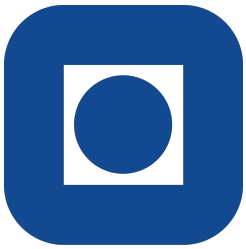

NTNU

TTK4255

Robotvision

Hyperspectral imaging

Mads Formo

Group 26

April 20, 2020

Contents

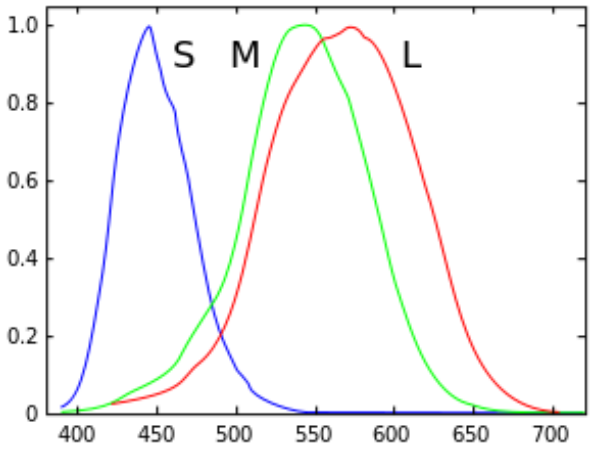

Figure 1: Graph for the human color sensitivity curves, according to Wikipedia [1]

1 Getting familiar with the data

1.1 Finding the spectral resolution

To find the spectral resoulution of the dataset, we load the *hico_wl* array, which contains the wavelength corresponding to band i. We loop through the array and compare each wavelength i with the the previous wavelenght *i-1* and we find that the average distance between the wavelengths is $5.728nm$, which seems to be constant between all wavelengths.

1.2 Relation to human color perception

The color sensitivity of the human eye is shown in fig. 1. As we can see, blue color has a peak around $450nm$ (S-curve), green peaks at $550nm$ (M-curve), and red at $600nm$ (L-curve).

1.3 Create a pseudo RGB image from the hyperspectral bands

From the *hico_wl* array, we find that Blue (450nm) is located at index $i = 8$, green (550nm) at $i = 25$, and finally red (600nm) at $i = 34$. We combine these indices from the HICO dataset and show it as an image to create a pseudo RGB image, shown in fig. 2.

1.4 Representative spectra for selected points

We want to look at the representative spectra of the points (20,20), (100,70) and (400,30), which is in deep water, shallow water and vegetation respectively. As we can see in fig. 3, we see that there is a clear difference in the spectra between water and vegetation. Both have amplitude peaks at the lower end of the spectra and then drop off in power as the wavelength increases. Vegetation however increases again in power at a wavelength of around 700nm, while the water is still decreasing. The findings here seem to agree to the findings of Lucke et al [2] as the general shape of the curves matches those of figure 12 in that report.

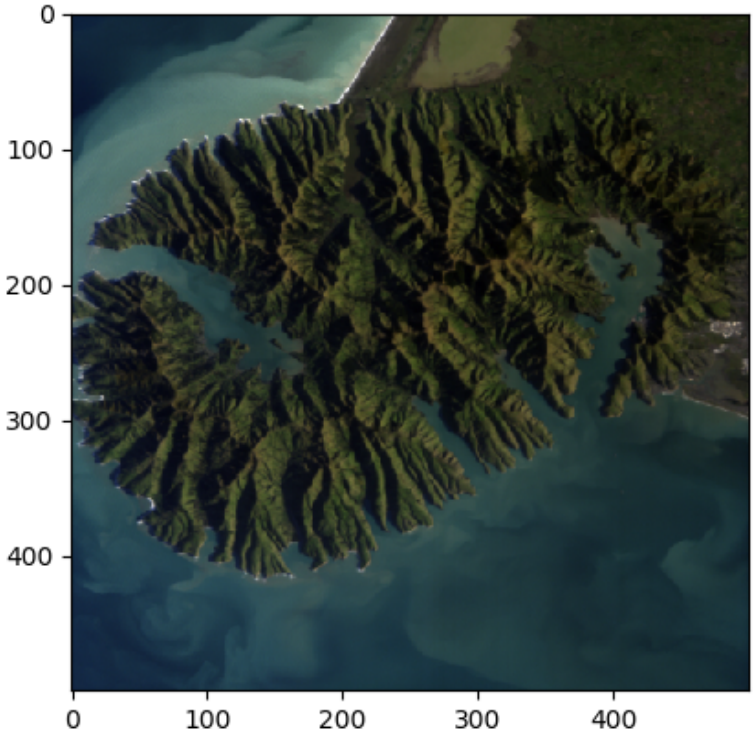

Figure 2: Pseudo RGB image, showing R (600nm), G (550nm), B (450nm)

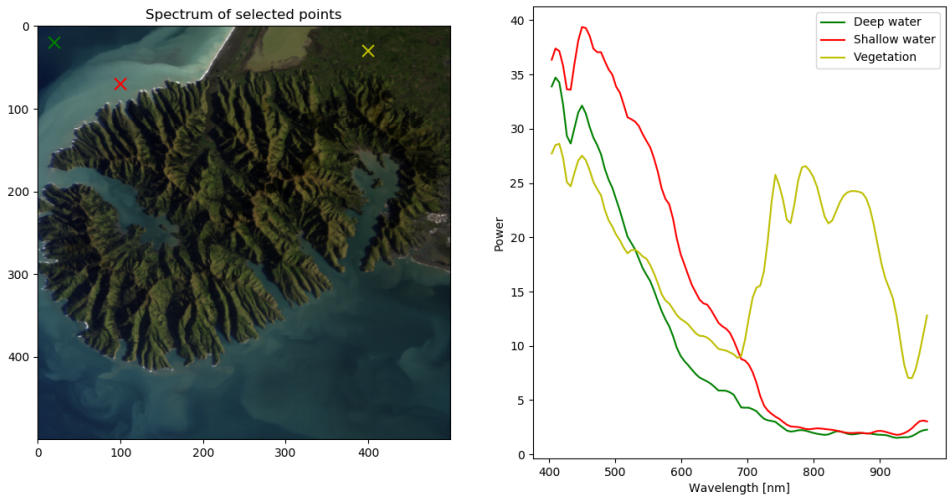

Figure 3: Representative spectra of specific points

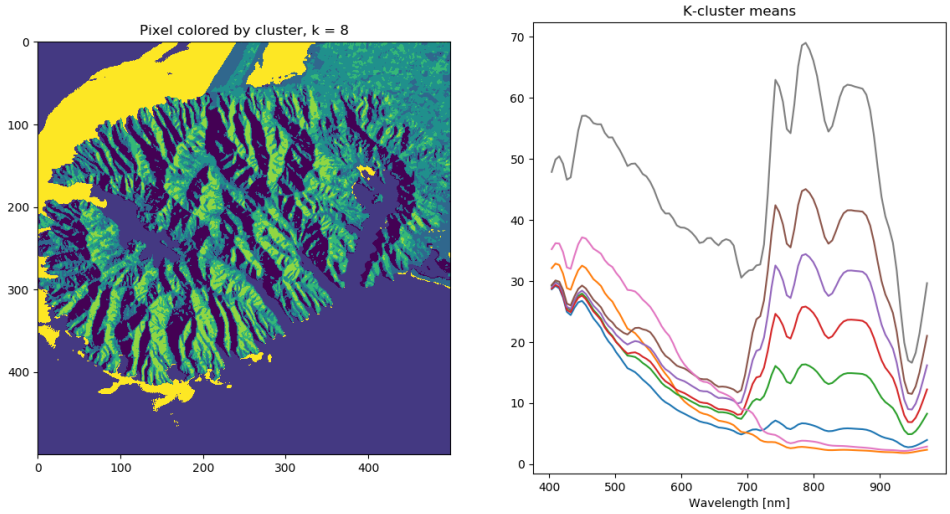

Figure 4: K-mean clusters of the image

2 Classification & Bio-geophysical Parameter Retrieval

2.1 Can we predict where there is chlorophyll through classification?

We will use *K-Means clustering* to classify the data. K-means clustering is a unsupervised learning algorithm that can be used to classify and cluster data into k different clusters. The data points are adjusted iteratively until all points are associated with the nearest cluster. We want to cluster each observation (pixels, with n spectral channels) into a specific cluster (environment class, ie. deep water, shallow water, vegetation).

As we can see from fig. 3, we know that those three different points have distinctly different spectra, thus it should be possible to classify them accordingly. The results of a K-mean clustering, run with Spectral Python's *kmeans* function [3], can be seen in fig. 4. We clearly see different classes for water, land, and vegetation, the latter containing lots of chlorophyll. We also see a very distinct class along the coast on the upper part of the image. This may very well be a collection of chlorophyll, but it might also just be shallow water, or more likely a combination of both.

2.2 How well can we directly estimate the chlorophyll content?

We use the NASA OBPG algorithm, defined in equation 4 in the assignment [4], as well as the parameters given there, to try to visualize the chlorophyll contents. Using the closest available wavelengths in the dataset, $\lambda_{green} = 553$ (i = 26) and $\lambda_{blue} = [444, 490, 507]$ (i = [7, 15, 18]). The results can be seen in fig. 5. We can clearly see high concentrations on the north west coast (assuming north is at the top of the image), same place as in fig. 4, but now we also see quite a bit on the southern coast as well. Thus it seems that this algorithm performs better than the k-means clustering.

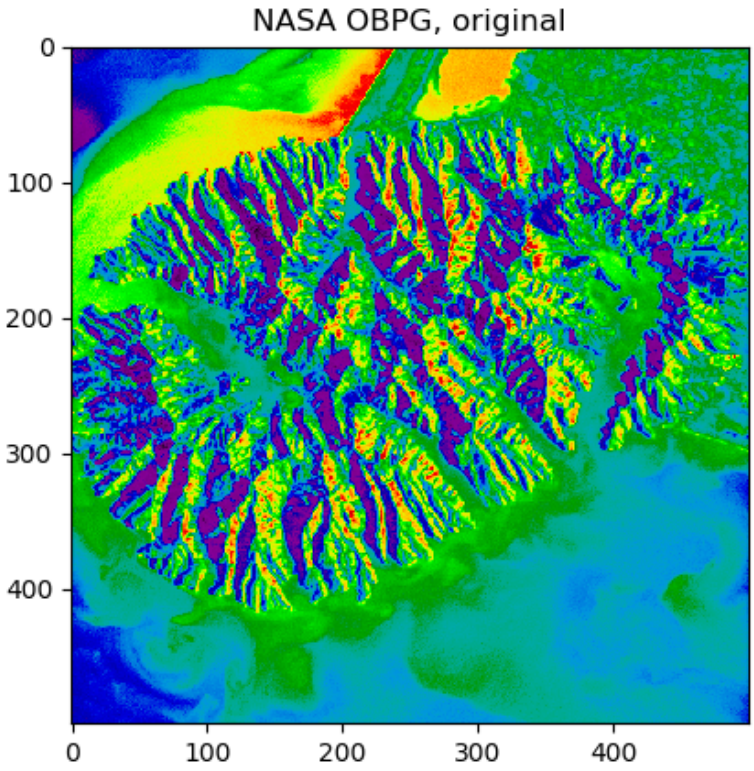

Figure 5: Results from the NASA OBPG algorithm, showing chlorophyll concentrations in the water

2.3 How can we estimate the reflectance from the surface of the ocean?

The data in the HICO dataset actually contains the measurement of radiance exiting the top of the atmosphere, and not the radiance of the water directly. Therefore we must recover the radiance R_{rs} of the water from the top of atmosphere (TOA) measurements. This is done with the empirical line (ELM) method, as described in [4], on the form eq. (1). Where a and b are terms that model the absorption of light in the atmosphere, which is to be estimated, and L is the measured value in the HICO dataset.

$$R_{rs}(\lambda) = \frac{L(\lambda) - b(\lambda)}{a(\lambda)} \tag{1}$$

After performing the atmospheric correction, the resulting image is shown in fig. 6. It is very difficult to see a clear difference between the two by only looking at the image, but if we look at the pixel values for the red, green and blue channels separately, we see that the blue color channel is only about 2% of the original, uncorrected pixel value, while the red and blue channels are about 9-10% of their original values. Thus it would seem that the atmospheric correction removes some of the blue color of the image.

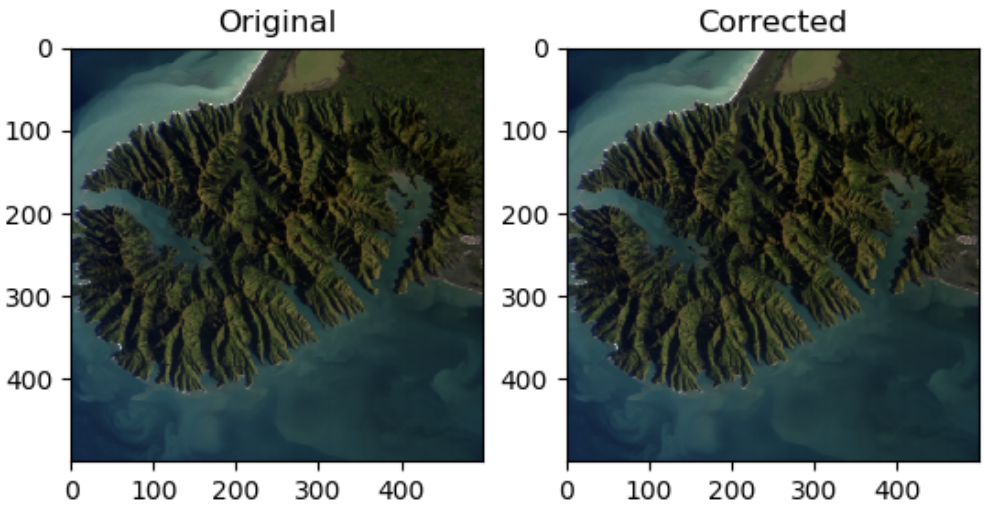

Figure 6: Pseudo RGB image of the atmosphere corrected image

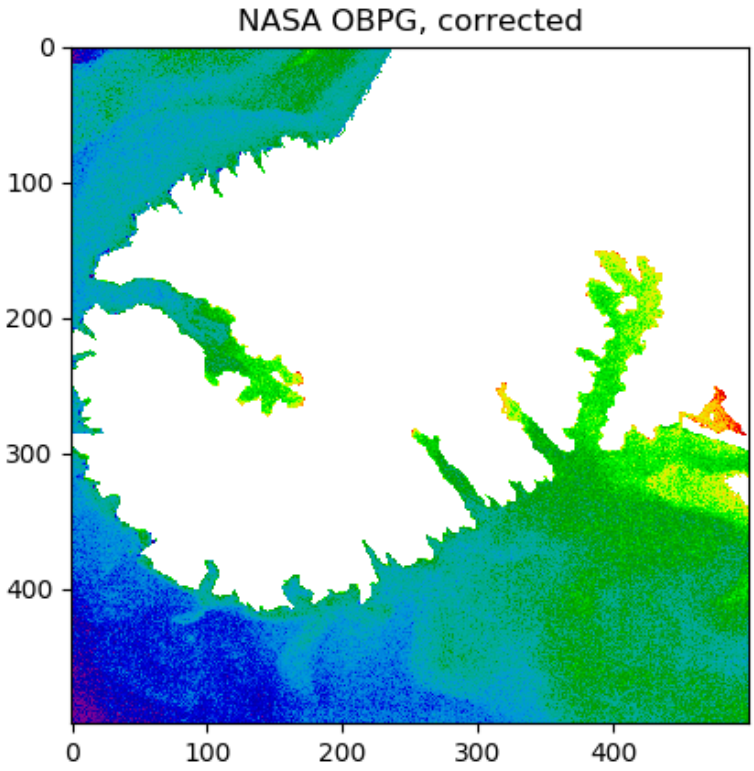

Figure 7: Results from the NASA OBPG algorithm using the atmospheric correction described in section 2.3, showing chlorophyll concentrations in the water

2.4 Compute chlorophyll concentration using atmosphere-corrected data

After performing the atmospheric correction, as described in section 2.3, we perform the NASA OBPG algorithm again on the corrected image cube. As we can see from the results in fig. 7, it is quite different from the original results from fig. 5. The strong concentration we found at the north west coast of the original (fig. 5), is no longer to be seen in the corrected image. Instead, we see a high concentration of chlorophyll on the south eastern coast, as well as a small patch in the bay on the western part of the landmass.

Rewrite after fixing images, colors are wrong

2.5 Classify land versus water

We want to classify only the chlorophyll content of the water, we're not interested in looking at the land. Therefore we want to mask out the landmass and show only the water. We again use K-means clustering to classify the data into different classes. By performing it multiple times for varying numbers of classes and comparing the result from fig. 4, fig. 8, as well as all other iterations from 1-10 classes on both corrected and original data, we find that performing k-means clustering with 10 clusters on the atmospherically corrected image cube gives the best performance regarding classifying

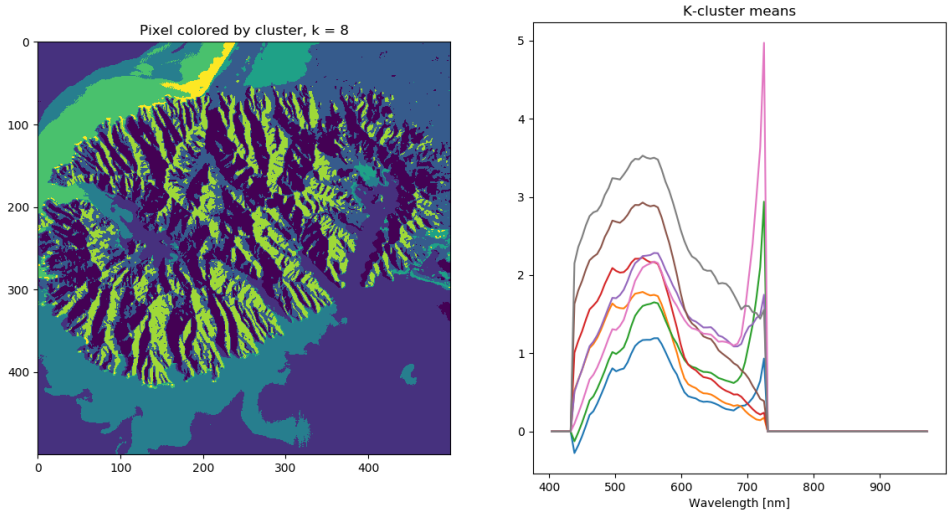

Figure 8: K-mean clustering performed on the atmospheric corrected image cube

water vs land. Setting the max number of iterations to 100, the algorithm converges after around 80 iterations, giving a good result.

After the k-means clustering is performed, we look at the spectral plot of the different classes. We know from section 1.4 roughly how the spectra of water looks compared to land/vegetation. We the pick the spectral lines from the k-means result that most resemble the spectral lines of water, and we remove any lines that resemble those of land/vegetation. We then end up with a handful of different classes that represents water, and the rest is land. We set the pixel values of all points inside the water classes to 1, and all other pixels to 0 to create the mask. The mask itself is shown in fig. 10. When put over the image, we get fig. 9.

Colors are wrong, fix code

2.6 Other bio-geophysical parameters

In the available spectral measurements between 400-1000nm we may find several other kinds of bio-geophysical parameters. In general, bio-geophysical parameters include biological (plant species, interactions in the ecology, biotic productivity), geological (soil types, erosion), and physical (light, heat) [5]. Examples include detecting phytoplankton pigments and gelbstoff [6], or land degradation processes in semi-arid geographical areas [7].

2.7 Alternative atmospheric correction methods

One of the drawbacks of using the empirical line method (ELM) is that it requires at least one field, laboratory, or other reference spectrum, preferably 2 or more like the 2 reference spectra used in this report (shallow and deep water) to create the linear regression used in removing the solar- and atmospheric path radiance. If such reference spectra, for all required wavelengths, is not available for the region of interest, this method can not be used. [8]

There exists a whole lot of different atmospheric correction tools and algorithms. Harris Geospatial

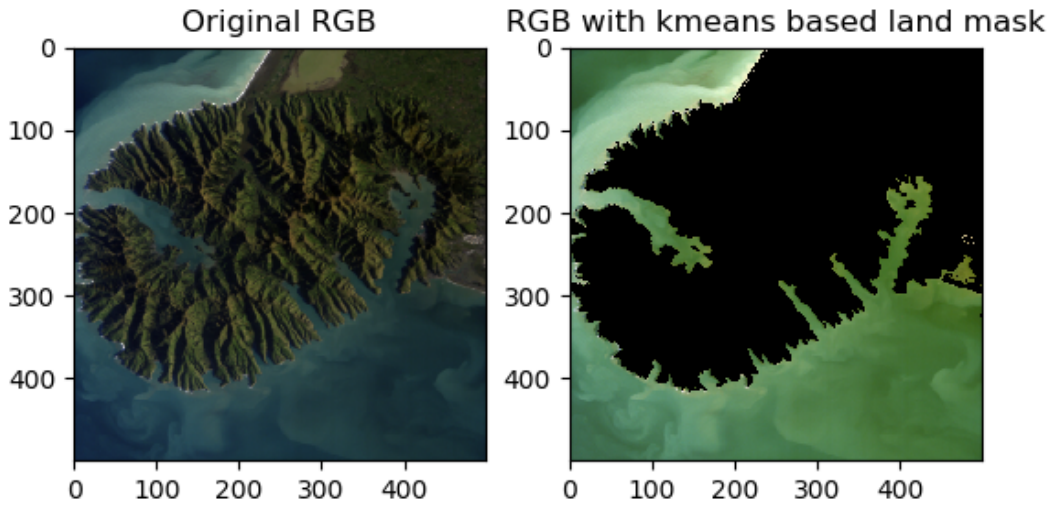

Figure 9: Masking away the land area

lists 9 different tools available for ENVI [9], including Dark Subtraction, Empirical Line, FLAASH, Flat Field, IAR, Log Residuals, QUAC, Thermal Atmospheric Correction, and Convert to Emissivity and Temperature.

For instance, the Flat Field Correction algorithm can be easily implemented by normalizing the images to an area of known "flat" reflectance, where the average spectrum from a region of interest can be used as reference.

3 Dimensionality Reduction & Noise Filtering

The HICO image cube is converted to a matrix on the form L x N, with L spectral bands rows, and $N = WH$ columns.

3.1 What is dimensionality reduction?

Dimensionality reduction is the act of reducing the amount of random variables to consider in for instance machine learning and statistics, involving feature selection and feature extraction. Thus making the data smaller and making analyzing it faster and easier.

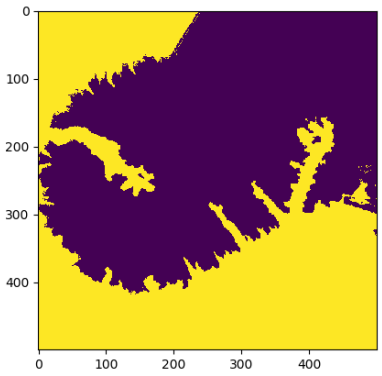

Figure 10: The mask found in section 2.5, based on kmeans clustering

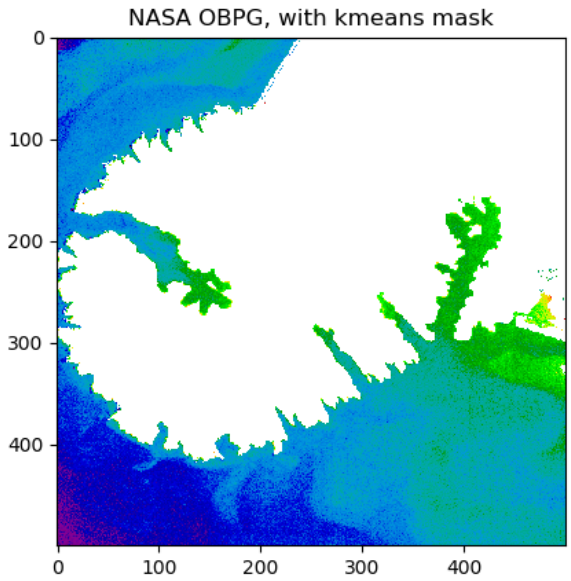

Figure 11: Running the OBPG algorithm on the masked image, using mask from fig. 10

Considering fig. 3, we see that there are quite a few similarities in the spectra of the various points. Looking at the two points in the water (shallow and deep), we see that the spectra are almost identical to each other for wavelengths between around 400 - 470nm, except for the clear difference in the amplitude. Even the spectra for the vegetation seem to share this characteristic. For wavelengths between 470 - 800nm all spectra seem to have unique signatures, until the two points in the water again seem to share quite similar spectra for wavelengths over 800nm, with only small variations between the two. An other thing to consider is the fact that the atmospheric correction algorithm used in section 2.3 is invalid for wavelengths outside the range 438 - 730nm, so these invalid bands can be dropped. Thus there seems to be some opportunities to reduce the dimensionality in the spectral dimensions for the aforementioned wavelengths.

In the spatial direction, we could for instance ignore all of the landmass, as we are only interested in looking at the water. This can be done as described in section 2.5, where all points on land are simply set to 0, thus reducing the spatial dimension significantly by removing a big chunk of the image. The spatial dimension can be reduced further by doing a nearest neighbor approach to cluster data together, for instance in the deep water in the upper left part of the image, fig. 2, where a portion of the ocean seem to be very similar in appearance. This can be done using for example k-means clustering, as described previously in section 2.1.

3.2 Principal Component Analysis (PCA)

PCA is a dimensionality reduction technique that transforms the columns of a dataset into a new set features, by finding a new set of directions that explain the maximum variability in the data [10]. These new coordinate axes/directions are known as the Principal Components (PCs). The dataset

columns contains the amount of variance of the data, computing the PCs will help explain the vast information in the original data in a fewer amount of columns.

If we do PCA on a matrix X with 100 rows and 1000 columns, using 10 basis vectors the PCA components will have the following shapes:

- Original data shape: (100, 1000)
- PCA dataframe shape: (10, 1000)
- PCA weights shape: (100, 10)

Meaning we have 10 weights per column, $10 \times 1000 = 10000$ weights in total.

The weights in the PCA is the eigenvectors of X. Each principal component is the dot product of its weights and the mean-centered data (each column of X is subtracted from its own mean, so the mean of each column is zero).

$$PC_i = weights \cdot X_{meancentered} \tag{2}$$

3.3 How does dimensionality reduction via PCA afect classification?

PCA was implemented using the existing PCA algorithm from *sklearn.decomposition*. First we convert the HICO image cube into matrix form, as described in section 3, We then run the PCA with $P = 10$ principal components on this data matrix, returning an $P \times N$ size matrix representing the compressed data. That is 10×250000 in size, down from 100×250000, a ten times reduction in size. Looking at the variance of the 10 principle components, shown in eq. (3) and plotted in fig. 12, we clearly see that all the variance in the dataset is contained in the first 3 principle components, with a total of 91% being in the first component alone. Thus we could potentially reduced the dataset even further, down to only 3 components, without losing any significant data.

$$[0.91, 0.08, 0.01, 0, 0, 0, 0, 0, 0, 0,] \tag{3}$$

Running K-means clustering on the PCA compressed data we notice a great increase in performance. Clustering the compressed data, running kmeans for 1-10 classes and max 50 iterations, now only takes about 53 seconds, compared to the 128 seconds it takes to classify the uncompressed data. The resulting clustering can be seen in fig. 13, which looks more or less identical to the clustering from the original data in fig. 4. From the cluster means plot (right on fig. 13) we see that there is almost no variation for $PC > 2$, except for the one outlier.

Reducing the number of PCs to $P = 3$ we get even faster performance, using only 22 seconds to do the classification. The resulting image is identical to fig. 13, as all the principal component containing all the variance/information in the dataset is still kept, even though the data is significantly more compressed.

If the PCA is done using 100 principle components instead, we get a negative impact on the performance. Now the same kmeans clustering spends 145 seconds, while giving the same results as with 10 or even 3 components.

Turning P even lower, to $P = 1$, the kmeans finishes after only 15 seconds. However, we have lost some information. It is no longer possible to classify the data correctly as we have removed too much information. Some noticeable differences here is the inability to separate the different classes of the

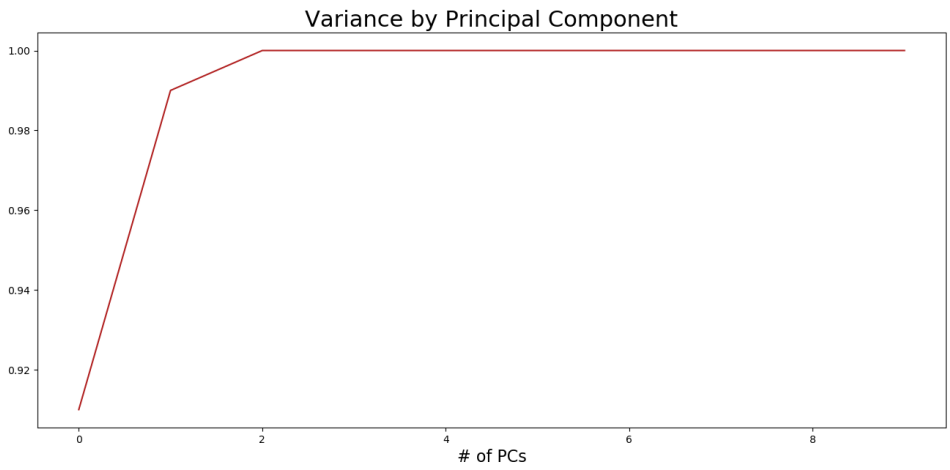

Figure 12: Plotting the cumulative variance of the principal components

ocean, as there is no longer individual classes for shallow and deep water, but just one class in total for the ocean. Thus, this data is now useless for our purposes.

3.4 Maximum Noise Fraction

Maximum Noise Fraction, as described in [11], is summarized here and in [4]. It is also known as Noise Adjusted Principal Components, and will here be used to remove additive noise on the image.

$$Cov(X) = \Sigma = \Sigma_s + \Sigma_n \tag{4}$$

Σ_s is the covariance of the signal and Σ_n the covariance of the noise. We find the eigendecomposition of $\Sigma_n\Sigma^{-1}$ to find the left-hand eigenvectors A^T and the right hand eigenvectors V. We then use these to calculate $\hat{X}$ for eq. (6).

$$Y = A^T X \tag{5}$$

$$\hat{X} = V_{(:,1:P)}Y_{(1:P,:)} \tag{6}$$

The MNF algorithm is tested on 2 different test cases, with 1 and 2 components. The result from case 1 with 1 component is shown in fig. 14, and result from case 2 with 2 components in fig. 15.

For the first case, we find that using only one component gives the best result, for both PCA and MNF. Using 2 components, we get an average pixel error of 47.4% for MNF and 47.4% for PCA, whilst with only one component, the error becomes 37.2% and 37.2% for MNF and PCA both. This implies that parts of the variance of the noise may be contained into the principle component that gets dropped when only keeping one component. The overall results are very similar, there is no clear difference between the MNF and PCA results. For comparison, the mean error of the unfiltered image is 59.6%, so it is clear that both algorithms improves the error.

For case 2 however, we find that using 1 component works better for PCA, but 2 components work better for MNF. Using 1 component we get 44.1% for MNF and 40.0% for PCA, but when using

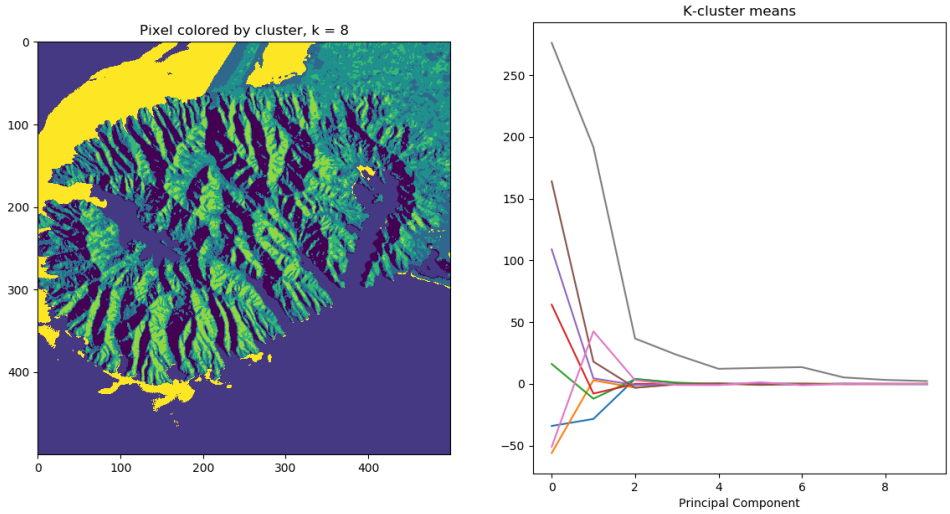

Figure 13: K-means classification performed on the PCA compressed data, using 10 Principal Components.

$P = 2$ we get an error of only 4.9% for MNF while PCA has 46.3% error. In this case, the MNF manages to remove almost all of the noise from the image, giving a result that looks almost identical to the original image.

3.5 Maximum Noise Fraction on HICO noisy

To do MNF on the HICO_noisy dataset, we first have to estimate the noise component. This is done using the between-neighbor difference, assuming neighboring pixels have similar spectra and thus the difference between the neighbors is the noise.

$$X_n = [x_1 - x_2, x_2 - x_3, \ldots, x_{N-1} - x_N] = [\hat{n}_1, \hat{n}_2, \ldots, \hat{n}_{N-1}] \tag{7}$$

After estimating the noise component, we find the covariance Σ_n of it. The *HICO_noisy* image cube is then ran through the MNF algorithm using various different amount of components. PCA is also performed, using the same number of components. The result can be found in table 1

P	**100**	**50**	**30**	**10**	**5**	**3**	**1**
MNF	0.1023	0.0724	0.0561	0.0325	0.0274	0.0312	0.2424
PCA	0.1023	0.0729	0.0566	0.0330	0.0254	0.0268	0.1048

Table 1: Comparing the results of using MNF and PCA on the HICO_noisy dataset.

As we can see from the table, the difference between MNF and PCA is quite small. They perform very similarly for most Ps, but they have some differences for smaller P. Using $P = 1$ we find the largest difference, where PCA show an error of 10.48% while MNF show an error of 24.24%. Both PCA and MNF have the smallest error for $P = 5$, where they get 2.54% and 2.74% respectively.

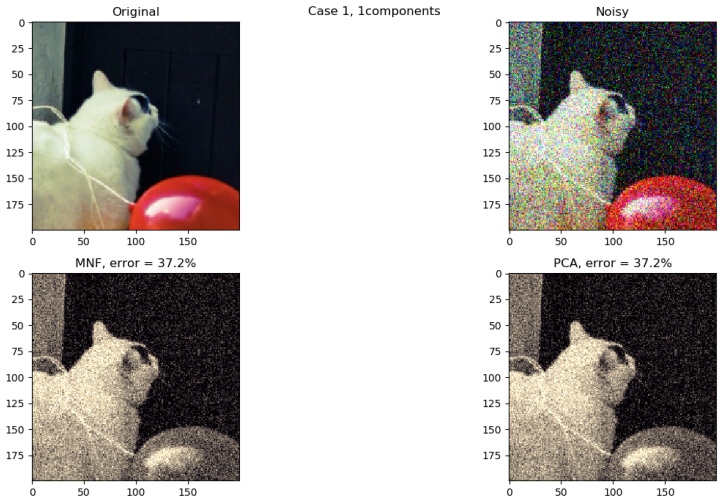

Figure 14: Maximum Noise Fraction and Principal Components analysis performed on test case 1, with 1 component used

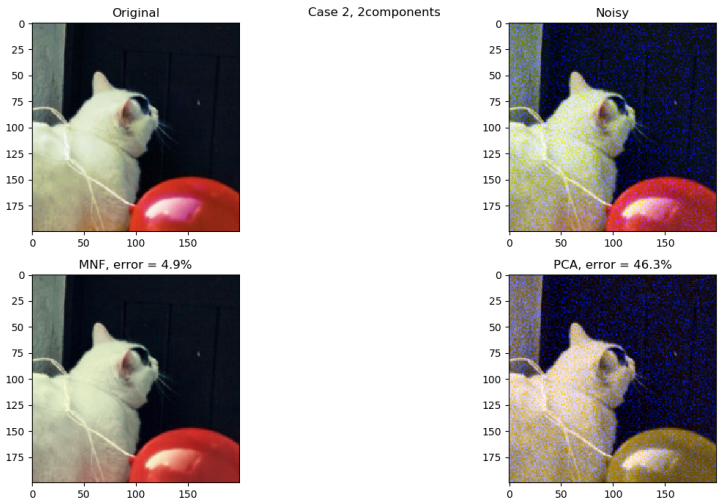

Figure 15: Maximum Noise Fraction and Principal Components analysis performed on test case 2, with 2 component used. Between-neighbor difference is used to approximate the noise covariance used in the MNF.

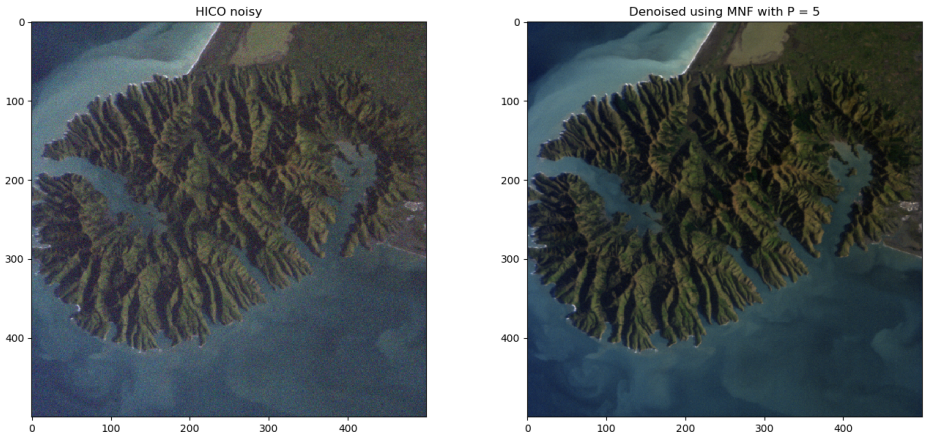

Figure 16: Removing the noise from the noisy HICO image (left) using Maximum Noise Fraction (right)

The RGB representation of the HICO_noisy data can be seen in fig. 16. Here the MNF has been performed using 5 components, which gave the smallest error. As can be seen, the noise has been significantly reduced.

3.6 Discussing the results

An assumption of PCA is that we have a reasonably high signal to noise ratio. Thus, if that assumption is true, PCA would be optimal. If the signal to noise ratio is low however, MNF would perform better. Also, looking at the results on the test image case 2 in section 3.4, fig. 15, the noise in the image seems to be only one single color, all the noise seems to be blue, compared to case 1 where the noise is random colors. In this case (case 2), MNF outperforms PCA significantly. But then again, the signal to noise ratio for case 2 is also lower than it is for case 1, so the result might very well be due to that fact, which is supported by the assumption given earlier.

The Maximum Noise Fraction (MNF) is similar to the Principal Components Analysis (PCA), with the main difference being that the Principal Components associated with the MNF are ordered by descending signal-to-noise ratio rather than overall image variance.

3.7 How can we best use the subspace?

4 Fun but definitely hard problems

4.1 Deep learning

Hyperspectral images contains very large amounts of data, so using a fully connected deep learning network would be unfeasible due to the enormous amount of weights and nodes required to pass the full-scale image cube. Using this HICO set as an example, with dimensions $500 \times 500 \times 100$, totalling 25.000.000 datapoints for the input layer alone. This input layer would have to be connected to

another layer with N nodes, requiring $N \times 25000000$. And thats just for the first layer. If more layers are used, we are going to run out of memory fast.

Using a convolutional neural network may be an option. Applying convolutional filters on the input image, we may reduce the number of input parameters significantly, though at the cost of increased computational complexity.

Another option may be to simply reduce the input image size. This may be done using the dimensionality reduction techniques discussed in section 3. Alternatively, we may crop the original image into smaller subsets of images, each image containing all spectral dimensions (the various wavelengths), but only a smaller image in the spatial dimension. Similarly, the images can be cropped in the spectral dimension, removing any uninteresting spectral bands and keeping only the bands we are interested in studying.

4.2 Multispectral-hyperspectral image fusion

4.3 Spatial-spectral methods

4.4 Locating methane emissions

References

[1] Wikipedia. *Spectral sensitivity*. Jan. 2020. URL: https://en.wikipedia.org/wiki/Spectral_sensitivity.

[2] Robert L. Lucke et al. "Hyperspectral Imager for the Coastal Ocean: instrument description and first images". In: *Appl. Opt.* 50.11 (Apr. 2011), pp. 1501–1516. DOI: 10.1364/AO.50.001501. URL: http://ao.osa.org/abstract.cfm?URI=ao-50-11-1501.

[3] Thomas Boggs. *SpectralPython*. Jan. 2014. URL: https://www.spectralpython.net.

[4] Sivert Bakken, Joe Garret, and Simen Haugo. "Hyper Spectral Imaging Project". In: *TTK4255 Robotic Vision, NTNU* (Feb. 2020). URL: https://ntnu.blackboard.com/bbcswebdav/pid-881058-dt-content-rid-25343529_1/xid-25343529_1.

[5] ESA Sentinel Online. *Bio-geophysical Variable Mapping*. URL: https://sentinel.esa.int/web/sentinel/thematic-areas/land-monitoring/bio-geophysical-variable-mapping.

[6] ZhongPing Lee, Kendall L. Carder, and Robert A. Arnone. "Deriving inherent optical properties from water color: a multiband quasi-analytical algorithm for optically deep waters". In: *Appl. Opt.* 41.27 (Sept. 2002), pp. 5755–5772. DOI: 10.1364/AO.41.005755. URL: http://ao.osa.org/abstract.cfm?URI=ao-41-27-5755.

[7] H. Kaufmann et al. "SAND - a hyperspectral sensor for the analysis of dryland degradation". In: *IEEE International Geoscience and Remote Sensing Symposium*. Vol. 2. 2002, 986–988 vol.2.

[8] Evanthia Karpouzli and Tim Malthus. "The empirical line method for the atmospheric correction of IKONOS imagery". In: *International Journal of Remote Sensing - INT J REMOTE SENS* 24 (Mar. 2003), pp. 1143–1150. DOI: 10.1080/0143116021000026779.

[9] Harris Geospatial Solutions. *Atmospheric Correction*. Jan. 2020. URL: https://www.harrisgeospatial.com/docs/atmosphericcorrection.html.

[10] Selva Prabhakaran. *Principal Component Analysis (PCA) – Better Explained*. Mar. 2019. URL: https://www.machinelearningplus.com/machine-learning/principal-components-analysis-pca-better-explained/.

[11] A. A. Green et al. "A transformation for ordering multispectral data in terms of image quality with implications for noise removal". In: *IEEE Transactions on Geoscience and Remote Sensing* 26.1 (1988), pp. 65–74.