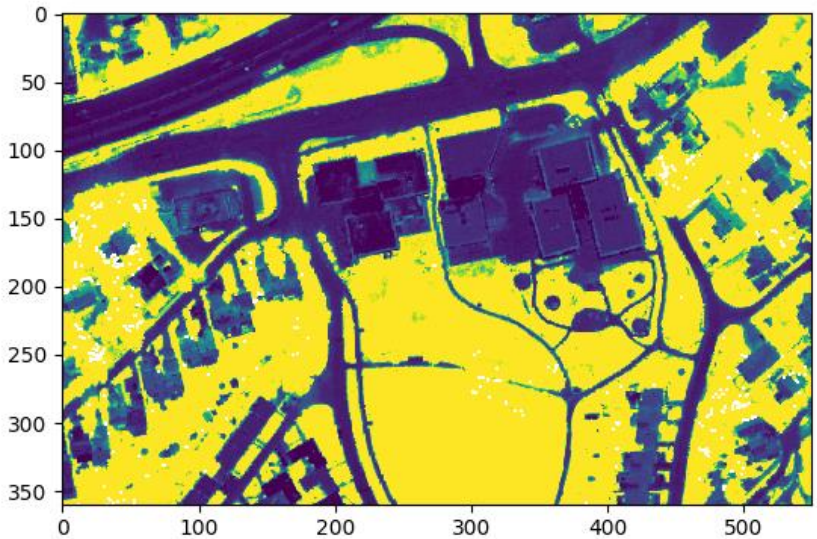
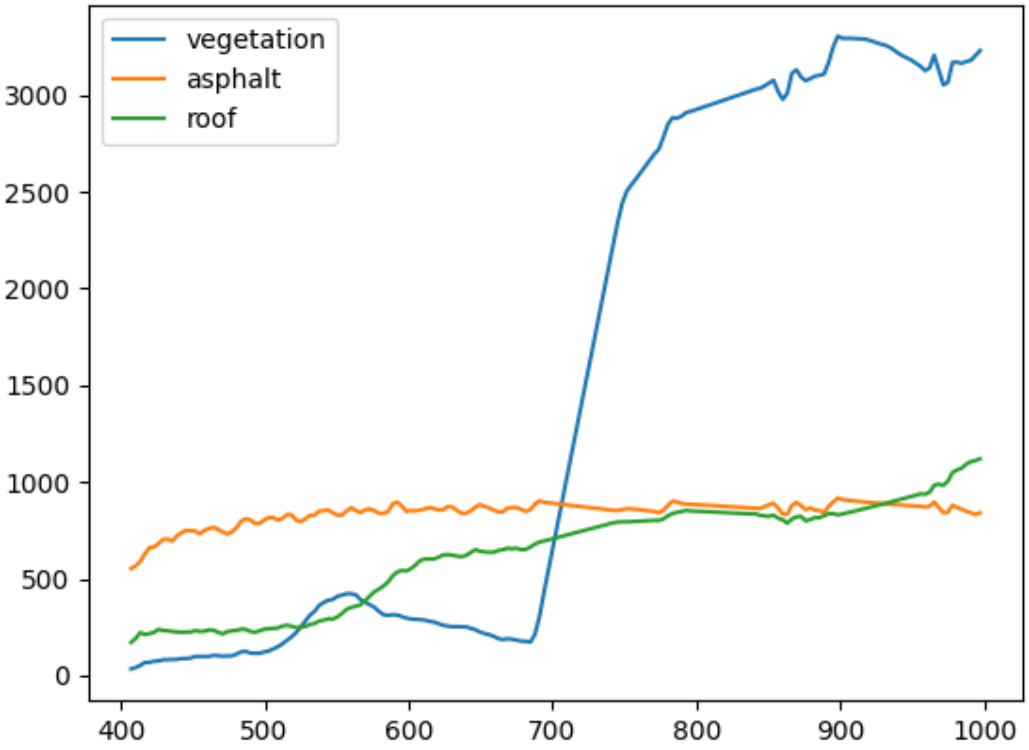
Assignment 3

3. Display an RGB image from the hyperspectral image and included it in your report.

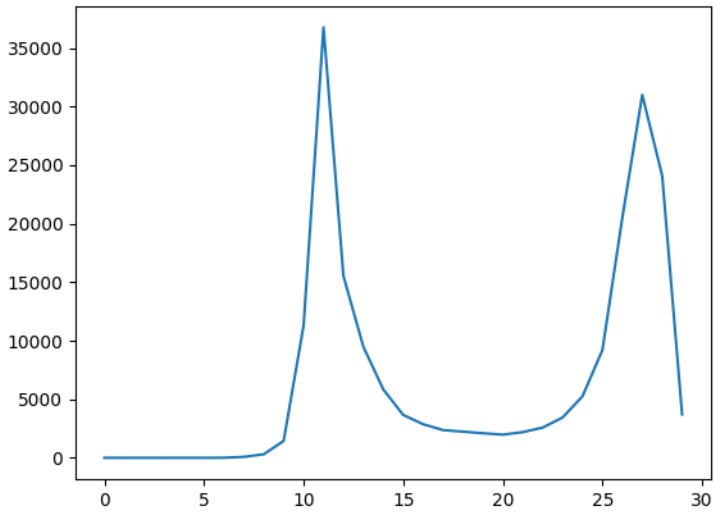
5. Display the NDVI image in a sensible range and include it in the report.



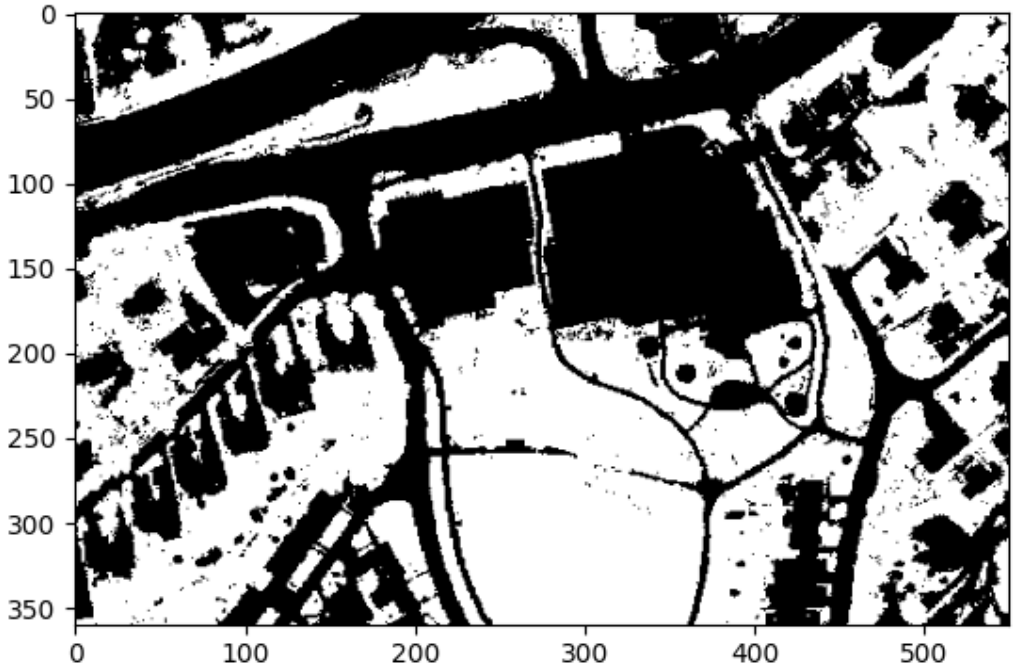
6. Select a point with vegetation, one with asphalt and one with a roof and plot the spectra and show them in the report.



7. Compute and display a histogram of all the NDVI values in the image, include it in the report.



8. Make a threshold of NDVI > 0.6 and set all values with NDVI > 0.6 to zero. Display this image. What do you see ? (include in report)

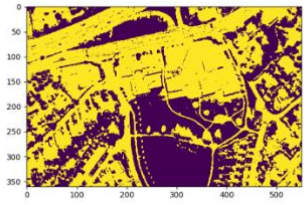


Here vi can easily see the difference between vegetation as white and asphalt/rooftops as black .

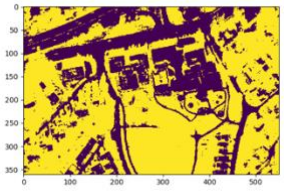
9. Can you use this information to determine the fraction area of the image that has vegetation ?

We can calculate how many of the pixels that are one, and how many that are 0. Then divide the amount of 1 (vegetation) on the total amount of pixels. We then get a fraction of 0.49863131313131315 % vegetation.

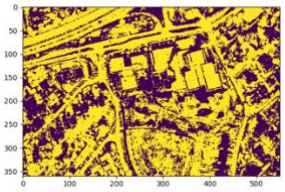
10. Carry out a principal component analysis (PCA) of the image, using the PCA function from the Spectral Python. Study the score images and loading plots. Can any of the first 3 components be used to identify vegetation in the image ? Include the first 3 score images and loading plots in your report.



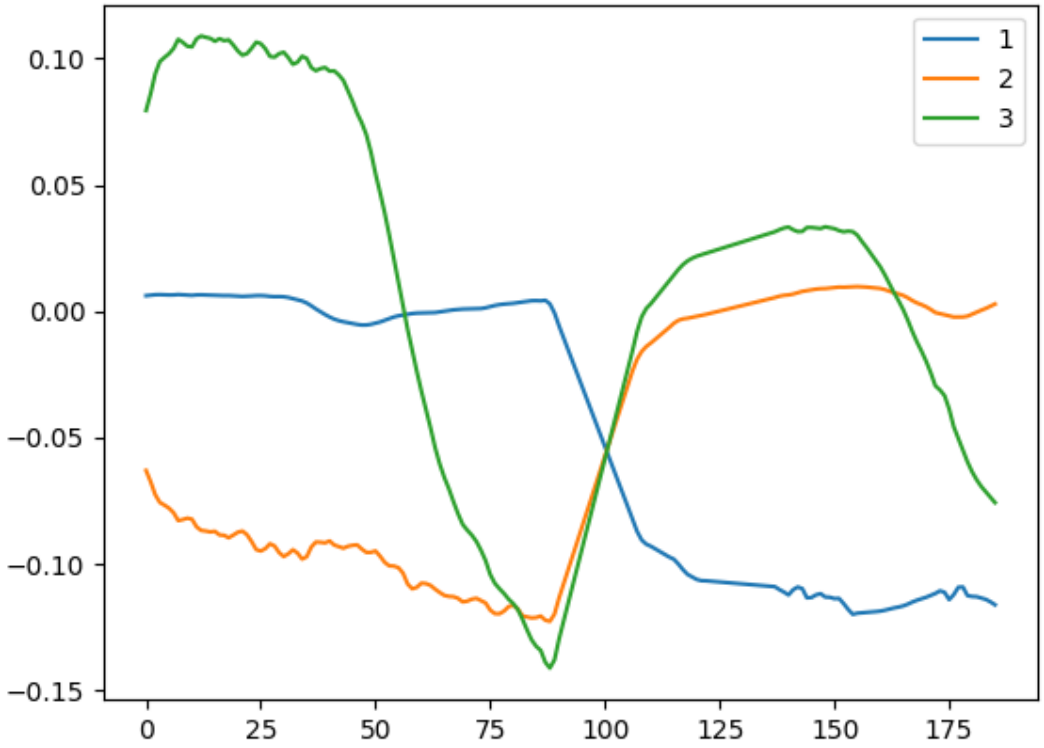
3.



2.

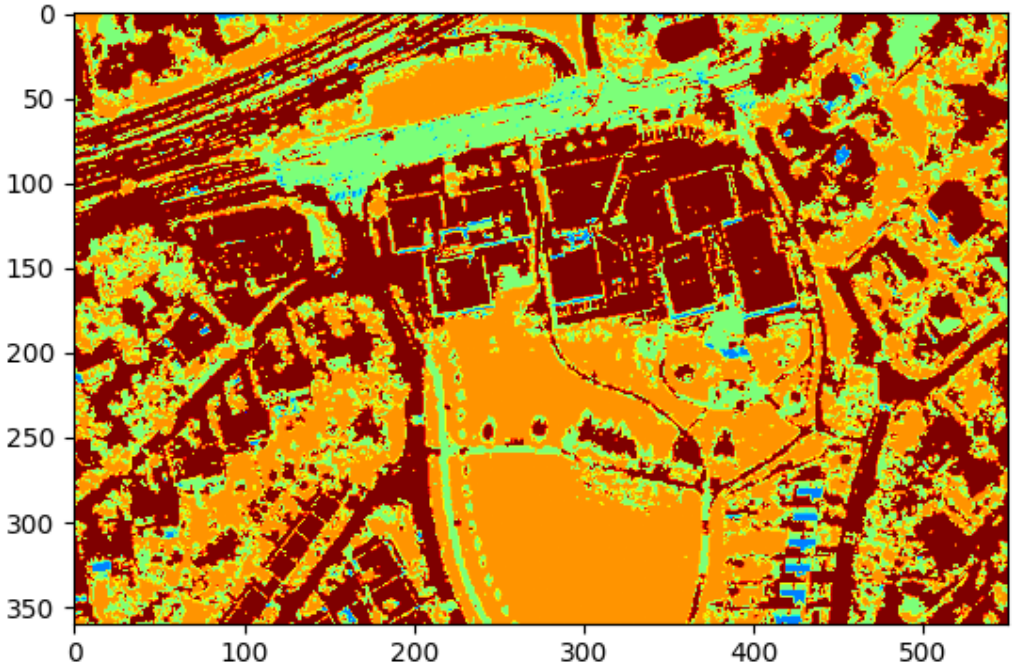


1.



We can see that image 1 is the best to spot vegetation. We can see the small dip around the band of green and we can see that there are very low values around the infrared values. All this indicating that there are grass. This image is the same as the vegetation spectra found earlier in the task but now its flipped.

11. Carry out a k-means clustering on a selected number of principal component score image, try k-means with 2, 3, 4 and 5 classes. How many components would you chose in order to group all the pixels with vegetation ? Include one of the k-mean calculations in the report.
The more components the more accurate grouping for the vegetation. But I would at least add 3. This is because if we look at the eigenvalue we see that it flattens after the 2 one. This means that there are no big chances and we get most of the information from the 2 first ones.
K means image with 3 components from PCA.



12. Which method to determine the amount of vegetation do you think is most appropriate in this hyperspectral image ? Describe why you prefer one method before another.

NVDI is the most appropriate. It is a standard procedure for vegetation and it gives a degree for vegetation, not just a yes and no answer. It is also the fastest to compute.

Code used in this assignment

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from spectral import *
import numpy as np
import matplotlib.pyplot as plt

# %% Part1 Load image

hyperim = np.load("sandvika.npy")
wavelength = envi.read_envi_header('Visnir.hdr')['wavelength']
ww = [float(i) for i in wavelength]

# %% Part2 Defines band numbers

def color_band(image_band, wavelength_in_function):
    diff_band = []
    for i, value in enumerate(image_band):
        index_tuple = (i, abs(value - wavelength_in_function))
        diff_band.append(index_tuple)
```

```
    closest = sorted(diff_band, key=lambda t: t[1])[0]

    return closest[0]


bands = {
    'blue': color_band(ww, 440),
    'green': color_band(ww, 535),
    'red': color_band(ww, 645),
    'NIR': color_band(ww, 800)}

print("bands = ", bands)
band_red = hyperim[:, :, bands['red']]
band_green = hyperim[:, :, bands['green']]
band_blue = hyperim[:, :, bands['blue']]
band_nir = hyperim[:, :, bands['NIR']]

# %% Part3 Display an RGB image

imshow(hyperim, (bands['red'], bands['green'], bands['blue']),
       stretch=((0.02, 0.98), (0.02, 0.98), (0.02, 0.98)))

plt.show()

# %% Part4 Computes the NDVI

def calculate_NDVI(NIR, Red):
    NDVI = (NIR - Red) / (NIR + Red)
    return NDVI

ndvi_ima = calculate_NDVI(band_nir, band_red)

# %% Part5 Display the NDVI image

plt.imshow(ndvi_ima, vmin=0, vmax=0.7)
plt.show()

# %% Part6 Select a point with vegetation, asphalt and roof

vegetation = np.array(hyperim[300, 300, :].reshape(-1, 1))
asphalt = np.array(hyperim[75, 5, :].reshape(-1, 1))
roof = np.array(hyperim[350, 424, :].reshape(-1, 1))
plt.figure()
plt.plot(ww, vegetation)
plt.plot(ww, asphalt)
plt.plot(ww, roof)
plt.legend(['vegetation', 'asphalt', 'roof'])
plt.show()

# %% Part7 Histogram of all the NDVI
ndvi_ima = np.nan_to_num(ndvi_ima)
ndvi_histogram, bins = np.histogram(ndvi_ima, bins=30)
plt.plot(ndvi_histogram)
plt.show()

# %% Part8 Make a threshold of NDVI

ndvi_thresholded = ndvi_ima[:]
th = 0.6
ndvi_thresholded[ndvi_thresholded < th] = 0.0
ndvi_thresholded[ndvi_thresholded >= th] = 1.0
```

```
imshow(ndvi_thresholded)
plt.show()

# %% Part9 Fraction area of the image that has vegetation
grass = 0
not_grass = 0
for i in range(len(ndvi_thresholded)):
    for j in range(len(ndvi_thresholded[0])):
        if ndvi_thresholded[i][j] == 1:
            grass += 1
        elif ndvi_thresholded[i][j] == 0:
            not_grass += 1

fraction = grass / (not_grass + grass)
print(fraction, "%")
"""
Nearly equal with grass and not.
"""

# %% Part10 Carry out a principal component analysis

pc = principal_components(hyperim)
pc_0999 = pc.reduce(fraction=0.999)
img_pc = pc_0999.transform(hyperim)
plt.imshow(img_pc[:, :, 0], vmin=-0.1, vmax=0.15)
plt.show()
plt.imshow(img_pc[:, :, 1], vmin=-0.1, vmax=0.15)
plt.show()
plt.imshow(img_pc[:, :, 2], vmin=-0.1, vmax=0.15)
plt.show()
loadings = pc_0999.eigenvectors
plt.plot(loadings[:, [0, 1, 2]])
plt.legend(['1', '2', '3'])
plt.show()

# %% Part11 Carry out a k-means clustering

# (m, c) = kmeans(img_pc, 2, 5)
# (m, c) = kmeans(img_pc, 3, 5)
# (m, c) = kmeans(img_pc, 4, 5)

(m, c) = kmeans(img_pc[:, :, 0:3], 5, 30)
plt.imshow(m, 'jet')
plt.figure()
for i in range(c.shape[0]):
    plt.plot(c[i])

imshow(m, stretch_all=True)
plt.show()

# %% Part12

"""
NVDI is the most appropriate. It is a standard procedure for vegetation
with known performace,
gives a degree (not just yes/no) answer. It is also the fastest to compute.
PCA would be second,
it can also give degree. Nice here is that it may highlight other
interesting aspects.
"""
```