyperspectral image noise reduction methods and investigate the dependencies on noise reduction parameters.
3. Use the newly created noise evaluation methodology to compare different denoising algorithms and determine the most efficient one to use.

1 Related Work and Theoretical Background

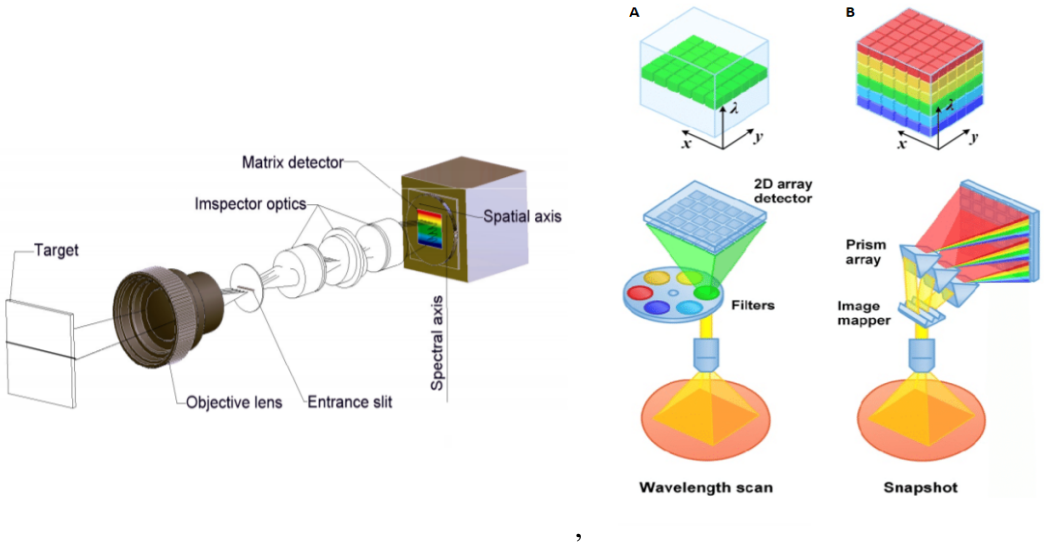
1.1 Hyperspectral images

1.1.1 Hyperspectral imaging technology

Hyperspectral images are created with a hyperspectral camera. Hyperspectral cameras provide a contiguous electromagnetic spectrum ranging from visible over near-infrared to shortwave infrared spectral bands (from 0.3 μm to 2.5 μm) [28]. The final captured spectrum is the consequence of molecular absorption and particle scattering, allowing to distinguish between materials with different characteristics. A hyperspectral image can be captured in more than one way and it depends on the hyperspectral image spectrometer that is being used. Examples of different spectrometers include push broom, filter wheel, liquid crystal tunable filters [15]. One of the most popular approaches that are used today is the push broom spectrometer approach. Push broom spectrometers have a prism component attached which is used to seperate the incoming reflected light into narrow wavelengths. Each width of the wavelengths is equal and represents a captured image band. The wavelength is then recorded on a small light sensitive chip. Besides the prism a push broom spectrometer has a lens and a camera. After a single wavelength is captured the camera is moved, so that the prism can split and record a different wavelength. This is why the spectrometer is called a push broom - the broom pushes the spectrometer components and this way the whole spectrum is scanned and captured. The final image is a combination of all of the overlayed captured wavelengths. A push broom spectrometer working schema can be seen on the left side of figure 1. An anlternative way for capturing these images is using a snapshot spectrometer (referred as "snapshot hyperspectral imaging"). In this scenario an array of prisms is used instead of one prism like in the push broom scenario. This array of diffracting elements is combined with various electronic sensor arrays which can capture single light photons. The idea is that the dispersion elements disperse all of the captured light into all of the wavelengths at the same time. All of the disperesed light is then captured into these sensors giving the full image in all of the bands at the same time. This light capturing method is much faster than the push broom since all of the wavelengths are captured at the same time. But on the downside there is a tradeoff between spatial and spectral resolution for capturing image speed and photon throughoutput. This is because there are a lot of sensor elements and prism elements that have to fit in a small camera area. In a push broom spectrometer there is only one diffracting element and a large pixel capturing element can be used. Both of these spectrometers image capturing techniques can be seen on the right side of figure 1.

1.1.2 Hyperspectral images data cubes

A full hyperspectral image (after all of of the wavelengths are captured and combined) is a three dimensional data cube. Two of the dimensions represent the spatial information in a pixel x and y array and the third dimension represents the spectral information of the captured object. Each of the pixel of the data cube represents a precise measurement of the light that was reflected from the corresponding location on the captured object at a specific wavelength. This means that each pixel that was captured has a one dimensional array of intensity values across all of the bands, so each pixel has an absorption spectrum on the captured wavelengths axis. Now by having this three dimensional data cube the researcher can slice the data in multiple ways (by wavelength, by pixel, by corresponding object location) and use the results for detail spectral analysis. A sample hyperspectral data cube can be seen in figure 2. One important thing to note is that since

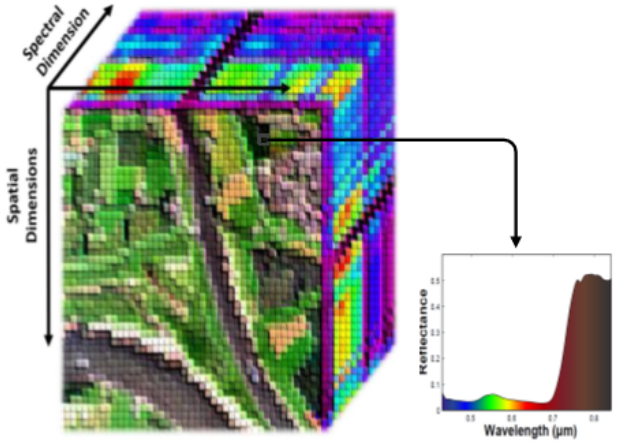


,

Figure 1. Main push broom spectrometer components (left figure) [20]. Two of the most popular hyperspectral images capturing techniques: push broom (right figure A) and snapshot (right figure B) [33]. Push broom - the broom pushes each filter to capture a single wavelength at a time. Snapshot - an array of dispersion elements disperses the captured wavelengths to seperate sensors for a full captured image at all wavelengths.

each spectral band is very narrow (only a few nm), the captured signal for that wavelength is also very weak. This means there is a trade-off between spatial and spectral resolution for the image. Therefore, to increase the resolution hyperspectral images are taken airborne (attaching a hyperspectral camera to a drone or low a flying plane, or other machine). This ensures that more light for each wavelength is being captured and the images have higher spectral resolution.

One more important factor to consider when analyzing a hyperspectral image data is the data size. HSI by default is large in size. For example a scan of a single plant could easily be around a gigabyte in size [23]. For example, during this research the file sizes were ranging from 250 megabytes to 700 megabytes. If we want to analyze the whole spectrum of such images it will take considerably longer for the processing to take than compared to smaller files. An easy solution for this is to not use all of data from all the bands. For example, if we have captured 800 different bands we can only use 400 bands by combining the bands into bigger bins where each bin is the same size and contains information from multiple bands. However, when using only part of the data the researcher might miss some valuable information that was present only in specific wavelengths and the analysis of the image might not be deep enough. A different approach is to use only the wavelengths that the researcher is interested in. If we know exactly what we are looking for in the images (which disease signs are dominant in which band) we can only look into the bands that we are interested in. However, this approach requires additional knowledge of the image background and does not work for preventive scanning tasks. In conclusion, the researcher should consider all of the factors for the data analysis task and decide whether to use all of the data for a more detail analysis and sacrifice speed, or it is enough to combine multiple bands leaving out some information during compression for a more faster analysis (when there are a lot of images to process).



,

Figure 2. A sample hyperspectral image three dimensional data cube schema (left) and a sample reflectance spectra whithin one pixel (right) [15].

1.2 Evaluating Noise in Hyperspectral Images

1.2.1 Noise origins

Since the main object of the study is noise, the following will discuss what the main types of noise are found in hyperspectral images. A noisy image can be defined as [18]:

$$g(x, y) = f(x, y) + n(x, y) \tag{1.1}$$

where x, y - pixel coordinates in the pixel plane of the image, $f(x, y)$ - signal pixel intensity without the noise (clean image where Signal-To-Noise ratio would be 1) and $n(x, y)$ - the intensity of noise of pixel at cordinates x, y.

This means that the total signal itensity of each pixel is determined by the actual intensity of the collected light and the noise factor existing in the channel. Depending on the type of noise the factor $n(x, y)$ varies. There are 3 most common types of noise that are found in hyperspectral images [32]:

1. Impulse noise
2. Gaussian noise
3. Periodic noise

Firstly, let us briefly discuss the impulse noise. The defining characteristic of this noise is that it is defined by a random distribution of placement and intensity of the noise. For impulse noise the noise component $n(x, y)$ will have random black or white pixel intensity values. This means that this noise causation are all the external factors affecting the signal when the images are taken. In a scenaraio where a drone is flying and filming crops this noise can be caused by various factors such as: the wind and drone vibrations during imaging, varying weather conditions, existence of solid particles in the air dispersing reflected light, varying density of crops on the ground, varying terrain causing difference in reflection and others. This noise usually is the dominant type of noise in hyperspectral images and it is the hardest one to account for. Next, we have Gaussian noise. This noise is not accidental and not random - this noise follows the Gaussian dependency and

follows the normal distribution. The probability density function p of a Gaussian noise variable z is given by [7]:

$$p_G(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(z-\mu)^2}{2\sigma^2}} \tag{1.2}$$

where z represents the grey level, μ the mean value and σ the standard deviation [7].

This noise is often caused by electronic and optical imperfections: sensor inaccuracies, imperfect optics of the camera lens, noise of the electronic circuits. Also, this can be sensor noise caused by poor illumination and/or high temperature, and/or transmission. In digital image processing Gaussian noise can be reduced by using a spatial filter but when attempting such procedure an undesirable outcome may result in the blurring of the image edges and details. This is because they usually correspond to the blocked high frequencies. Conventional spatial filtering techniques for noise removal include: mean filtering, median filtering and Gaussian smoothing [3]. Lastly, we have the periodic noise which is characterized by its periodic repetition in the image, so it can be described as as a periodic function [18]:

$$n(x, y) = Asin(U_o x + V_o y) \tag{1.3}$$

where A is the amplitude of noise intensity, U_o and V_o - noise frequency components.

A common source of periodic noise in an image is from electrical or electromechanical interference during the image capturing process [17]. An image affected by periodic noise will look like a repeating pattern has been added on top of the original image. In the frequency domain this type of noise can be seen as discrete spikes. Significant reduction of this noise can be achieved by applying notch filters in the frequency domain [17].

1.2.2 Spatial and spectral correlation coefficients

In order to estimate the magnitude of noise in hyperspectral images we will calculate the correlation coefficients between two adjacent noisy channels e.g between two adjacent wavelengths. We will, also, calculate the correlation coefficients for each image pixel and their neighboring pixels betweeen all image bands (all wavelengths). The correlation coefficient between close bands R_1 will be defined as [32]:

$$R_1 = \frac{\sum_{m=1}^{M}\sum_{n=1}^{N}(A_{mn} - \overline{A})(B_{mn} - \overline{B})}{\sqrt{(\sum_{m=1}^{M}\sum_{n=1}^{N}(A_{mn} - \overline{A})^2)}\sqrt{(\sum_{m=1}^{M}\sum_{n=1}^{N}(B_{mn} - \overline{B})^2)}} \tag{1.4}$$

where M and N are the pixel matrixes of the same dimension of the neighboring bands that R_1 is calculated, and $\overline{A}$ and $\overline{B}$ are the mean values of the of A and B.

Next, we will also find the correlation of each pixel intensity values and its neighboring pixel values between all the bands. The calculation of the correlation coefficient between two neighboring pixel intensity values on all the bands will be defined from the Pearson correlation coefficient as:

$$R_{2i} = \frac{\sum_{i=1}^{N}(Y_{1i} - \overline{Y_1})(Y_{2i} - \overline{Y_2})}{\sqrt{\sum_{i=1}^{N}(Y_{1i} - \overline{Y_1})^2}\sqrt{\sum_{i=1}^{N}(Y_{2i} - \overline{Y_2})^2}} \tag{1.5}$$

where N is the total number of all of the bands, Y_{1i} is the pixel at location A_{mn} intensity value, Y_{2i} is one of the neighboring pixel at location $A_{m+-1n+-1}$ intensity values, and, $\overline{Y_1}$ and $\overline{Y_2}$ are the

means of intensities of the center pixel and one of neighboring pixels across all of the bands. The total correlation coefficient for the pixel will be the average across all of the correlation coefficients calculated for that pixel and its neighbors.

$$R_2 = \frac{\sum_{i=1}^{8} R_{2i}}{N} \tag{1.6}$$

These correlation factors will help us to evaluate the noisiest channels and the noisiest pixels. This methodology is based on the fact that the existing noise in the images does not correlate well between bands such as the main signal does. Signal intensity correlates well between adjacent channels and between adjacent pixels. That means that the correlation coefficients R_1 between close bands and R_2 for all pixels in the same band should be very simmalar to other bands/pixels. Therefore, if there is relatively much noise existing in some channels, adjacent channels R_1 values will be low and the noise component $n(x, y)$ will be much larger than in the noise free channels. The same can be applied to the pixels, the same pixel intensity values in all the wavelengths should have a high correlation coefficient between other neighboring pixels. That means the pixels which will have the lowest correlation coefficients R_2 values will be the most noisy ones and their noise component $n(x, y)$ will be much larger than in the other noise free pixels. After having determined the noisiest channels and pixels the next step would be to apply noise reduction methods for the noisiest channels/pixels and calculate the correlation coefficients again to evaluate the results.

1.2.3 Linear regression based noise estimation algorithms

Besides calculation of the corellation coefficients another type of methods can be used. These type of methods use linear regression based models to evaluate the noise in each seperate band and they also give a noise estimate for the entire image. The correlation coefficient method was not capable of estimating the whole hyperspectral image. We will examine and later experimentally test two linear reggression based methods: LMLSD (Local Mean Local Noise Standard Deviation) and SSDC (Spatial Spectral Direct Corellation). Both of these algorihtms were developed in the 1990's and both of these algorithms have also been improved since they were initially created. The new versions of these algorithms are called the RLSD and HRDSDC. First of all, we will understand and test out the original versions, so using the newer one is out of scope of this research. Firstly, let us examine the LMLSD algorithm. This method assumes that the whole hyperspectral image is made up of many homogeneous blocks. This means that the photographed image should contain the same photographed objects, materials, substances etc. Having different materials in the image would drastically reduce the accuracy of this method. The first step is to divide the entire image into non-overlapping blocks where in each block we would have $w * w$ pixels (w - block spatial dimension). The LMLSD only takes into account the spatial correlations of pixels in each band without the spectral correlations. This means that the blocks are 2D and each will contain $w * w$ number of pixels in each band. Also, each band will have the same number of blocks, since each band has the same spatial dimensions. Now the local mean (LM) of the signal in each band can be calculated as [16]:

$$LM = \frac{1}{w^2} \sum_{i=1}^{w} \sum_{j=1}^{w} x_{i,j,k} \tag{1.7}$$

where w is the block dimension, $x_{i,j,k}$ - pixel value in band k. Now it is assumed that the deviation from this local mean (LM) is due to noise, therefore, the noise is estimated by the local noise

standard deviation (LSD) and is estimated for each block as [16]:

$$LSD = \sqrt{\frac{1}{w^2 - 1} \sum_{i=1}^{w} \sum_{j=1}^{w} (x_{i,j,k} - LM)^2} \tag{1.8}$$

where LSD is considered to be as the noise standard deviation of the block in band k and LM is the signal mean for that block.

When each block has the LSD parameter calculated the minimum and maximum values of the standard deviations are found. In this interval bins of equal width are created. The number can be from 1 (the mean value of standard deviation) up to N depending on the images nature. The number of blocks falling into each bin are counted and the bin with the highest number of blocks is going to represent the noise standard deviation for the entire image independent of bands noise deviations. The actual noise standard deviation mean value in the bin is considered as the estimated noise of the HSI [16]. Depending on the number of bins the results will vary. If the hyperspectral image is not homogeneous than the estimated noise value can vary a lot from the mean of standard deviation. Also, if the number of bins is too high than the differences between the means in neighboring bins can be very small. This means that an expected interval to represent the image noise standard deviation is going to be split into multiple ones with small differences. Now the largest bin can become any other bin which has a few more LSD block values and it will be incorrectly considered as the noise estimate for the image.

An alternative method which solves the previously mentioned problem is the SSDC algorithm method. This method is more advanced than the first one since it uses spectral (between band) and spatial (within band) correlations. This method uses these correlations to de-correlate the image data with linear reggression. The estimated noise is assumed to be the unexplained residuals between the real values and the modeled values with linear reggression. As with the LMLSD method the image is divided into $w * w$ blocks in each band. Each pixel in the block at (i, j) coordinates in band k can be predicted by the pixels spatial and spectral neighbors. Now the residual is calculated as [16]:

$$r_{i,j,k} = x_{i,j,k} - \hat{x_{i,j,k}} \tag{1.9}$$

where $\hat{x_{i,j,k}}$ is the predicted value of $x_{i,j,k}$ with linear reggression and $\hat{x_{i,j,k}}$ is computed as [16]:

$$\hat{x_{i,j,k}} = a + bx_{i,j,k-1} + cx_{i,j,k+1} + dx_{p,k} \tag{1.10}$$

where p expresses the spatial adjacent pixels of $x_{i,j,k}$ and a, b, c, d are the coefficients determined by multiple linear reggression. The same noise standard deviation LSD of a block can be calculated as the sum of all residuals in the block and would be equal to [16]:

$$LSD = \sqrt{\frac{1}{w^2 - 4} \sum_{i=1}^{w} \sum_{j=1}^{w} r_{i,j,k}^2} \tag{1.11}$$

where the mean value of all the blocks in band k is considered as the band noise estimate value. For the whole image - the mean value of all LSD values. In this formal equation 1.10 for LSD calculation only one band above $(k + 1)$, one band below $(k - 1)$ and one spatial neighbor for multiple linear reggression is used. However, the p can be extended for up to 8 adjacent pixel neighbors for more precise calculations, since correlation must exist between all pixels in the same

band. If more neighbors of $x_{i,j,k}$ are going to be used than more coefficients in the multiple linear reggression have to be determined. At first glance this method is better than the LMLSD approach, since it also includes spectral bands which also have correlation with the signal between bands. Also, it solves the bin problem as discussed above for images that are not homogeneous. We will test these methods experimentally.

1.2.4 Image quality

Previously mentioned models can be implemented to determine and evalute noise in a HSI image. However, we do not have a parameter which would compare different filtering techniques by the overall image quality. Visual comparisons are not possible due to the nature of HSI data (data is captured and combined into data cubes from multiple wavelengths). For example, median filtering can be very effective in removing noise as determined by our methods but a lot of useful information during median filtering is, also, lost. This is why there is a need for more advanced techniques. Therefore, we will introduce another parameter to compare different filtering techniques. This parameter will be Peak Signal-To-Noise ratio or PSNR. PSNR is most commonly used to measure the quality of reconstruction of lossy compression codecs (e.g., for image compression) and, also, in comparing digital image filtering methods. Also, it is commonly found on similar research work in HSI. PSNR is defined in dB as [29]:

$$PSNR = 10log_{10}(\frac{max^2(x)}{MSE(x,\dot{x})}) \tag{1.12}$$

where x is a vectorized clean image of length N, $\dot{x}$ is the vectorized denoised image and MSE (Mean Squared Error) is defined as:

$$MSE(x,\dot{x}) = \frac{1}{N}\sum_{i=0}^{N-1}(x(i) - \dot{x(i)}))^2 \tag{1.13}$$

The only issue with the above definition is that in our research we do not have a clean image baseline as the images are captured from real world scenarios. This means we will have to compare the noisy image and the denoised image. In this case we will still get good sense of the image quality for filtering techniques comparisons (as the baseline is the same) but the exact PSNR values could not be used in giving exact measureable values as we are calculating MSE with noise which brings extra devations. To perform an exact, measureable filtering image quality comparisons we would need to use a noisy free image, introduce artificial noise to it and then denoise it with all the techniques. However, this can be done in an extension of this work as the goal is to try to find the best denoising technique for real world HSI images with noise.

1.3 Denoising hyperspectral images

1.3.1 Median filter

The first technique that is going to be used for denoising hyperspectral images is a median filter. This approach is widely used in normal 2D pixel images noise reduction and it can also be applied to a hyperspectral data cube. In a data cube the median filter must be applied to each wavelength in a x and y pixel array. This will reduce the salt and pepper noise (impulse noise) in that wavelength. After applying the filter we can construct a new datacube with the same spatial dimensions. The

spectral dimension values will contain the new denoised, median values after the filtering. The median filter considers each pixel in the data cube band one by one and looks at its nearby neighbors to decide whether it is representative of its surroundings. Then it replaces the center pixel value with the median of those neighboring values. The median is calculated by first sorting all the pixel values from the surrounding neighborhood into numerical order and then replacing the center pixel with the middle pixel value. We will be implementing a 3 x 3 median filter. This means we will consider only the closest neighboring pixel values (in the general case - 8 neighboring pixel values) in the median calculation. We will be using the median filter to measure the noises levels of our hyperspectral images and to test out our noise evaluation methods as defined previously. After having analyzed if our evaluation methods work we will use the results to evaluate more advanced filtering techniques which will be discussed next.

1.3.2 Non-local means filtering

While the previously described median filter would be effective in denoising the HSI cube, it is also effective in clearing out the signal values, therefore, loosing information and details. In image denoising there are other, more advanced algorithms that are used. One of the most popular one is the non-local means denoising algorithm. Unlike local mean filter, which takes the mean value of a group of pixels surrounding a target pixel to smooth the image, non-local means filtering takes a mean of all pixels in the image, weighted by how similar these pixels are to the target pixel. This results in much greater post-filtering clarity, and less loss of detail in the image compared with local mean algorithms [4]. With this approach a band pixel array would be divided into patches or blocks of $w * w$ pixels. A scan of the entire band would begin to search for similar patches as the current one. After the search is completed denoising would take place. Denoising is done by computing the average pixel values of these most resembling pixels. For a formal mathemical equation the filtering can be defined as [6]:

$$u(p) = \frac{1}{C(p)} \int_{\Omega} v(q) f(p,q) dq \tag{1.14}$$

where Ω is the area of the pixel array in band k, p and q are the two points within the image band. In a patch scenario these would be two distinct patches in size $w * w$. The filtered value at the location p would be $u(p)$. $V(q)$ is the unfiltered value of the image in location q and $f(p,q)$ would be the weighting function based on the pixel patch distance and similarities. $C(p)$ is the normalizing factor given by:

$$C(p) = \int_{\Omega} f(p,q) dq \tag{1.15}$$

The purpose of the weighting function is to determine how closely related the image patches (pixels) are in the locations p and q and it can take various forms. The most used one is the Gaussian weighting function which sets up a normal distribution of noise with a mean and a standard deviation. This form assumes that the noise found in the image follows the normal distribution arround the signal mean values. Since in our HSI images the noise origin and variance is unknown it is safe to assume that this noise type should be accounted for. When the noise origins are known a more precise weighting function can be used. In our case we will be using the Gaussian weighting function, which is defined as [5]:

$$f(p,q) = e^{-\frac{|B(q)-B(p)|^2}{h^2}} \tag{1.16}$$

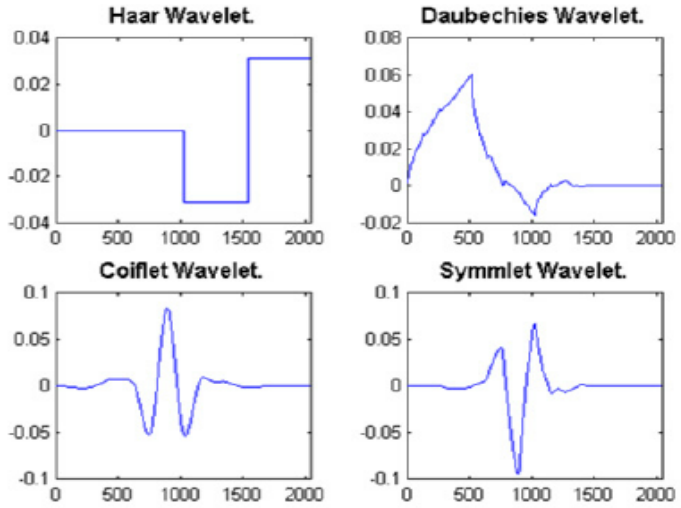
where h is the filtering factor (standard deviation of signal in the image or i.e noise) and $B(p)$ is the local mean value of the image surrounding the point p. In a real implementation some corner cutting must be made. For example, the search for similar pixels. The search would need be in a local neighborhood and not in the entire band as described with the integral of Ω (search size is found to be up to 35 * 35 neighboring pixels in some sources). This is because a search through entire band for each of the pixel patch would take a very long time to complete, especially, for a HSI data cube with 200 bands. Still as compared to the local median filtering where only the 1 x 1 neighbors are considered and without weight function, this filtering would provide with more precise results and with less lost information. Next, we will look at some examples of the non-local filtering techniques - BM3D and BM4D filtering methods. These will be discussed after wavelet transformation based filtering and thresholding methods since both of these concepts are found in BM3D and BM4D.

1.3.3 Wavelet transformation based image decomposition

Another common hyperspectral image filtering technique is by using 2D and 3D wavelet transformations. This approach combines the wavelet transformations with thresholding. Wavelet analysis is considered a state-of-the-art technique in signal processing; it transforms the signal into a scale-time representation (scalogram) with high resolution, preserving the temporal characteristics of the signal. It has a key advantage compared to another popular spectral analysis approach - the Fourier transformations. It captures both frequency and location (location in time) information. Wavelets emerged in the 1960's allowing to represent signal both in time and frequency domain simultaneously. This way, not only both time and frequency information is available, it is also possible to store signal efficiently [12]. Mathematically, wavelets are nothing but functions that divide the original signal into different frequency components and study each component. The basis functions of Wavelet Transform (WT) are scaled according to the frequency. There are different small waves (also known as mother wavelets) that can be used for the implementation of WT [9]. Some most popular forms of wavelets include Dubachies, Haar, Symlet, Coiflet, Mexican Hat, Morlet. Some of these wavelet forms can be seen in figure 3. Each of them comes from different wavelet families and includes different properties [9]. We decide which wavelet to apply in different cases and scenarios depending on the requirements of the application. There are two types of WT defined as Continues Wavelet Transform (CWT) and Discrete Wavelet Transform (DWT). In the CWT, the analyzing function is a wavelet ψ. The CWT compares the signal to shifted and compressed or stretched versions of a wavelet. Stretching or compressing a function is collectively referred to as dilation or scaling and corresponds to the physical notion of scale. By comparing the signal to the wavelet at various scales and positions, you obtain a function of two variables [1]. The common form for CWT can be written as:

$$C(a,b;f(t),\psi(t)) = \int_{-\infty}^{\infty} f(t)\frac{1}{a}\psi^*(\frac{t-b}{a})dt \tag{1.17}$$

where ψ^* denotes the complex conjugate of the analysing function (wavelet), $f(t)$ is the original signal, a is a scale parameter and b is a position in time. By continuously varying the values of the scale parameter, a, and the position parameter, b, you obtain the CWT coefficients $C(a,b)$. Another type of WT that exists is called the DWT which is an implementation of WT using mutually orthogonal set of wavelets defined by carefully chosen scaling and translation parameters (a and b). This leads to a very simple and efficient iterative scheme for doing the transformation [26].



,

Figure 3. Some of the wavelet families. [8].

The Discrete Wavelet Transform (DWT), transforms a finite energy signal to the scaled frequency domain. The one dimensional orthogonal DWT can be written as [29]:

$$x(t) = \sum_{k \epsilon Z} u_{j_{0,k}} \phi_{j0,k}(t) + \sum_{j=-\infty}^{j_0} \sum_{k \epsilon Z} \omega_{j,k} \psi_{j,k}(t) \tag{1.18}$$

where

$$\phi_{j,k}(t) = 2^j \phi(2^j t - k), \psi_{j,k}(t) = 2^{j/2} \psi(2^j t - k) \tag{1.19}$$

are the scaling and wavelet functions respectively and the inner products $u_{j,k} = \langle x, \phi_{j,k} \rangle$ and $\omega_{j,k} = \langle x, \psi_{j,k} \rangle$ are the scaling and wavelet coefficients, respectively.

In DWT the computation is performed only on a set of wavelets. This means that compared to CWT it is much faster in terms of computing time. However, a disadvantage of this faster computational time is that some information is lost during the conversion. This can be of high importance when trying to analyze and find hidden information in datasets. For hyperspectral data cubes which are very big in size the only option is DWT when we have limited computing power. This why only DWT will be used in future experimental analysis. Also, DWT can be seperately applied for each data cube dimension. This is reffered as 2D, 3D wavelet filtering. This is done by applying DWT to each dimensions seperately. For example, applying a 2D-DWT on 2D images means that 1D-DWT is applied on the data rows and afterwards on the data columns. The result are four sub-images which are called HH, HL, LH, LL. Analysis then continues on these sub-images. A 3D-DWT is when the 1D wavelet trasformation is applied on each data cube axis (x, y and spectral axis). The result would be eight sub-images numbered from HHH to LLL. Thresholding is then applied to these sub-images seperately. Further levels of decomposition could be applied to get other levels of decomposition, increase computing time and apply thresholding at a more granular level. The schemas for both decompositions can be seen in figure 4. We will implement 2D and 3D DWT wavelet based filtering techniques. In one case we will pass a 3D data cube with pixel and spectral information and process each axis seperately, in other case we will pass 2D images from each of the bands seperately.

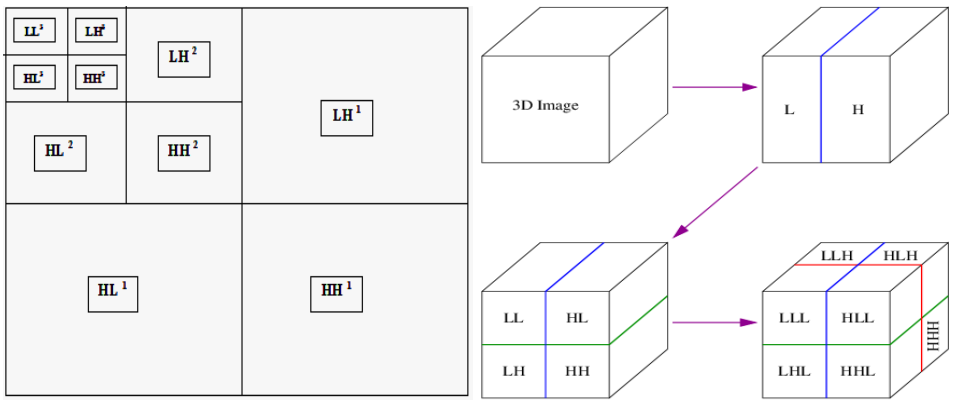


Figure 4. On the left - The 2D Discrete Wavelet Transform (DWT) with 3 Level Decomposition [27]. On the right - The 3D Discrete Haar Wavelet Transform (DWT) sample schema [19].

1.3.4 Wavelet denoising

Wavelet transformation based denoising algorithm consist of three main iterations. The scheme can be seen in figure 5. In order to denoise an image or signal we first need to decompose it with

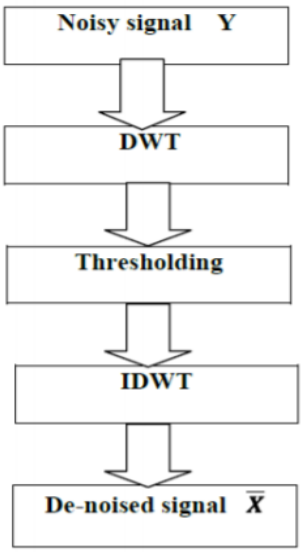


Figure 5. The main principal schema of DWT based denoising [12].

the wavelet transformation. This allows us to get a group of coefficients at different frequency levels. Afterwards, an analysis of the coefficients should be performed in order to determine the best thresholding values. By finding and applying the best thresholding technique we can eliminate unwanted data from our data sets and loose the least amount of information in the process. After thresholding is applied the signal or image is reconstructed by applying a reverse DWT. The most important step in the denoising algorithm is choosing the most effective thresholding technique. Most commonly four different estimators of threshold value are used given as heasure, minimax, rigsure and sqtwolog. Minimax and SURE threshold selection rules are more convenient when small details of the signal lie near the noise range [2]. But other estimators could be used which we will look at later. Thresholding can, also, be done in two different methods: soft and hard thresholding. Hard thresholding applies a strict barrier for the elements - if their absolute values

are lower than the barrier (hard threshold limit) they are set to zero. Soft thresholding does not apply this strict barrier and applies a softened version of it - if the element absolute values are over the thresholding limit they are not set to zero but set to the barrier value. Soft thresholding requires more computations but gives better denoising performance [2]. Both versions mathematical expressions can be seen in equations 1.20 and 1.21 and an example is given in figure 6 for a line space signal.

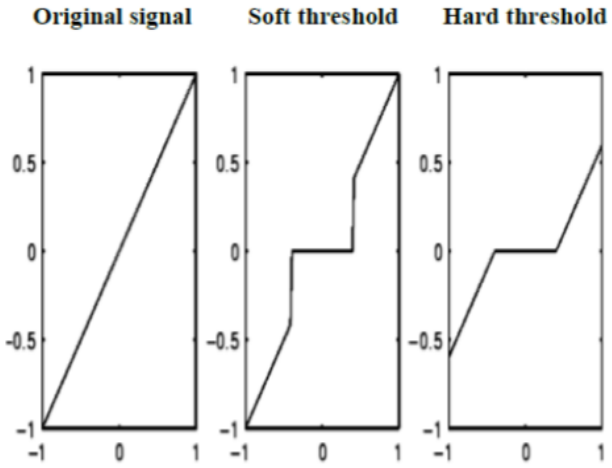


Figure 6. Soft and hard thresholding applied on a line space signal [2].

$$T_{hard} = \begin{cases} x & |x| >= threshold \\ 0 & |x| < threshold \end{cases} \tag{1.20}$$

$$T_{soft} = \begin{cases} Sign(x)(x - threshold) & x >= threshold \\ 0 & -threshold <= x < threshold \\ Sign(x)(x + threshold) & x < -threshold \end{cases} \tag{1.21}$$

For our research we will be using few well known thresholding functions. This is to examine the possible dependencies for the same wavelet filtering technique only with different thresholding functions. Firstly, we will use the following soft thresholding function, as it is defined as the most efficient one providing one of the better denoising results [14]:

$$\alpha = Sign(w_y)max(0, |w_y| - \frac{\lambda}{2}) \tag{1.22}$$

where the w_y and α (the elements of vectors w_y and α) are the wavelet coefficients of y and x, respectively. In other words, w_y is the full vector that is obtained after applying the 2D or 3D DWT. It has all of subbands as seen in figure 4. Each element of each subband is processed and the above thresholding is applied to each element. The outcome of each element is a new element $\alpha_{y,x}$. As it can be seen from equation 1.22 a calculated value of λ is used. It represents the thresholding barrier and it is the regularization parameter. It will impact the thresholding size. This value can be calculated as [29]:

$$\lambda = \sqrt{2log(N)}\sigma_n \tag{1.23}$$

where N is the number of pixels in the image and σ_n is the noice variance. Since we do not know how much initial noice is in the images we will estimate it as it was given by Donoho and Johnstone [14]:

$$\sigma_n = \frac{median(W_y(HHorHHH))}{0.6745} \tag{1.24}$$

where $W_y(HHorHHH)$ are the smallest subsets of 2D and 3D wavelet coefficients for subbands HH and HHH, respectively. Secondly, we will use the universal thershold value λ with a different soft thresholding implementation. This λ was introduced by Donoho and Johnstone and they proved that the λ estimation for thresholding together with soft thresholding as defined in equation 1.21 is enough to satisfy the requirements of most applications [24]. So, we will combine λ with soft thresholding defined in equation 1.21 to perform the filtering. Lastly, we will use the Bayes Shrink thresholding function which is, also, commonly used and provides good denoising results. The Bayes Shrink minimizes the Bayes' risk estimator function assuming a generalized Gaussian form [10]. The threshold is given by:

$$\gamma_{Bayes} = \frac{\sigma_n}{\sigma_x} \tag{1.25}$$

where σ_n is the noice variance and σ_x is the estimated signal standard deviation which is given by [10]:

$$\sigma_x = \sqrt{max(w_y^2 - \sigma_n^2, 0)} \tag{1.26}$$

As it can be seen we will be using 3 different thresholding instances and comparing the results for the same wavelet filtering. Hence, our implementation of the wavelet denoising schema as defined in figure 5 becomes:

1. Step 1 - Apply Direct Wavelet Transformation for a 2D data cube band or apply DWT for a 3D data cube to gain the wavelet coefficient decomposed subbands.

2. Step 2 - Estimate the noise in the HH or HHH subband and calculate λ or γ_{Bayes} as given by 1.23, 1.26 for each band in a 2D DWT scenario or the whole hyperspectral cube in a 3D DWT scenario.

3. Step 3 - Apply the thresholding algorithm for the decomposed subbands as defined in each of the 3 different thresholding methods with the previously calculated thresholding barriers.

4. Step 4 - Apply the Inverted Direct Wavelet Transformation to retain the data set. Combine all bands into a single data cube.

5. Step 5 - Compare the filtered data cube with the original data cube with the noise estimation algorithms created in 1.2.

1.3.5 BM3D

The previously mentioned proposed method on non-local means filtering was improved and a new image denoising industry standard method was introduced - the block matching 3D (BM3D). BM3D is generally considered to achieve the best performance in image denoising and it is being applied to hyperspectral images. The non-local means method denoises similar patches but only by

performing a patch average which amounts to a 1D filter in the 3D block. The 3D filter in BM3D is performed on the three dimensions simultaneously [22]. In other words, instead of filtering in the local neighborhood, similar image patches are recognized. Different patches have smaller correlation of noise than local neighborhoods giving better results than the local neighborhood-based filtering schemes [13]. We will briefly summarize the BM3D filtering algorithm. Consider an image region X centered arround a particular pixel $n(x, y)$ in a 2D image. Sample region can be seen in figure 7. The full BM3D algorithm is implemented in two phases. First phase steps are defined below.

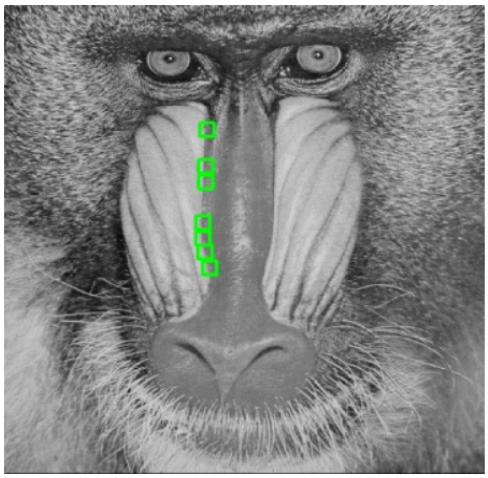
 ,

Figure 7. A sample image with single set of similar blocks recognized in the BM3D algorithm [13].

1. Step 1 - Search for regions X' which are similar to a source region X. These regions are then compared in a transformation domain (commonly Fourier DFT transformations such as Hadamard or Walsh transformations are used). The transformed coefficients are compared with a designed threshold values and those below are set to 0.

2. Step 2 - Similar patches are put on top of each other in a 3D matrix. A 3D transformation on the matrix is applied to obtain new transformation coefficients. Hard or soft thresholding is performed on the coefficients (this is refferred to as level 1 filtering on similar patches). Inverse 3D transformation is applied to obtain the filtered blocks.

3. Step 3 - The filtered blocks are returned and placed back at the image.

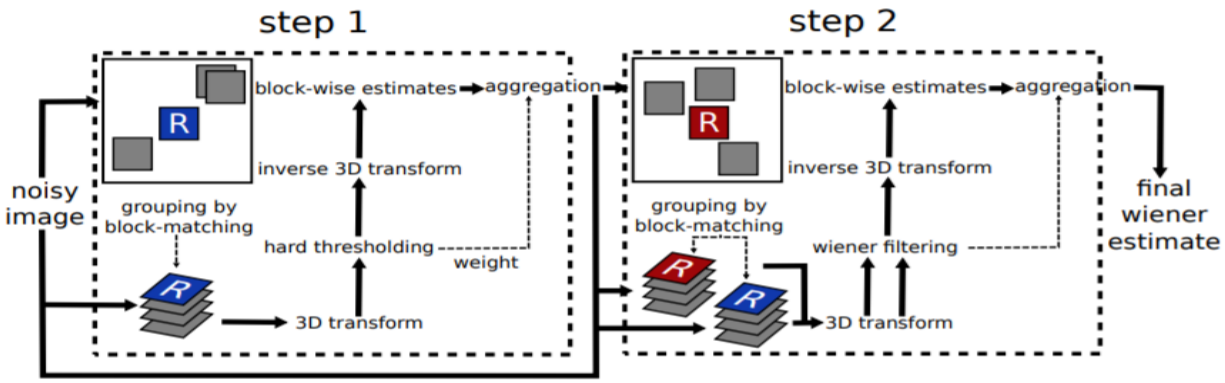
4. Step 4 - The same procedure (steps 1 - 3) are repeated for each pixel in an image. Since the same pixels will appear multiple times in different blocks an extra aggregation step is used. During aggregation weighted coefficients are calculated which are based on the count of 3D transformation coefficients that are above the threshold in step 2. Higher number of transformation coefficients that are set to zero (above the threshold) means more noise is present in the blocks and vice versa.

After all of the above steps are completed the first phase is completed. The second phase is very similar to the first phase. Same block matching is performed for each pixel except the filtered baseline is now used instead of the original image. Second phase steps are defined below.

1. Step 1 - Simmilar blocks are found and a 3D matrix is created. 3D transformation is performed on the matrix and transformation coefficients are obtained.

2. Step 2 - The transformation coefficients are filtered using the Wiener filter.
3. Step 3 - Inverse 3D transformation is applied to obtain the final version of the data patches.
4. Step 4 - Final aggregation step is performed to obtain the final estimates.

The first phase is the initial run of the algorithm, also, reffered to as the coarse run. The second phase is run on the output of the first phase in order to refine the results for better filtering by applying the Wiener filter. The Wiener filter minimizes the mean square error between the estimated signal and the desired signal. The goal of the Wiener filter is to compute the statistical estimate of an unknown signal by using a related signal as input (filtered signal). That signal is filtered to produce the estimate of for the final output. This step was incorporated over time after comparing the results with the Wiener filter and without it. The full BM3D algorithm scheme can be seen in figure 8. This algorithm can be adapted for HSI images. The initial image in our case would be the single band image and each band would be used as the source in BM3D. The final data cube is a sum of the filtered bands. Although the algorithm could be improved to search for similar patches in all bands and not only in a single band. However, this would drastically increase computation time. As it can be understood from the algorithm complexity fully implementing this advanced filtering technique from scratch is out of the scope of this research paper. We will be using an already built library that is used for 2D visual images and we will adapt it to HSI images. After performing this filtering we will compare the results with the wavelet and median filtering methods. Also, there are multiple improved versions of this algorithm. One of the versions includes

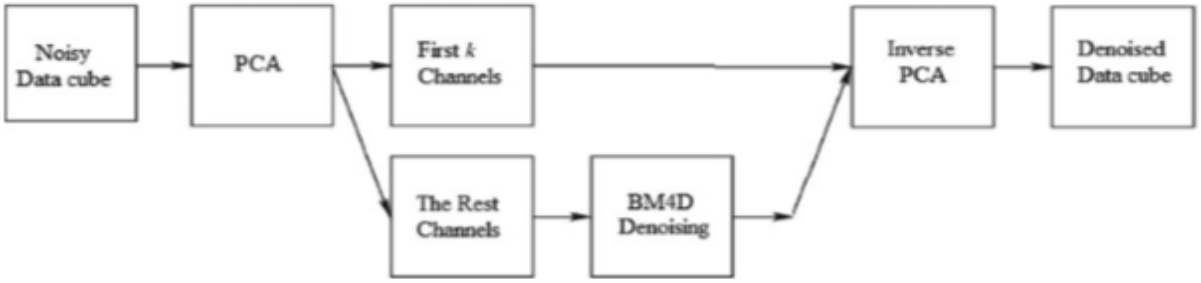


,

Figure 8. Scheme of the BM3D algorithm for a two step implementation. First phase implementation - finds matching blocks in the image, stacks them in a 3D block, applies a 3D transformation, thresholding and aggregation. Inverse 3D is applied for transforming back to a filtered image. Second phase implementation - performs same block matching on the filtered output of phase 1. Instead of thresholding a Wiener filtered is used to refine the filtering. Inverse 3D transformation is used to obtain the final data estimates [22].

a PCA step (Principal Component Analysis). PCA allows to reduce the dimensionality and size of the data. Instead of looking for simmilar patches and performing BM3D directly on the band image, principal components are obtained to seperate the noise features from the fine features. Denoising is then applied only to the noisy PCA output channels and not on all channels. This can save valuable information from being filtered out and only the noise is targeted. This way we can retain most of the features from a HSI data cube. The noisy channels are estimated by performing a wavelet transformation and calculating the HH subband noise (full details in topic 1.3.4). Full

details of this algorithm can be seen in figure 9. The only difference here is that instead of BM3D an alternative version BM4D is used of the algorithm. BM4D is an adapted version which uses a 3D data input (the full data cube) instead of a 2D image for block matching. This approach is more precise since it takes into account spatial and spectral correlations between pixels but it, also, greatly increases computation time and because of that we will not use it.



,

Figure 9. Flow chart of a proposed method for denoising hyperspectral imagery with the BM4D denoising algorithm and principal component analysis [11].

2 Experimental Results

2.1 Hyperspectral Image Noise Evaluation

2.1.1 Experimental methods and data

For the experimental part of this research we will be using images that were captured with a hyperspectral camera attached to a drone that was flying over various farmers crops at mulitple locations in Lithuania. A sample raw hyperspectral image that was captured containing multiple wavelengths can be seen in figure 10. This particulary captured image contains 192 different wavelengths ranging from 400 nm to 1000 nm at aproximately 4 nm. This means in this image we have 192 differrent bands that were captured and overlayed on the same pixel location. In this image each band consists of an 849 x 1674 pixel array and at each pixel value is the captured light reflectance spectrum from the corresponding crops location. As discussed previously these images contain noise due to various factors, so in our case each pixel array value will have the dependencies of $g(x, y) = f(x, y) + n(x, y)$.

The data analysis will be performed using the Python programming language. All the algorithms (R_1, R_2 coefficient calculation, LMLSD, SSDC, median filter, wavelet filtering, thresholding, BM3D adaptation) will be implemented from scratch without external libraries. The only and main library used for reading hyperspectral data is Spectral Python (SPy). For the SSDC algorithm we will, also, use the scikit library for training a multi variable linear reggression. Other libraries that will be used will be covered in their respective results sections. SPy library will help us to read all the wavelengths and pass them to Numpy arrays. The raw data files are in .BSQ and .HDR files (header and binary data contents files). Each of the .BSQ file varies in size (from 100 megabytes to 700 megabytes) and varies in pixel arrays size. All the files have the same number of bands (192). This is because they were all captured with the same hyperspectral camera. For this analysis we will be using and providing the results for 5 different hyperspectral images. All these images were captured by the same drone camera at the same farmers location in Lithuania. The crop in the images is cultivated rapeseed. The images contain different places of the rapeseed fields and were captured at different time. For now, we will not examine in more details the background of each

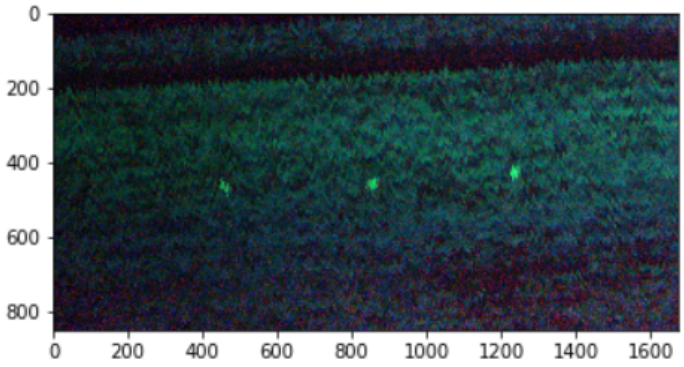


Figure 10. A sample hyperspectral image that was captured by flying a drone with a hyperspectral camera over a farmers crop field. This sample image consists of 192 bands that are all overlayed in this figure. On the x and y scales are the pixel locations on the pixel array which size is 849 x 1674. Each pixel at the image has a location defined in the array and each pixel has 192 different light intensity values at each of the bands that we were photographed by the drone.

image, since we do not have the extensive knowledge of the farmers field and will assume that each image is very similar to each other. Images naming conventions are provided below:

1. Image 1 - HSI-1 - Rape field image at 13:02 at 1460 - 2200 field meter.
2. Image 2 - HSI-2 - Rape field image at 13:02 at 2950 - 3650 field meter.
3. Image 3 - HSI-3 - Rape field image at 13:02 at 5760 - 6520 field meter.
4. Image 4 - HSI-4 - Rape field image at 14:40 at 17350 - 18120 field meter.
5. Image 5 - HSI-5 - Rape field image at 17:34 at 9683 - 10102 field meter.

2.1.2 Correlation coefficients

One of the approaches to determine noise in HSI is to calculate the R_1 for all neighboring bands and to calculate the R_2 coefficients for each pixel and its neighboring pixels. The idea in this approach is that only the $f(x, y)$ component should have correlating values between bands and between neighboring pixels. For the pixels and the bands that R_1 and R_2 coefficients will be low we can say that these bands or pixels contain a lot of noise, since the $n(x, y)$ factor is dominant and denoising should take place. We will experimentally evaluate the proposed calculations of R_1 and R_2 correlation coefficients by using raw hyperspectral image files which have not yet been preprocessed.

Firstly, the R_2 coefficient is calculated for each pixel in the data cube. This is done by iterating through each pixel value in the pixel array and calculating R_{2i} for the current pixel and one of the neighboring pixels. This way the correlation is measured for two neighboring pixels across all the wavelengths. This is done for each pixel neighbor (total 8 times). The final R_2 value for a pixel is the mean of all of the R_{2i} neighboring values. This will give us an estimate value of which pixels in the data cube have the lowest correlation between their neighbors and, therefore, contain a lot noise. The results of R_2 measurements for all of the pixels for two distinct data cubes of image 1 and image 4 can be seen in figure 11. This figure contains three dimensional surface plots where the x and y are the spatial dimensions of the data cube and the z axis contains the R_2 coefficient values. The color bar chart on the right side indicates the values of the correlation coefficient R_2 for the spatial dimension. Color blue indicates low corellation for a pixel in the image and color red indicates high correlation for that pixel. As seen from the plots the highest correlation coefficient R_2 values never reach the maximum value of 1 for any pixels. The average value for R_2 in HSI-1 is arround $R_2 = 0.82$ and in HSI-5 is only arround $R_2 = 0.69$. However, what is more interesting is that in both images there are areas where the correlation coefficient is very low (indicated by the color blue). The R_2 for these areas is only arround $R_2 = 0.3$. This can be understood that these pixels contain a lot of noise and, therefore, have low correlation values.

Next, we calculate the R_1 coefficient for all the bands. The previous R_2 coefficient helped us with determining the noisiest pixels and this coefficient will help us to determine the noisiest channels, so we can apply filters only to the noisy bands. This calculation is done by iterating through all the bands in the data cube and getting the pixel spatial arrays for the two neighboring bands A and B. Now, R_1 is calculated by the previously defined equation for these two neighboring bands. The results for the same two data cubes HSI-1 (1) and HSI-4 (4) are plotted at figure 12. As seen from the figure there are big differences in the correlation coefficients R_1 for different bands. For some bands the $R_1 = 0.3$ and for other bands it can be as high as $R_1 = 0.9$ In both images the

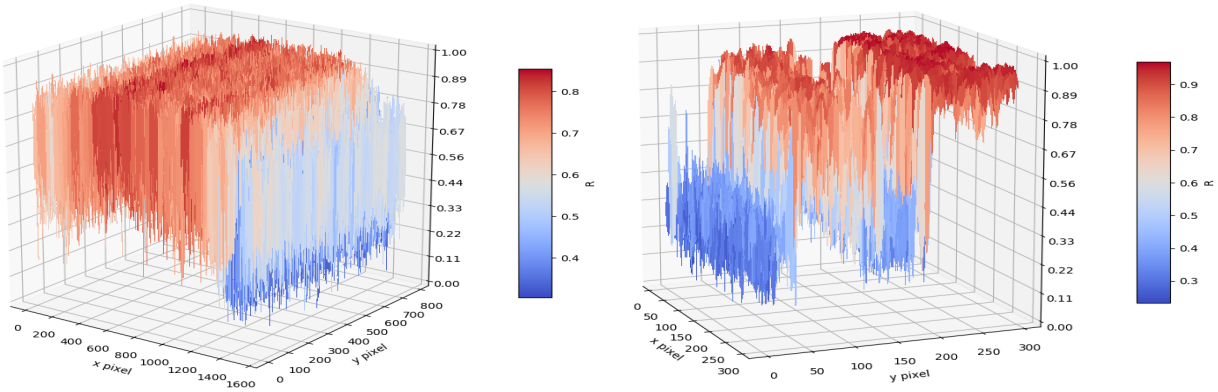


Figure 11. Pixel correlation coefficient R_2 surface plot for two seperate hyperspectral images across pixels in all bands $R_2 \epsilon [0; 192]$. On the left - hyperspectral image HSI-4 (4) ranging from pixel array $x \epsilon [0; 1620]$ and $y \epsilon [0; 849]$. On the right - hyperspectral image HSI-1 (1) ranging from pixel array $x \epsilon [0; 300]$ and $y \epsilon [0; 300]$.

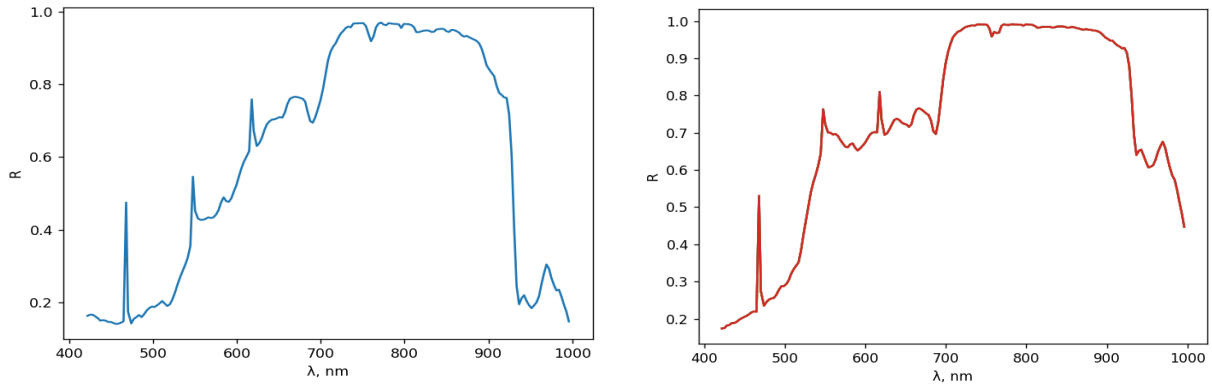


Figure 12. Correlation coefficient R_1 measurements for two adjacent bands plotted against the wavelength axis for two distinct images. On the left - hyperspectral image HSI-4 (4) correlation coefficient R_1 plot. On the right - hyperspectral image HSI-1 (1) correlation coefficient R_1 plot.

bands where $\lambda \epsilon [700; 900] nm$ the coefficient R_1 is arround $R_1 = 0.9$ which means the signal is very strong and there is little to no noise. However, at wavelengths where $\lambda \epsilon [400; 520] nm$ and where $\lambda \epsilon [920; 1000] nm$ the correlations drops significantly for both images. This indicates that these are the noisiest channels and denoising models should be applied to these channels. With this information there is no need to apply denoising to all the channels as it will affect the overall image quality and will take a much longer time to process.

2.1.3 Median filtering results

To test out whether our methodology for determining noise is correct we will implement a simple median filter. How the median filter was applied is described in section 1.3.1. We will aplly the same median filtering for the signal correlation coefficient methods and linear reggresion based algorithms.

Firstly, we will apply the filter to the correlation coefficient method. The expected outcome would be to have the both R_1 and R_2 coefficients higher in all the bands and all the pixels. This is expected since the noise origin is random and averaging out the collecting values should reduce the random noise component from the pixel value arrays. Same noise reduction techniques are

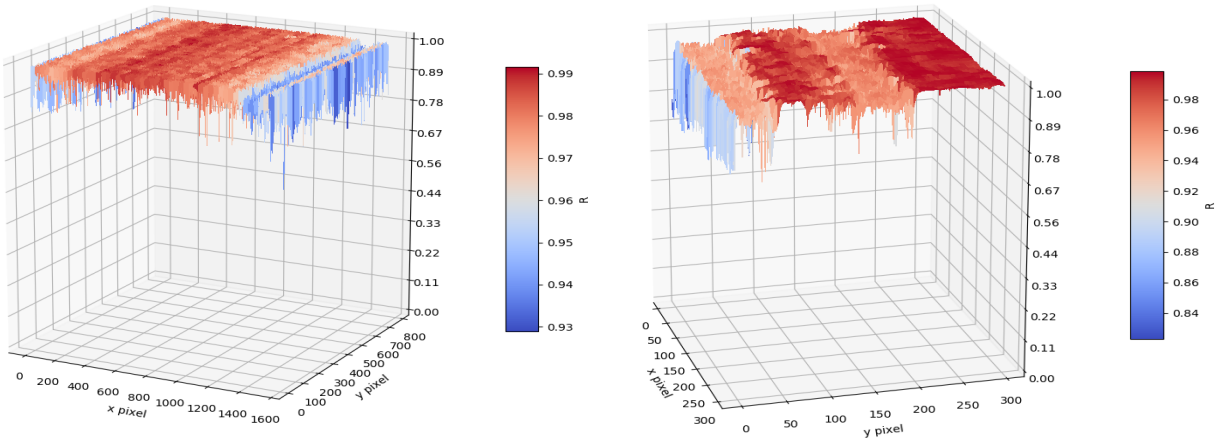


Figure 13. Pixel correlation coefficient R_2 surface plot for two seperate hyperspectral images across pixels in all bands $R_2 \epsilon [0; 192]$ after a 3x3 median filter was applied to all of the bands. On the left - hyperspectral image HSI-4 (4) correlation coefficient R_2 surface plot. On the right - hyperspectral image HSI-1 (1) correlation coefficient R_2 surface plot.

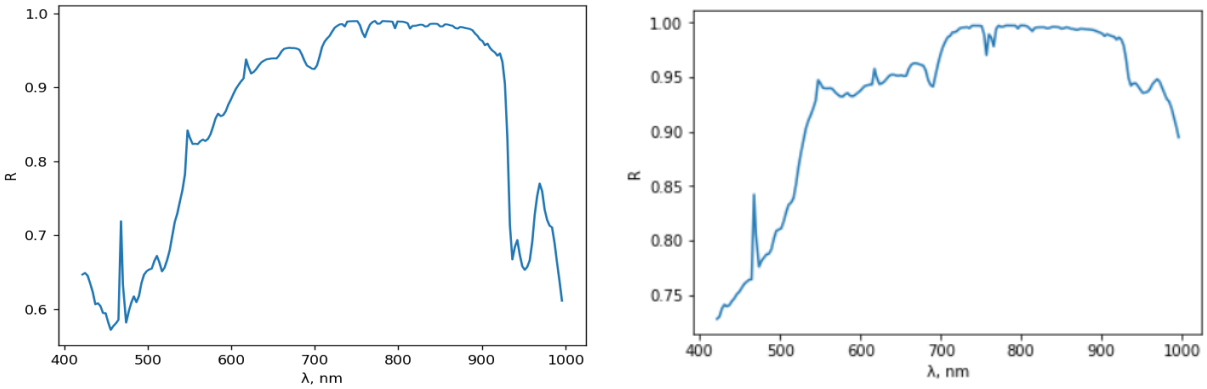


Figure 14. Correlation coefficient R_1 measurements for two adjacent bands plotted against the wavelength axis for two distinct images after a 3x3 median filter was applied to all of the bands. On the left - hyperspectral image HSI-4 (4) correlation coefficient R_1 plot. On the right - hyperspectral image HSI-1 (1) correlation coefficient R_1 plot.

applied for simple 2D images. The results for images 1 and 4 correlation R_1 and R_2 coefficients after the median filter was applied can be seen in figure 13 and in figure 14. As it can be seen from the plots both corellation coefficients R_1 and R_2 have increased for both images as we expected. We will compare how effective median filtering was for the correlation coefficients later when we will evaluate all the used methods.

Also, a new table was created for all the results on all images before and after the median filtering was applied. The table can be seen as table 1. This table holds the average R_1 value which is the average of all R_{1i} for each pixel and it holds the average R_2 value which is the average of all R_{2i} for all of bands. It also holds the average R_2 value for the the bands that had the lowest R_{2i} (the threshold was chosen for the wavelengths $\lambda \epsilon [400; 700]$). All this information is measured for the raw data cube (before filtering) and the new data cube (after filtering).

Table 1. HSI correlation coefficients R_1 and R_2 before and after median filtering is applied.

Hyperspectral image	Before/After median filtering	Average pixel correlation R_2	Average band correlation R_1	Average low band correlation R_1
HSI-1	Before	0.82	0.62	0.26
HSI-1	After	0.99	0.93	0.89
HSI-2	Before	0.92	0.62	0.27
HSI-2	After	0.99	0.95	0.90
HSI-3	Before	0.92	0.62	0.47
HSI-3	After	0.99	0.93	0.89
HSI-4	Before	0.69	0.62	0.38
HSI-4	After	0.97	0.85	0.62
HSI-5	Before	0.69	0.63	0.28
HSI-5	After	0.98	0.86	0.64

2.1.4 LMLSD and SSDC algorithms

As mentioned previously we will also implement the LMLSD and SSDC algorithms for estimating noise in all the images.

1) The LMLSD algorithm was implemented by dividing each image band into $w * w$ elements. Since the image does not a have a regular shape, each band was added with empty 0 values to make an equal amount of $w * w$ blocks in a band. Then the pixel array was broken into a 1D array and grouped by $w * w$ blocks (in all cases we have chosen a block dimension size of $w = 4$). For each block we calculated the local signal mean LM values as described in equation 1.7. Afterwards the list was iterated again and we calculated the noise standard deviation LSD as described in equation 1.8. The bin number for estimating the noise was chosen to be equal to $bin = 150, 50, 10$. Multiple tests were run with different bin sizes to see how the result changes. It was noted that the number of bins chosen has a significant impact of what the estimate noise value is for the image, therefore, this parameter is very important for the LMLSD method. It was found that the images are not very homogeneous, since the LSD estimate values for the entire image significantly change based on the bin number and they do not represent the actual noise level in the bands. This conclusion was made by comparing the overall noise estimate value for the image given from the $bins$ parameter and the overall LSD mean value. The reason for is this is because the pixel values are very small and close to each other, splitting them into bins makes the largest small interval split into multiple and now a different bin with more variance is selected by the algorithm for the overall noise. The results before and after filtering can be seen in table 2.

As seen from the results in all images the noise standard deviation LSD decreased after applying the median filter while the signal mean value LM stayed the same. Also, from the table 2 it can be seen that the LSD mean value which was calculated from averaging all of the bands LSD values differ significantly from the one estimated in the bins method. Furthermore, a figure was created for assesing the noise in each band alongside with the signal mean LM values which can be seen in figure 15. It shows the LM mean values for all bands with their noise LSD deviations represented as error bars. From the figure it can be noticed that the overall noise estimate calculated with the bins method does not represent the average error bar in the figure. This means that this method does not provide accurate results for estimating the overall noise in the images we have selected.

Table 2. HSI signal local mean LM and noise standard deviations LSD values as calculated with the LMLSD algorithm before and after median filtering is applied.

HSI	LM signal mean	LSD Before filtering (bins)	LSD After filtering (bins)	Bins number	LSD before filtering (mean)	LSD after filtering (mean)
HSI-1	0.277	0.011	0.0052	150	0.038	0.02
HSI-1	0.277	0.016	0.0047	50	0.038	0.02
HSI-1	0.277	0.0183	0.011	10	0.038	0.02
HSI-2	0.255	0.019	0.01	50	0.045	0.028
HSI-3	0.277	0.02	0.012	50	0.053	0.032
HSI-4	0.277	0.041	0.02	50	0.078	0.039
HSI-5	0.264	0.041	0.006	50	0.078	0.03

2) The SSDC algorithm was implemented in a vary similar way, since it also processes the image in $w * w$ blocks and calculates the signal mean LM (which is the same as in LMLSD) and LSD for each block. The biggest difference is that it requires a multi variable linear reggresion model to predict the value if each i, j pixel in band k. We have used the Scikit learning library for training a multi variable reggresion model. For the inputs for one bands pixels $x_{i,j,k}$ we have chosen pixel array at band $x_{i,j,k+1}$, pixel array at band $x_{i,j,k-1}$ and 8 pixel arrays at the same band $x_{i,j,k}$ with shifted pixel values representing each neighboring pixel of (i, j). After fitting the model we got 10 slope coefficients (b, c, d, e etc.) of each array as described in equation 1.10. It should be noted that the intercept coeffcient a was chosen to be equal to 0. This was done intentionally, since without it the trained model was much more accurate and with higher R^2 values. After running the predictions for a band array we got a pixel array of predicted values of band pixels $x_{i,j,k}$. With this information the residuals for each block can be calculated with the equation 1.9. Finally, we calculated the noise standard deviation for each block with equation 1.11. The final and overall noise estimate for the image is the average of all LM values for all the blocks. The results for

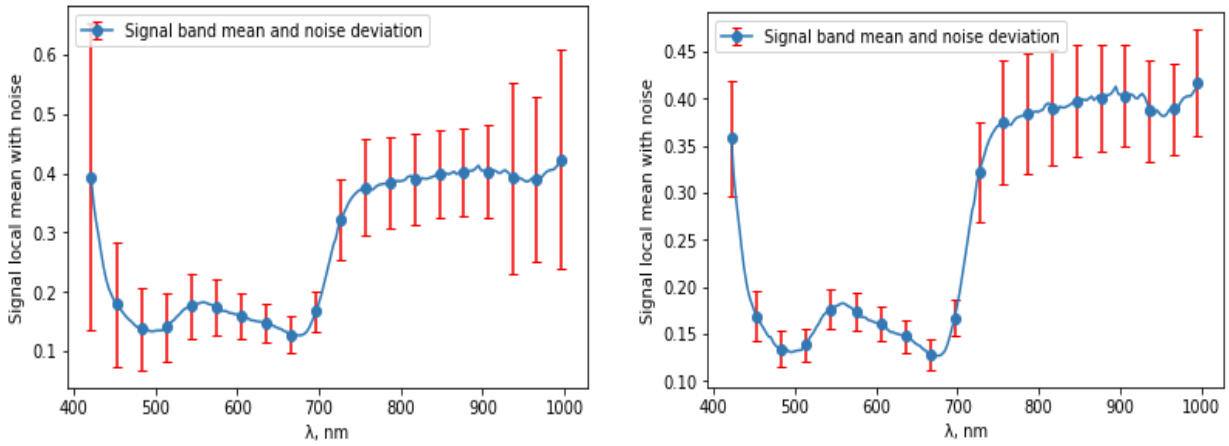


Figure 15. Hyperspectral image HSI-4 (4) signal local mean LM with noise standard deviation LSD mean values for each block $w * w$ in each band calculated with LMLSD algorithm before median filtering is applied (left) and after median filtering is applied (right).

Table 3. HSI signal local mean LM and noise standard deviations LSD values as calculated with the SSDC algorithm before and after median filtering is applied.

HSI	LM signal mean	LSD Before filtering	LSD After filtering
HSI-1	0.277	0.232	0.0181
HSI-2	0.255	0.232	0.027
HSI-3	0.277	0.233	0.027
HSI-4	0.277	0.232	0.043
HSI-5	0.264	0.233	0.03

all of the images before and after filtering can be seen in table 3. As seen from the results in all images the standard noise deviation has decreased. As with the LMLSD method the mean values of LM and LSD were calculated for each band and can be seen in figure 16. These results proved to be more precise than the ones calculated with the previous algorithm since the overall image noise estimate is much closer to the mean values of each band. This means this method is better for estimating noise in our selected images.

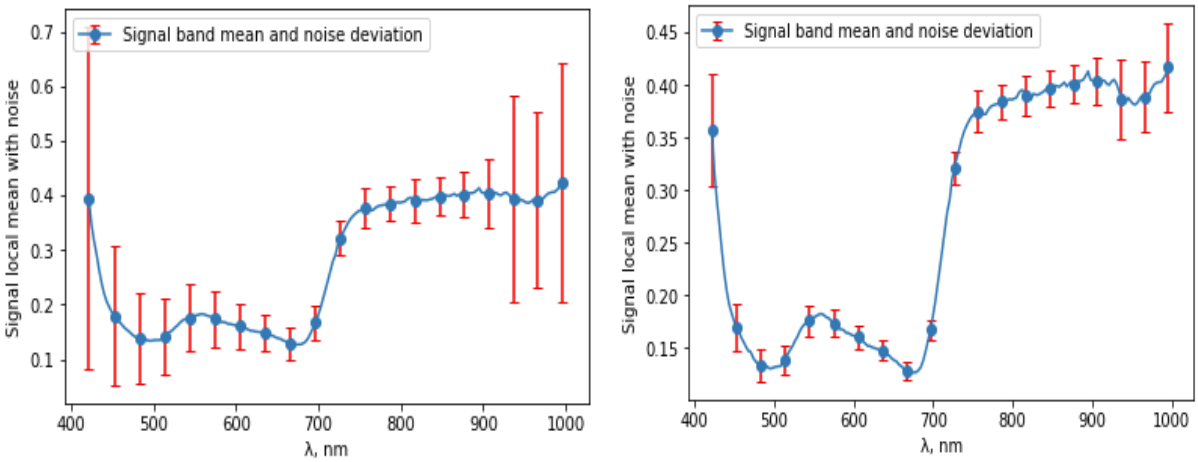


Figure 16. Hyperspectral image HSI-4 (4) signal local mean LM with noise standard deviation LSD mean values for each block $w * w$ in each band calculated with SSDC algorithm before median filtering is applied (left) and after median filtering is applied (right).

2.1.5 Noise evaluation method comparisons

After applying multiple noise estimation methods and the same filtering algorithm in all cases we can compare the filtered results with the original results and see if different noise evaluation methods provide the same results on how effective the filtering was. The comparisons can be seen in table 4. This table compares the increases in overall correlation corellation coefficients R_1, R_2 and the decreases in overall noise standard deviations for entire images before and after filtering. This table shows that in all cases applying the median filter has reduced noise deviation by a factor of $f - [2; 8]$ and increased the corellation coefficients by a factor of $f - [1.2; 1.83]$ depending on the image, initial noise and the noise evaluation method chosen. However, all the methods provided a different factor on how much the filtering was effective. The biggest differences are in the

reggresion based methods. In LMLSD the overall noise reduction varies from a factor of $f - [2; 7]$ and for SSDC it varies from a factor of $f - [4; 8]$. The results from the correlation coefficient methods were found to have less variance. According to our measurements before (table 1, table 2, table 3) we found that the HSI-4 (4) and the HSI-5 (5) images contained more noise than the others. According to these results the factor increases were, also, highest for these images. This means that the filtering in all of the methods was the most effective for the most noisy images. Also, a similar conclusion can be made for the most noisy bands - most of the band figures (figure 11, figure 13, figure 16) showed that the most noisy bands of the same image were in the wavelengths $\lambda\epsilon[400; 700]nm$ and noise decreased the most in these bands, therefore, these methods were, also, capable of finding the most noisy bands. Only the LMLSD had different results than the others for identifying the noisiest channels. However, since as mentioned previously the initial noise of the images is unknown, therefore, it is too dificult to say which of the methods is the most precise. To test it out, the same expirment would have to be conducted in a controlled noise environment. Now with the following information we can compare more advanced filtering techniques. We will

Table 4. HSI noise evaluation method variable comparisons before and after median filtering is applied (A - After filtering, B - Before filtering).

HSI	Pixel correlation R_{2A}/R_{2B}	Band correlation R_{1A}/R_{1B}	Low band correlation R_{1A}/R_{1B}	LMLSD LSD overall bin value LSD_B/LSD_A	SSDC LSD overall value LSD_B/LSD_A
1	1.2	1.35	1.62	3.4	5.1
2	1.07	1.37	1.74	1.9	3.88
3	1.07	1.34	1.67	1.67	5.07
4	1.4	1.17	1.4	2.05	5.4
5	1.42	1.2	1.32	6.8	7.76

only use SSDC and band correlation R_1 for comparing different filtering methods. This is because we have proven that LLMLSD is not consistent on providing correct noise estimate values and pixel correlations will be covered with SSDC method as a more advanced technique.

2.1.6 2D and 3D Wavelet based filtering

Here we will implement a filtering technique based on 2D and 3D wavelet transformations and thresholding. The theoretical background was already covered under topics 1.3.3 and 1.3.4. For performing the DWT and IDWT we will use the PyWavelets library which is very efficient on performing the actual wavelet transformation for a given N-Dimensional data set with a given input wavelet function. Firstly, we will test out the three different thresholding functions to examine which of them is performing the best. For this we will perform the SSDC analysis and correlation coefficient R_1 analysis. We will only look at the filtered values and will not compare to the original noisy values. The threshold function which will have the highest R_1 and the lowest SSDC LSD will be used to analyze further with the wavelet form analysis and with the 3D wavelet transformation.

Let us briefly discuss how the filtering is implemented. First, the HSI cube is split into all the band images and a direct 2D wavelet transformation from the PyWavelet library is applied to each band. For each of the transformation we will use the Coiflet wavelet form as it was previously

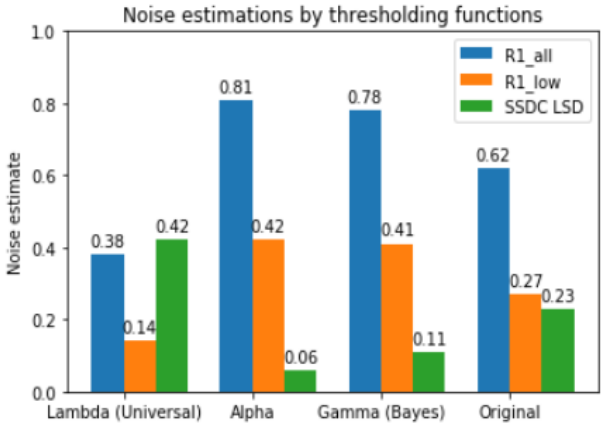


Figure 17. Median noise estimation values on all of the HSI images with different thresholding functions applied.

used for other similar research and referenced as one of the best performing. Later, we will test different wavelets to verify this. We get a subset image containing the decomposed image after the transformation is applied. We will only run a single decomposition level due to limited computing performance. Running a deeper decomposition would mean that the thresholding is applied on a more granular level and the noise variance σ_n is more precise as it is estimated from the decomposed subband of HH which is HH1. Next, thresholding is applied for all the subset bands. In all cases we are using soft thresholding for better results and selecting different thresholding functions: λ (Universal thresholding function), γ_{Bayes} (Bayes thresholding) and α (Universal thresholding with alternative soft thresholding implementation). All of these were defined in equations 1.23, 1.25 and 1.22. All the thresholding functions were implemented form scratch and applied to each of the subbands accordingly. The noise variance was estimated from the finest HH subband as referenced in equation 1.24. As a last step a reverse wavelet transformation is performed on the thresholding results and a 3D HSI cube is recreated. Finally, SSDC and R_1 analysis is performed on this data cube.

The results for this analysis can be seen in figure 17. Here we have provided the average R_1, R_{1Low} and SSDC LSD values across the 5 HSI images that we are using. As it can be seen from the results highest R_1 coefficients and lowest SSDC LSD values were achieved by using the α thresholding functions as defined in 1.22. Bayes thresholding γ_{Bayes} achieved very similar results only slightly less than α. More interestingly the final thresholding function - the universal λ did not provide good results and the filtering proved to do more harm to the image compared to the noisy image. After further investigation it was found out that the thresholding barrier was too far away compared to the values (in some runs too big, in other - too small). This caused too much image values to be adjusted with the soft thresholding function. This lowered the overall signal correlation and over-filtered the signal in most cases. To use this thresholding function we would need to adjust it in accordance to the transformation coefficient values. However, since α is already using the same λ barrier value but with a different soft thresholding implementation we will use ir to perform further research only on that single function.

Next, we will use three different wavelet functions for both 2D and 3D transformations to verify if the results are different. These will be: Haar, Daubechies and Coiflets wavelets. These wavelet forms can be seen under figure 3. The only difference for a 3D wavelet implementation is that in this case the whole 3D hyperspectral cube is passed to the wavelet transformation and

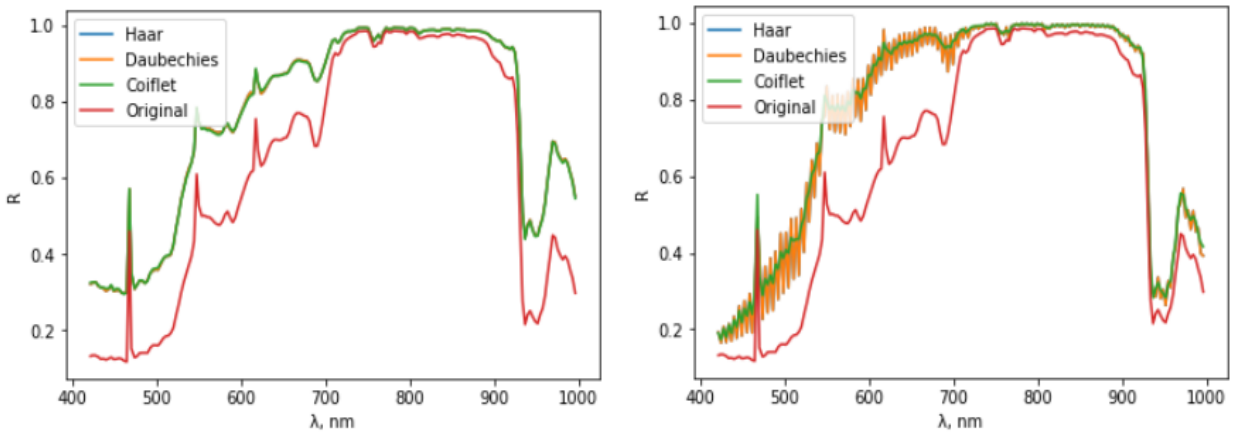


Figure 18. Band correlation coefficient R_1 plot for two hyperspectral images across all wavelengths before and after 2D and 3D DWT filtering was applied. On the left - hyperspectral image HSI-5 (5) correlation coefficient R_1 plot for 2D DWT filtering. On the right - hyperspectral image HSI-5 (5) correlation coefficient R_1 plot for 3D DWT filtering.

decomposition instead of each band separately. The HH subband in this case is a 3D smaller transformation cube. The schema for this can be seen under figure 4. Also, we are using only the α thresholding function as it performed the best out of the rest. The correlation coefficient R_1 analysis results for all of the wavelet forms after filtering for HSI-5 (5) can be seen in figure 18.

As it can be seen from figure 18 2D and 3D Wavelet filtering results were very similar. For the 2D DWT there were no dependencies found on the Wavelet form. Coiflets wavelet outperformed Haar and Daubechies wavelets only by 0.1%. This small change will not be taken into account and further research will be performed only on a single Coiflet wavelet function. However, for the 3D wavelet filtering there were more differences between wavelets as Daubechies wavelet based 3D filtering outputed a non continuous spectrum. The results seemed to fluctuate arround the mean values. This can partially be explained when looking at the thresholding value selection. During 3D wavelet thresholding the noise estimate value from equation 1.24 is estimated from data in all the channels. It is expected that the existing noise variance is similar in all channels. However, from previous research we know that channels where $\lambda\epsilon[700; 900]$ are less noisy and have higher correlation coefficients compared to $\lambda\epsilon[0; 700]$ channels. This affects the global noise estimate value and the thresholding parameter becomes less accurate. For 2D filtering in each channel iteration the noise estimate parameter is re-adjusted for that single channel. Same correlation estimation was repeated for 3D DWT with λ 10 times smaller and the R_1 value variances for that wavelet were lower. Regarding the values of R_1, both 2D and 3D wavelet filtering methods performed similar producing similar noise value estimates. 2D DWT filtering outputed slightly higher values with arround 15% incrase in R_{1All} and R_{1Low}. Details of this increase can be seen in table 5.

SSDC analysis results can be seen in figure 19. As it can be seen from the figures, 2D DWT outperformed the 3D DWT significantly. The overall average LSD for all the images in 2D DWT was about 2.5 times lower compared to 3D DWT. In 3D DWT the signal LM and noise LSD values seemed to be much closer to the original noisy images compared to 2D DWT. The only downside is that in 2D DWT signal LM values were affected by the filtering and deviated from the original LM values. However, this can be expected and the deviation is not too big containing much of the original signal shape and values. Detail results for all of noise estimate values for 2D and 3D

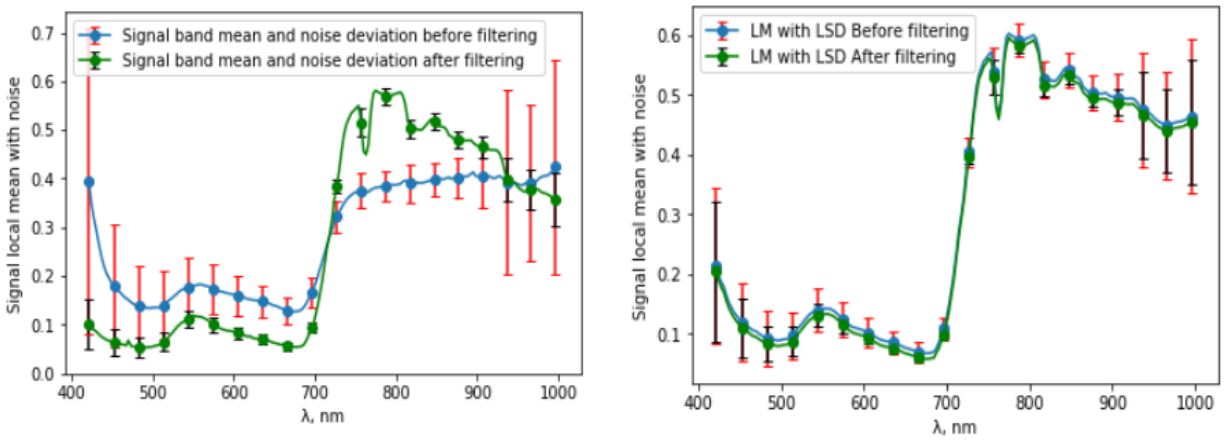


Figure 19. SSDC analysis results for HSI-5 5 before and after 2D and 3D DWT filtering was applied. On the left - SSDC results for 2D DWT filtering with Coiflets wavelet form. On the right - SSDC results for 3D DWT filtering with Coiflets wavelet form.

wavelet filtering can be seen in table 5. To avoid reduncancy the table only contains the change of relative noise estimate values. The original, noisy data values can be seen in tables 1 and 3. The main reason for different results was already mentioned in R_1 correlation analysis previously - different channels have different noise levels, therefore, a global σ for a global noise variance estimate provides an inaccurate thresholding function which results in some channels not being filtered out at a maximum precentage. In 2D DWT this is accounted for by calculating σ for each band.

Table 5. 2D and 3D DWT filtering results comparisons with soft thresholding applied (A - After filtering, B - Before filtering).

HSI	Wavelet filtering	Band correlation R_{1A}/R_{1B}	Low band correlation R_{1A}/R_{1B}	LM signal LM_A/LM_B	SSDC LSD overall value LSD_B/LSD_A
1	2D	1.35	1.62	0.39	6.12
1	3D	1.23	1.31	0.76	2.30
2	2D	1.37	1.74	0.89	5.04
2	3D	1.23	1.26	1.18	2.10
3	2D	1.37	1.68	0.99	3.87
3	3D	1.23	1.21	1.05	2.10
4	2D	1.18	1.41	0.56	2.47
4	3D	1.23	1.21	0.95	1.21
5	2D	1.21	1.32	0.73	3.65
5	3D	1.21	1.21	0.99	1.21

2.1.7 Block Matching 3D Filtering

Next, we will implement a BM3D based filtering model which will try to filter out data based on the BM3D algorithm. The details of this algorithm and the steps that are involved in this algorithm

were discussed in detail in topic 1.3.5. As mentioned in the topics, we will be using an existing BM3D library that is publicly available and acquired from http://www.cs.tut.fi/~foi/GCF-BM3D. The only difference is that this algorithm was created for denoising simple 2D images, we will be adjusting it to run for a 3D HSI data cube. Each band will be treated as a 2D image and provided as input to the algorithm. Therefore, the block matching, grouping, aggregation and all of the steps and phases defined in the section 1.3.5 will be done only on a single band image. The final filtered cube will be the aggregated image of all of the bands, similar to wavelet 2D filtering. The downside of this, is that spectral correlations are not taken into account. Also, for each band we will need to provide the noise variance estimate σ. This will be estimated for each band based on the same approach we already did from wavelet based filtering. For each band we will perform a 2D DWT transformation and calculate the σ from the suband HH, as defined in equations 1.24 and 1.23. A IDWT is not required as we are not performing thresholding directly and only estimating the noise variance. Hard thresholding will be performed by the BM3D software. As in wavelet filtering SSDC and correlation coefficient R_1 is going to be calculated before/after filtering to determine the filtering results. Before discussing the results, it is worth mentioning that the whole computation of filtering is very performance intesive and computation time is arround 4 times longer compared to wavelet or median filtering. Running the filtering for a single HSI image can take from an hour to 2 hours depending on the image size (all used HSI images contained 192 bands but different pixel count). If needed to run on a big number of images which are bigger in size a high computation machine might be needed. For reference, on our tests a local personal computer was used with 32 GB of RAM and i7-7500U CPU 2.9 GHz with 4 cores. Extending the BM3D algorithm to a BM4D which would include spectral corellations would dramatically increase computation time. Nevertheless, the correlation coefficent R_1 and SSDC estimate results for this filtering method can be seen under figures 20 and 21.

As it can be seen from the results BM3D proved to be a very good filtering method. It outperformed both 3D and 2D wavelet filtering methods in all noise estimates. The average band corellation coefficients R_1 and R_{1Low} were accordingly 1.44 and 3.01 times bigger compared to orignal values. SSDC results were even better - average SSDC LSD increase across all images was 8.77. This means the estimated noise was 8.77 times lower after the filtering compared to the

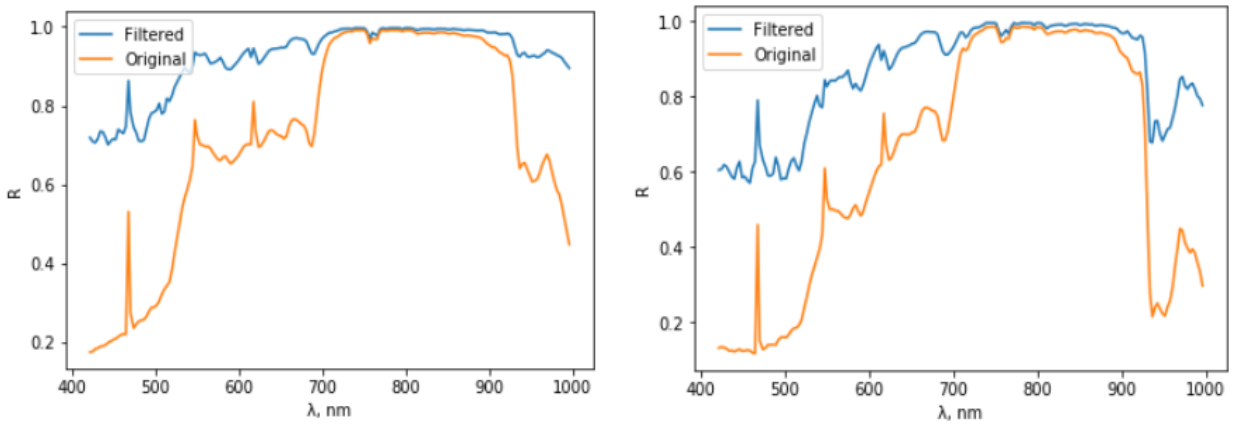


Figure 20. Band correlation coefficient R_1 plot for two hyperspectral images across all wavelengths before and after BM3D filtering was applied. On the left - hyperspectral image HSI-1 (1) correlation coefficient R_1 plot for BM3D filtering. On the right - hyperspectral image HSI-5 (5) correlation coefficient R_1 plot for BM3D filtering.

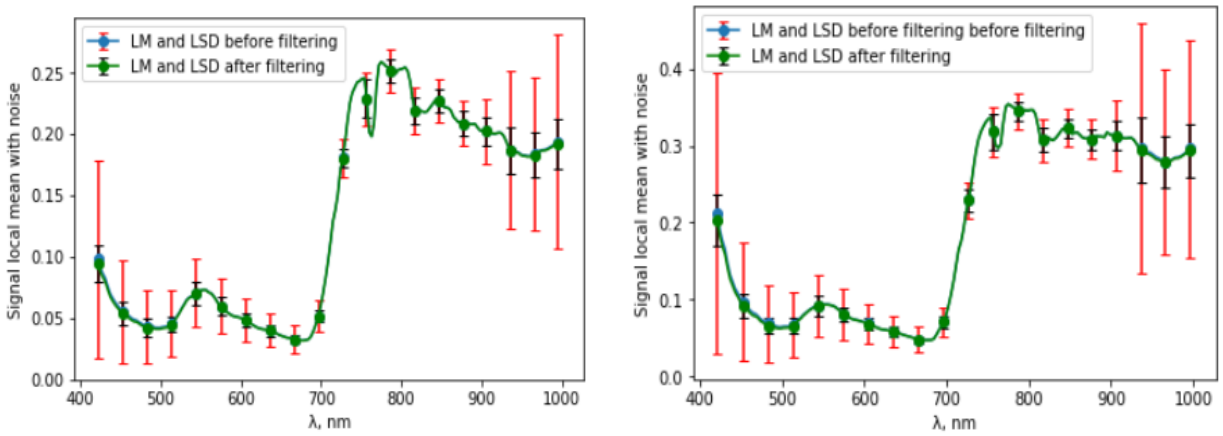


Figure 21. SSDC analysis results for two hyperspectral images before and after BM3D filtering was applied. On the left - SSDC results for HSI-1 (1) BM3D filtering with Coiflets wavelet form. On the right - SSDC results for HSI-5 (5) BM3D filtering with Coiflets wavelet form.

original. Furthermore, LM signal values were not changed after the filtering. This means only the noisy LSD parts were targeted and no useful information was lost, maintining the original signal shape and values. This was not demonstrated in any previous filtering methods including the median filter. The full noise estimate values can seen in table 6.

Finally, an improved version of the BM3D algorithm was implemented by including a PCA (Principal Components Analysis) step. This in theory should have improved the BM3D results since the whole algorithm would be performed only on PCA output components and not on the transformation coefficients. Full schema of the implemented algorithm can be seen in figure 9. However, in our implementation with the PCA analysis this was not proved to be the case. The final output filtered HSI cube did not improve in the noise estimated values. SSDC and correlation coefficient R_1 analysis results improved only slightly compared to the original noisy cube and performed the worst compared to the previously used filtering methods. The main reason for these results might be that the library responsible for BM3D filtering is not tuned to be used together with PCA components as the documentation on it is limited. Another possibility is that the PCA transformation was implemented incorrectly. However, for this analysis an existing implementation of PCA transformation in Scikit learning Pyhton library was used lowering this possibility. Therefore, we will not provide any results for this analysis step as the issues were not directly data related. However, the excellent results of the BM3D algorithm is sufficient and further, more detail PCA analysis step seemed to not be needed.

2.1.8 Image quality analysis

Lastly, we will try to asess the filtering qualities in terms of PSNR or Peak Signal Noise Ratio. This method is commonly used to compare how the filtering is effective by taking the noise-free image and the denoised image and calculating the MSE between these two images as noted in equation 1.12. The only problem we have is that we do not have a noise free image due to the nature of the research, so we will try to compare against the noisy image for all the filtering techniques by using the same baseline image and try to make conclusions from that information. PSNR will be calculated for each band of the HSI cube and we will obtain a PSNR to wavelength λ relationship. Also, we will calculate the average PSNR value across all bands. This will give a single dB value for easier comparisons between filtering methods. The results for the image quality examination

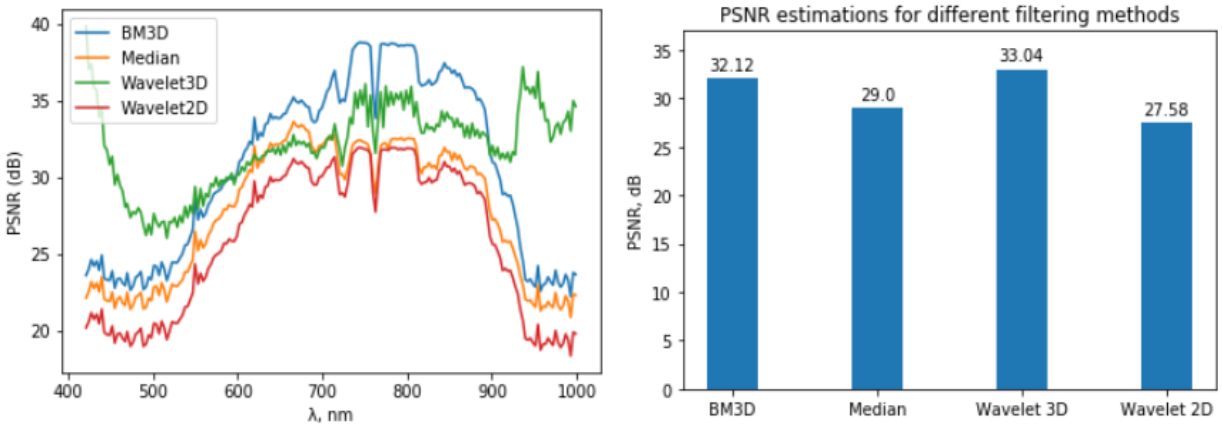


Figure 22. Image quality analysis results for hyperspectral images. On the left - PSNR and λ dependency for image HSI-5 (5) for different filtering techniques. On the right - Average PSNR estimation results across all the bands and all the images for different filtering techniques.

can be seen in figure 22. Here we have provided the PSNR and λ relationship for a single HSI-5 (5) image and the PSNR bar chart with the mean values of PSNR across all the images for different filtering methods. As it can be seen from figure 22 highest PSNR values were achieved by wavelet 3D filtering and BM3D filtering method. Typical PSNR values for Wavelet transformations with different wavelet forms are usually between $20-30dB$ [21]. In our case 3D wavelet outperformed 2D wavelet transformation quite significantly in this measurement ($33dB$ and $27.6dB$) which is interesting since in other measurements (SSDC and R_1) analysis 2D wavelet outperformed 3D wavelet with lower signal information loss. The reason for this can be seen on the detail PSNR wavelength λ relationship. 3D wavelet has a significantly higher PSNR in the lower and the higher bands ($\lambda\epsilon[400; 550]$ and $\lambda\epsilon[900; 1000]$). This is because we are using the noisy image as baseline and calculating MSE between the filtered and original as specified in equation 1.12. This means that the smaller the MSE between the noisy image and the filtered one - the more noise exists in these channels. That means wavelet 3D filtering performed the worst in these wavelengths as these are the noisiest ones (as stated by the previous analysis). The same can be said for the $\lambda\epsilon[550; 900]$ inverval. Bigger PSNR means smaller differences between the filtered and the original. The key here is that we already know that these subbands already have a high correlation R_1 and high SSDC LSD, so there is not much noise to filter in the first place, therefore, PSNR is higher in these bands. Regarding the BM3D PSNR values - usual values as stated on 2D image denoising are

Table 6. BM3D based filtering results (A - After filtering, B - Before filtering).

HSI	Band correlation R_{1A}/R_{1B}	Low band correlation R_{1A}/R_{1B}	LM signal LM_A/LM_B	SSDC LSD overall value LSD_B/LSD_A
1	1.48	3.23	1	11.05
2	1.5	3.18	0.99	7.73
3	1.53	3.21	1	13.65
4	1.32	2.96	0.99	4.97
5	1.37	2.86	1	6.48

found between $30dB$ and $40dB$ which our implementation has achieved [31]. But as mentioned before on most of the similar research a clean and a filtered image is compared (which we do not have). Therefore, looking in the absolute values should not be of most importance. Nevertheless, we have already got good insights on this PSNR analysis for our filtering methods. Lastly, we will summarize and perform a final method comparison, and provide final recommendations.

2.1.9 Filtering method comparison

Finally, we can compare all the results for the implemented filtering methods and draw conclusions. The final filtering method comparisons can be seen in table 7. This table shows all the average noise estimate value increase of all the HSI images compared to original values. As it can

Table 7. Filtering method comparison based on average HSI noise estimate values (A - After filtering, B - Before filtering).

Filtering method	Band correlation R_{1A}/R_{1B}	Low band correlation R_{1A}/R_{1B}	LM signal LM_A/LM_B	SSDC LSD overall value LSD_B/LSD_A	PSNR, dB
Median	1.27	1.5	0.6	8.3	29
Wavelet 2D	1.3	1.55	0.75	4.23	27.58
Wavelet 3D	1.22	1.24	0.98	1.78	**33**
BM3D	**1.44**	**3.1**	**1**	**8.76**	32.12

be seen from the table the overall best performing filtering method is the BM3D. It provides the best denoising results aswell as keeping the useful signal intact (reducing the SSDC LSD noise deviation by more than 8 times). The PSNR value was higher only for the wavelet 3D filtering method. The reasons for this were already discussed previously. The only downside for using the BM3D method is that it takes a long to compute. In our research in took from 1 to 2 hours on a single HSI data cube. Also, this method could be improved by extending the block matching algorithm to search for similar patches in all the bands and not only the current. This would increase band correlation R_1 dramatically but, also, dramatically increase computation time. The second best performing method was the 2D wavelet based transformation. This method reduced local noise by more than 4 times and increased band correlation by more than 20%. The downside is that the calculated PSNR value was the lowest. Although due to the missing clean image this fact should not be considered as impactful. The worst performing methods were the median filter and the wavelet 3D. The median filter performed very good on the SSDC LSD estimation (reducing the value by more than 8 times) but that was expected as the filter is only a simple 3x3 matrix that computed the averages of local pixel values. This approach is very efficient at denoising but a lot of information is lost and this can be seen in the lowest LM signal value LM_A/LM_B. The median filter is not used in real world cases for denoising data with important information and in our research it was only used to create a noise evaluation methodology. Lastly, the wavelet 3D performed the worst with the lowest noise estimate value increases. The LM signal value for wavelet 3D did not change much but this is due to not filtering enough noise. This can be concluded by looking into the PSNR value and the overall SSDC LSD value. The reasons for this were discussed previously but the main issue is that the thresholding parameter cannot be used for

all the bands as a global threshold value. The σ_n value must be calculated for each of the band separately as the noise levels are quite different in each band (as it is done in wavelet 2D filtering).

The results for the noise method evaluation were already covered in section 2.1.5. Here we will only mention the key factors. Initially the noise evaluation was supposed to be performed for LMLSD LSD and SSDC LSD but since LMLSD method was found as not being reliable enough it was not used to perform more complex filtering. Pixel correlation R_2 method was also not used in further analysis mainly due not having a need for it as SSDC LSD already covered neighboring pixel and band correlations. However, the method could be used when there is a need to analyze the provided HSI images if they are continuous and represent single substances. For example, if we had an image where multiple different objects were found (grain fields, houses, trees etc.) we could identify different regions of the image where the correlation coefficient R_2 was similar. Also, it could be used for image slicing as usually the corners of these images have a lower correlation R_2 coefficients due to hyperspectral camera lensing. Finally, as a last conclusion we would recommend using the BM3D filtering technique for HSI image denoising and SSDC LSD (optionally R_1, R_2 correlation methods) for evaluating noise.

Conclusions and Recommendations

1. In this research paper we have analyzed hyperspectral images, hyperspectral imaging technologies and the noise origins in hyperspectral images. We have analyzed and implemented multiple HSI noise evaluation methods: correlation coefficient based (R_1 and R_2) and linear reggression based (LMLSD and SSDC). To compare different noise evaluation methods we have implemented a median filter as described in section 1.3.1. For the analysis we have used 5 different noisy hyperspectral crop images captured from an airborne drone. We have concluded that in all noise evaluation methods the noise estimate parameter values were lower after filtering than in the original image, therefore, concluding that our designed noise evaluation methodology can be used to represent noise levels in HSI images.

2. After analyzing the different noise evaluation methods we have concluded that the correlation coefficient based estimate methods and the linear reggresion based methods all give different results on how effective the filtering is. This is because our knowledge of the initial noise found in the images is limited. Therefore, we cannot say which of the methods represent the existing noise in the most accurate way. Nevertheless, we have concluded that the linear reggresion based algorithm LMLSD - does not provide accurate results in estimating the whole HSI data cube noise. The main reasons for the inaccurate results are due to the images not being fully homogeneous with varying noise levels on different bands which caused LMLSD LSD to provide inaccurate and inconsistent results. However, the other implemented methods - SSDC, R_1 correlation coefficient and pixel correlation R_2 provided with similar results on determinig which are the most noisy bands and two of them were used to evaluate advanced filtering algorithms.

3. More advanced filtering algorithms were implemented and compared. The best results were given by the BM3D algorithm which outperformed wavelet 2D, 3D and the basic median filter in all noise estimate parameters. More importantly it did not change the original signal LM values and the initial signal remained intact after filtering (full results in table 7). This was not demonstrated by any other method and this method is our recommendation for denoising HSI images. Direct 2D and 3D Wavelet Transformation based filtering was, also, implemented with different threshold functions and different wavelet forms. It was found that there are minimum dependencies on the selected wavelet form but big dependencies on thresholding functions. The best results were achieved with the λ threshold parameter and α threshold function (equations 1.23, 1.22). 3D DWT based filtering performed worse compared to 2D due to images not having homogeneous noise in all bands and, therefore, impacting the thresholding parameter accuracy. As expected, median filtered provided good SSDC LSD results but it had the biggest changes in signal LM values, changing the original useful signal information.

4. Finally, Peak-Signal-To-Noise (PSNR) Ratio parameter (equation 1.12) was calculated to asess image quality for different filtering technologies. The results showed that the worst performing method was the wavelet 3D. Other methods provided simillar results with BM3D having the highest PSNR values in $\lambda\epsilon[550; 900]$ inverval and lowest in $\lambda\epsilon[400; 550]$ and $\lambda\epsilon[900; 1000]$. This means the biggest filtering was done the lowest and highest bands, and least filtering was done in mid-range. Therefore, concluding that the BM3D targeted the noisiest channels for the filtering leaving the least noisy intact.

Future research work

As discussed throughout the research project there are multiple possibilities for a research work extension. These will be sumarized in the following list:

1. Comparing different noise evaluation methods with known noise origins. Another issue that was encountered in the HSI analysis was that the initial noise and what kind of noise (Gaussian, impulse, periodic etc.) was in the images was not known. Without that knowledge it is too hard to clearly tell which of the noise evalution methods were the most accurate since they showed different noise parameter levels. This can be solved by running the noise evaluation metohds on HSI where the initial noise is known. This can be done by finding a clean HSI and adding artificial noise (Gaussian, Periodic, Impulse etc.) on it. Afterwards, denoising with the noise removal techniques and cross checking the correct PSNR with our noise estimate parameters.

2. Finding the best performing thresholding function to use with the 2D DWT transformation. As it was found in the research there is a high dependency for denoising results on which thresholding function is used and what is the thresholding parameter. A more detail research could be performed by implementing more complex thresholding functions such as heasure, minimax, rigsure thresholding functions.

3. Implementing more and different advanced noise removal techniques. In current research we only deeply looked at BM3D and wavelet based denoising. Even though these two are the most popular ones there are other techniques that could be tried out.

References

[1] Continuous wavelet transform and scale-based analysis. http://www.mathworks.com/help/wavelet/gs/continuous-wavelet-transform-and-scale-based-analysis.html. Accessed: 2019-11-15.

[2] Rajeev Aggarwal, Jai Karan Singh, Vijay Kumar Gupta, Sanjay Rathore, Mukesh Tiwari, and Anubhuti Khare. Noise reduction of speech signal using wavelet transform with modified universal threshold. *International Journal of Computer Applications*, 20(5):14--19, 2011.

[3] Tudor Barbu. Variational image denoising approach with diffusion porous media flow. In *Abstract and Applied Analysis*, volume 2013. Hindawi, 2013.

[4] Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 60--65. IEEE, 2005.

[5] Antoni Buades, Bartomeu Coll, and Jean Michel Morel. On image denoising methods. *CMLA Preprint*, 5, 2004.

[6] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. Non-local means denoising. *Image Processing On Line*, 1:208--212, 2011.

[7] Dr Philippe Cattin. Image restoration: Introduction to signal and image processing. *MIAC, University of Basel. Retrieved*, 11:93, 2013.

[8] Juan Cebrion and Carlos Travieso. *2-D Discrete Wavelet Transform for Hand Palm Texture Biometric Identification and Verification*. 03 2012.

[9] Yakup Cengiz and Yard Doç Umut Arıöz. An application for speech denoising using discrete wavelet transform. In *2016 20th National Biomedical Engineering Meeting (BIYOMUT)*, pages 1--4. IEEE, 2016.

[10] S Grace Chang, Bin Yu, and Martin Vetterli. Adaptive wavelet thresholding for image denoising and compression. *IEEE transactions on image processing*, 9(9):1532--1546, 2000.

[11] Guangyi Chen, Tien D Bui, Kha Gia Quach, and Shen-En Qian. Denoising hyperspectral imagery using principal component analysis and block-matching 4d filtering. *Canadian Journal of Remote Sensing*, 40(1):60--66, 2014.

[12] Çiğdem Polat Dautov and Mehmet Siraç Özerdem. Introduction to wavelets and their applications in signal denoising. *Bitlis Eren University Journal of Science & Technology*, 8(1), 2018.

[13] Igor Djurović. Bm3d filter in salt-and-pepper noise removal. *EURASIP Journal on Image and Video Processing*, 2016(1):13, 2016.

[14] David L Donoho and Jain M Johnstone. Ideal spatial adaptation by wavelet shrinkage. *biometrika*, 81(3):425--455, 1994.

[15] Alexandre Y Fong and Elliot Wachman. Hyperspectral imaging for the life sciences. *Biophotonics International*, 15(3):38, 2008.

[16] Lianru Gao, Qian Du, Bing Zhang, Wei Yang, and Yuanfeng Wu. A comparative study on linear regression-based noise estimation for hyperspectral imagery. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 6(2):488--498, 2013.

[17] Rafael C Gonzalez and Richard E Woods. Image processing. *Digital image processing*, 2:1, 2007.

[18] Rafael C Gonzalez, Richard E Woods, et al. Digital image processing [m]. *Publishing house of electronics industry*, 141(7), 2002.

[19] Xavier Heurtebise and Sébastien Thon. Discrete tools for virtual sculpture. pages 415--422, 01 2006.

[20] Timo Hyvärinen. What makes push-broom hyperspectral imaging advantageous for art applications. *SPECIM, Spectral Imaging Ltd*, page 9, 2016.

[21] Mohideen Kother, Perumal Arumuga, Krishnan Nallaperumal, and Mohamed Sathik. Image denoising and enhancement using multiwavelet with hard threshold in digital mammographic images. *International Arab Journal of e-Technology*, 2, 01 2011.

[22] Marc Lebrun. An analysis and implementation of the bm3d image denoising method. *Image Processing On Line*, 2:175--213, 2012.

[23] Amy Lowe, Nicola Harrison, and Andrew P. French. Hyperspectral image analysis techniques for the detection and classification of the early onset of plant disease and stress. *Plant Methods*, 13(1):80, Oct 2017.

[24] Guomin Luo, Daming Zhang, and DD Baleanu. Wavelet denoising. *Advances in wavelet theory and their applications in engineering, physics and technology*, page 634, 2012.

[25] Anne-Katrin Mahlein. Plant disease detection by imaging sensors – parallels and specific demands for precision agriculture and plant phenotyping. *Plant Disease*, 100(2):241--251, 2016. PMID: 30694129.

[26] Michel Misiti, Yves Misiti, Georges Oppenheim, Jean-Michel Poggi, et al. Wavelet toolbox user's guide. *The Math Works Ins*, 1996.

[27] Yusuf Perwej, Firoj Parwej, and Asif Perwej. Copyright protection of digital images using robust watermarking based on joint dlt and dwt. *International Journal of Scientific & Engineering Research*, 3(6):1--9, 2012.

[28] Behnood Rasti, Paul Scheunders, Pedram Ghamisi, Giorgio Licciardi, and Jocelyn Chanussot. Noise reduction in hyperspectral imagery: Overview and application. *Remote Sensing*, 10(3):482, 2018.

[29] Behnood Rasti, Johannes R Sveinsson, Magnus O Ulfarsson, and Jon Atli Benediktsson. Hyperspectral image denoising using 3d wavelets. In *2012 IEEE International Geoscience and Remote Sensing Symposium*, pages 1349--1352. IEEE, 2012.

[30] Serge Savary, Andrea Ficke, Jean-No Aubertot, and Clayton Hollier. Crop losses due to diseases and their implications for global food production losses and food security. *Food Security*, 4(4):519--537, Dec 2012.

[31] Department of Signal Processing TAMPERE UNIVERSITY OF TECHNOLOGY. Image and video denoising by sparse 3D transform-domain collaborative filtering. http://www.cs.tut.fi/~foi/GCF-BM3D/, 2019. [Online; accessed 20-12-2019].

[32] Pekka J Toivanen, Arto Kaarna, Jarno S Mielikainen, and Mikko Laukkanen. Noise reduction methods for hyperspectral images. In *Image and Signal Processing for Remote Sensing Viii*, volume 4885, pages 307--314. International Society for Optics and Photonics, 2003.

[33] Yu Wang, Nicholas P. Reder, Soyoung Kang, Adam Glaser, and Jonathan Liu. Multiplexed optical imaging of tumor-directed nanoparticles: A review of imaging systems and approaches. *Nanotheranostics*, 1:369--388, 08 2017.