ROYAL MILITARY ACADEMY
172 POL
Capt. Maxime Séverin

Academic year 2021-2022
2^{nd} Master

SE511 : Strategic Military Sensors
Hyperspectral unmixing

2EV COC REMACLE Maximilien

Bruxelles,
December 10, 2021

Contents

List of Figures

1. Introduction

When we want to observe something by taking an image of it, we often imagine an image that looks like what we would see with our own eyes. For that, we try to capture what is called the (visible) light. This is actually the portion of the electromagnetic radiation that is perceived by the human eye. These electromagnetic waves are in the "visible spectrum", which means they have a wavelength approximately between 400 and 700 nm. Figure 1.1 shows the visible spectrum and the colours associated with it. The majority of people actually perceive three different bands of wavelengths (or frequencies) that cover the visible spectrum: one higher wavelength band, the reddish colours, one intermediate wavelength band, the yellow-green colours and one lower wavelength band, the blue-violet colours. In a classical image or video, taken by a smartphone or a camera, for example, each element of the picture (pixel) contains three pieces of data, which are the intensities of the electromagnetic radiation in these three different bands that the human eye perceives. That is the reason why such images are sometimes called RGB (red-green-blue) images.

Figure 1.1: Visible spectrum (the values correspond to the wavelengths in nm) [4]

However, visible light is only a very small part of the whole electromagnetic spectrum. Figure 1.2 shows the electromagnetic spectrum, with the visible light highlighted. Some sensors are sensitive to radiation out of the visible spectrum. Multispectral and hyperspectral sensors capture light of the visible spectrum, but also in the ultraviolet (UV) and infrared (IR) domains. In these sensors, each pixel captures more than three different spectral bands. Multipectral sensor, for instance, have about a dozen spectral bands, that are rather wide ($\approx$ 100 nm). On the other hand, hyperspectral sensors have at least a hundred different spectral bands that are a lot narrower ($\approx$ 10nm). In this paper, we'll try to exploit some data captured by an hyperspectral sensor mounted on a plane and see if we can extract some information from these images, that will most certainly be different than what we get from classical RGB-images.

All the code and images used and showed in this paper are on the linked GitHub repository [3].

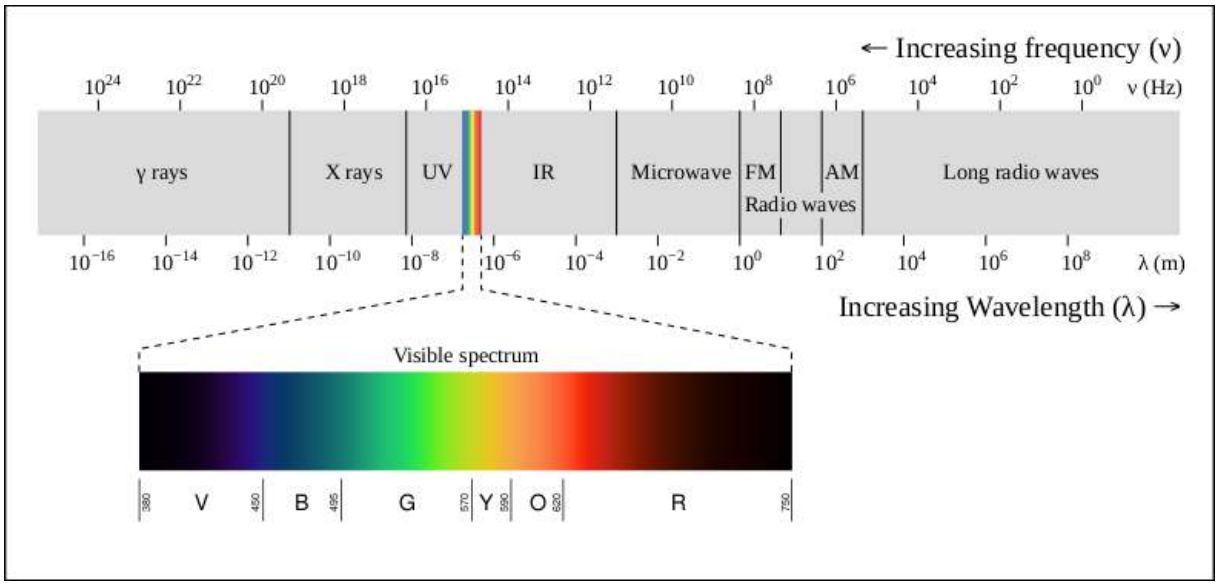

Figure 1.2: Electromagnetic spectrum [4]

2. Data

The data we will use was gathered during a flight over Pavia, Northern Italy. From this flight, two hyperspectral images are available on the Grupo de Inteligencia Computacional (GIC) website [2]. The first image is of Pavia center. It has a dimension of 1096 by 715 pixels[1] and 102 spectral bands. The second image is of Pavia University. It has a dimension of 610 by 340 pixels[2] and 103 spectral bands. In addition to the hyperspectral images, ground truth images are also provided for both scenes. The ground truth images are shown on figures 2.1 and 2.2. On the Pavia center image, we can clearly see the pixels that have been discarded by the provider of the images.

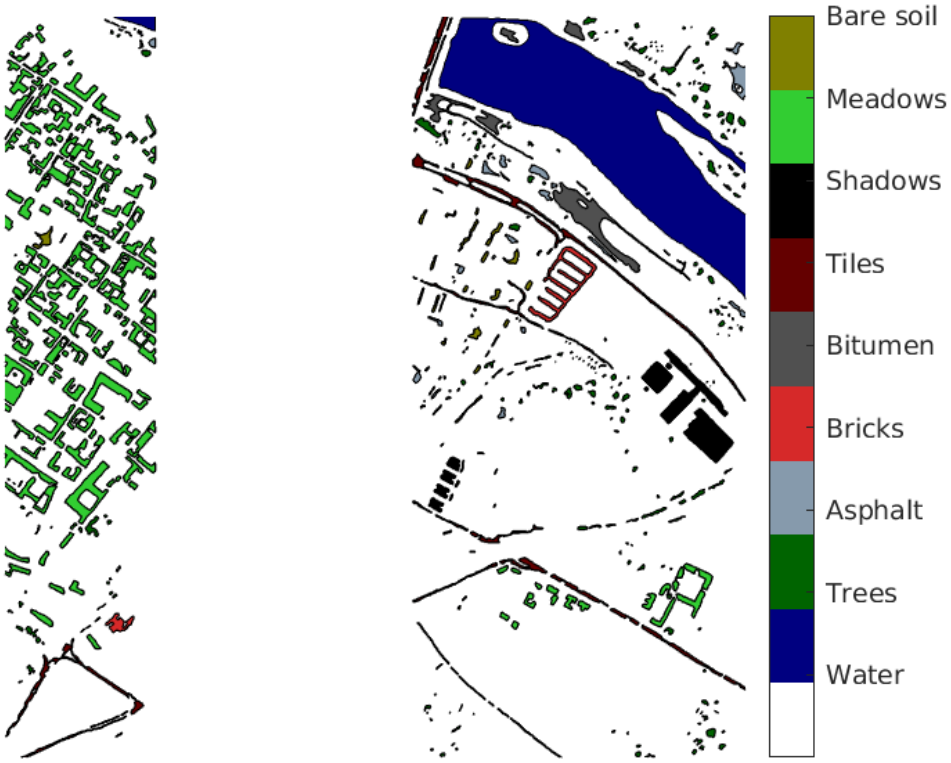

Figure 2.1: Ground truth, Pavia center

These two scenes were captured using a ROSIS sensor. On the Center for Remote Sensing (CRS) website [1], we read that: "*Reflective Optics System Imaging Spectrometer (ROSIS), first flights were performed in 1992, the current version ROSIS-03 is operated since 1999. The total field of view is ±8º and this sensor covers the spectral rangef rom 430 to 860 nm. The number of*

[1] originally 1096 by 1096, but some samples "*containing no information*"[2] have been discarded
[2] originally 610 by 610, but some samples "*containing no information*"[2] have been discarded

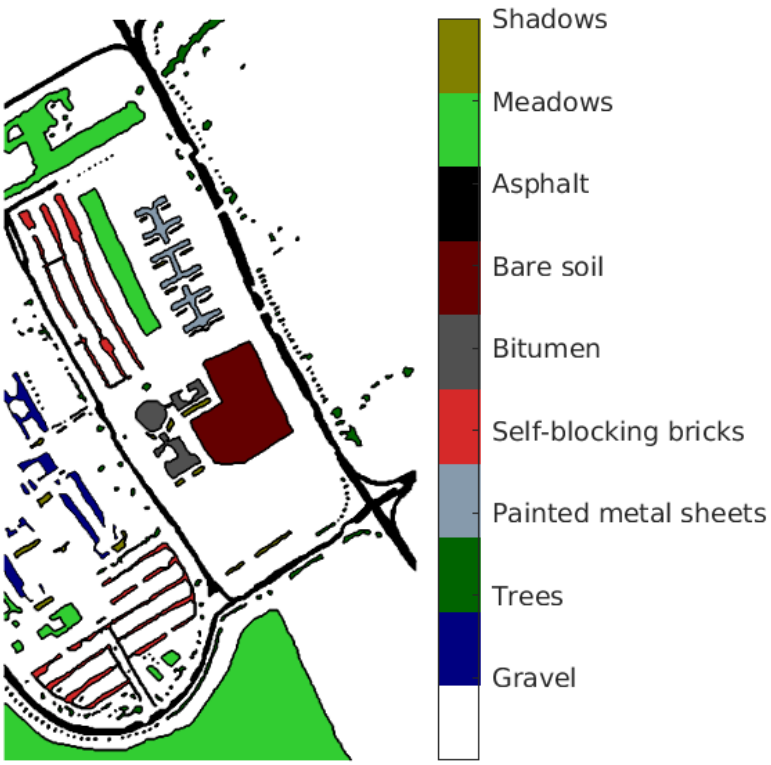

Figure 2.2: Ground truth, Pavia university

bands is 115with the band width is equal to 4.0 nm. The main objective of the ROSIS project is the detection of spectral fine structures especially in coastal waters. This task determined the selection of the spectral range, bandwidth, number of channels, radiometric resolution and its tilt capability for sun glint avoidance. However, ROSIS can be used just as well for monitoring of spectral features above land or within the atmosphere."

3. Exploitation of the data

The implementation of the methods described in this chapter is done in Matlab. The whole code is to be found in appendix B or on the GitHub repo of this paper [3].

3.1 Objective

Hyperspectral sensors have really narrow spectral bands. Hence, the energy received by a pixel for each separated band is weak. To compensate for this effect, the pixels are rather large and they cover each a wide area on the ground. Thus, small objects can't be seen, and each pixel picks up the reflections of several objects. The spectral signature of a pixel is then the addition of all spectral signatures of the different objects that are in the area the pixel covers. Furthermore, the observed spectrum also depends on other factors, such as: the incidence angle, the aging of the pixel or atmospheric effects. Our objective will then be to try to determine how much of each object we find in every pixel.

3.2 Notations

$$\vec{\boldsymbol{m}}_{\boldsymbol{k}} = [m_k(\lambda_1), m_k(\lambda_2), ..., m_k(\lambda_L)]^T \tag{3.1}$$

is called the endmember of element k. It is the spectral signature of a certain object k. L is the amount of spectral bands we get with our sensor.

$$\vec{\boldsymbol{r}} = [r(\lambda_1), r(\lambda_2), ..., r(\lambda_L)]^T \tag{3.2}$$

is the measurement at one particular pixel.

3.3 Linear Mixing Model

Assuming that the reflectance of a pixel is simply the addition of the reflectances of all the objects in this pixel, we can say that, with p different materials in a pixel:

$$\vec{\boldsymbol{r}} = \sum_{k=1}^{p} \alpha_k \vec{\boldsymbol{m}}_{\boldsymbol{k}} + \vec{\boldsymbol{n}} \tag{3.3}$$

where α_k is called the abundance of element k. It is the amount of element k in the pixel. Obviously, $0 \leq \alpha_k \leq 1$.

If we group all the α_k and $\vec{\boldsymbol{m}}_{\boldsymbol{k}}$, like this: $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, ..., \alpha_p]^T$ and $\boldsymbol{M} = [\vec{\boldsymbol{m}}_{\boldsymbol{1}}, \vec{\boldsymbol{m}}_{\boldsymbol{2}}, ..., \vec{\boldsymbol{m}}_{\boldsymbol{p}}]$, we can then write:

$$\vec{\boldsymbol{r}} = \boldsymbol{M}\vec{\boldsymbol{\alpha}} + \vec{\boldsymbol{n}} \tag{3.4}$$

Our objective is then to determine α_k for the element(s) we are interested in.

3.4 Project questions

Let's focus a moment on the task that has to be done. After having implemented it on Matlab, we'll discuss the results in the next chapter. The theory and the formulas used come directly from the course SE511 : Strategic Military Sensors. I don't find it relevant to explain it again in a paper written for this course.

3.4.1 Extracting the endmembers

First, we want to find the endmember of an element we are interested in. To do this, we have the hyperspectral image and the ground truth as well. The most straightforward way to go to approximate $\vec{\boldsymbol{m}_k}$ for a specific material k, is to simply take the average of the measurements $\vec{\boldsymbol{r}}$ for each pixel where the ground truth says that it is composed of element k.

3.4.2 Computing the covariance matrix and the resulting filter

Now, we need to compute the covariance matrix. Which is simply defined as:

$$\hat{\boldsymbol{R}} = \frac{1}{k}\sum_{k=1}^{K} \vec{\boldsymbol{r}_k}\vec{\boldsymbol{r}_k}^T \tag{3.5}$$

From there, we can calculate the resulting filter $\boldsymbol{w}$:

$$\boldsymbol{w} = \frac{\hat{\boldsymbol{R}}^{-1}\boldsymbol{d}}{\boldsymbol{d}^T\hat{\boldsymbol{R}}^{-1}\boldsymbol{d}} \tag{3.6}$$

3.4.3 Computing the abundance

Finally, to compute the abundance of the chosen element in a pixel, we just have to multiply the filter with the pixel measurement:

$$\hat{\alpha}_k = \boldsymbol{w}^T\vec{\boldsymbol{r}} \tag{3.7}$$

4. Results and discussion

4.1 Results

The results of the computed abundances are to be found in Appendix A. They are also available on the GitHub repository linked to this paper [3] for a better image quality.

The method explained in the previous chapter has been applied for each different element from the ground truth data, for both hyperspactral images. The result is an image of the scene with the abundance plotted as a colormap (light yellow : $\alpha = 1$, dark blue: $\alpha = 0$).

We can use the ground truth data to try to assess the quality of our results. We will take every different label from the ground truth label. For one specific material, we will take the abundance of this material between all pixels that have the corresponding label. So, we should expect high values (near to 1). The results are reported in tables 4.1 and **??**.

Ground truth	Mean abundance
Water	0.93443
Trees	0.85398
Asphalt	0.8812
Bricks	0.8487
Bitumen	0.90498
Tiles	0.91882
Shadows	0.85192
Meadows	0.89748
Bare soil	0.88454

Table 4.1: Mean abundances for Pavia center

Ground truth	Mean abundance
Asphalt	0.87886
Meadows	0.91513
Gravel	0.86931
Trees	0.85005
Painted metal sheets	0.89725
Bare soil	0.80948
Bitumen	0.90646
Self-blocking bricks	0.9
Shadows	0.90853

Table 4.2: Mean abundances for Pavia center

4.2 Discussion

In general, we see that we obtain rather high values for the abundances of the elements on the pixels that have the corresponding label in the ground truth image. The values are rarely below 85%. And visually, it is also easy to see that the abundance values are high where they should be, according to the ground truth.

We also see something else: there are a lot of other places where the abundances values are really high. Let's take the example of the self-blocking bricks at Pavia university. There, we see two zones where the ground truth says there should be this material (in red on the image), but there are also a lot of other places on the image where the abundance for these bricks is high. There could be several explanations to this:

- The ground truth is of poor quality
- The ground truth is based on classical RGB images and the hyperspectral unmixing method is able to detect smaller objects than it is possible with a human eye from above or with a classical camera or with any method that was used to generate the ground truth image.
- There are pixels with different materials on it, so it is possible that those pixels haven't received any labels on the ground truth image.
- There are other kinds of material there that have a similar spectral signature.

Given that those extra pixels with high abundances values often form some realistic shapes, it is probable that our hyperspectral unmixing method actually reveals some other places where some materials are located, even if the ground truth doesn't say so.
In Pavia center, we also see with the ground truth that there is a river. But when we compare the ground truth with the abundances values of the water, we clearly see on the image that part of the river is missing in the ground truth. So, it appears that sometimes, there is no label on a pixel of the ground truth image, while apparently there should.

5. Conclusion

In this work, we implemented the endmembers of different materials using some ground truth data and hyperspectral images. Then, we implemented a filter that allowed us to calculate the abundances of different elements on two images taken in Pavia, Italy. The result are rather convincing. We also noticed that the abundance values are also high where the ground truth doesn't specify they should be. But it is difficult to say if it is some imprecision in the method, if there are different yet very similar materials, or if the hyperspectral unmixing finds more of certain materials than the method used to create the ground truth data.

A. Results

A.1 Pavia Center

The rest of this page is intentionally left blank.

A.1.1 Water

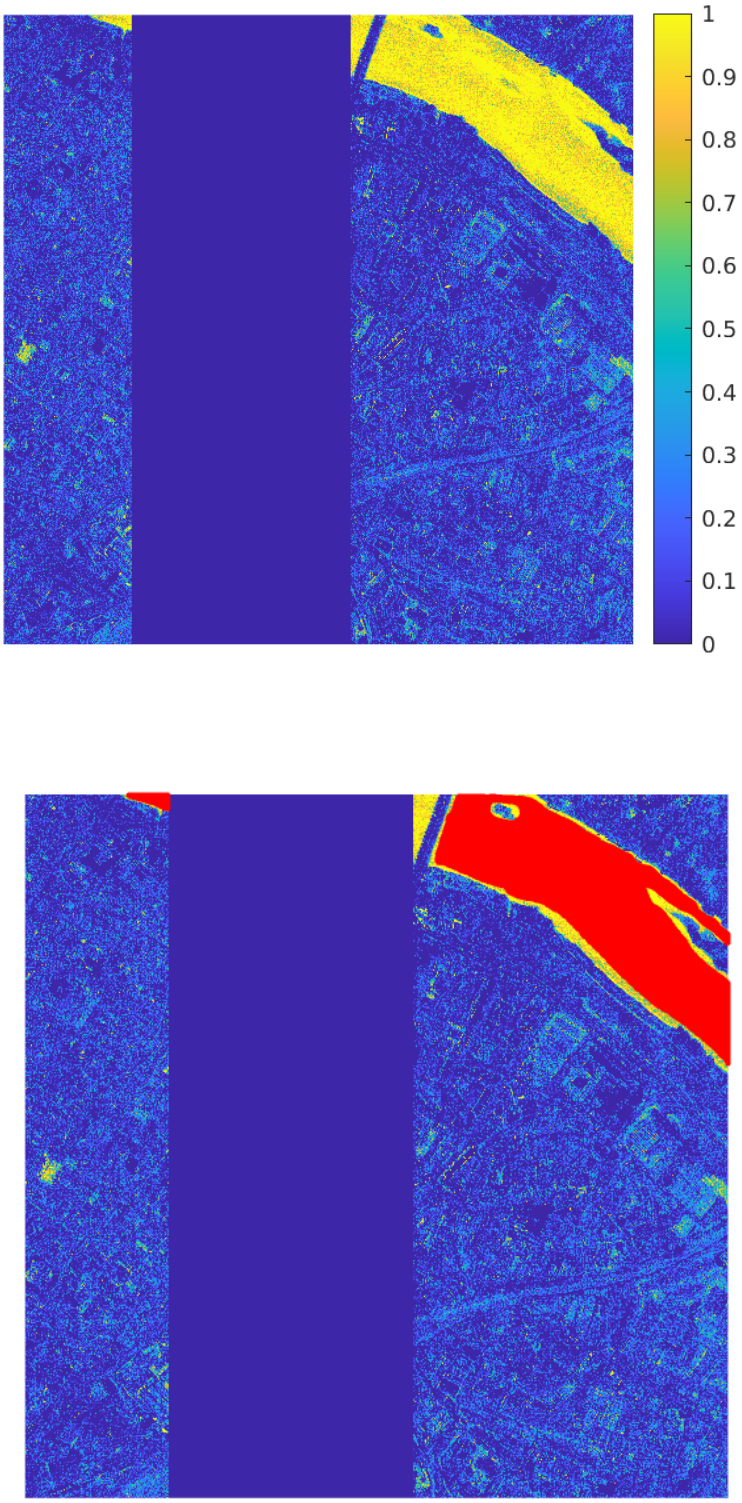

Figure A.1: Abundance of Water, Pavia center

A.1.2 Trees

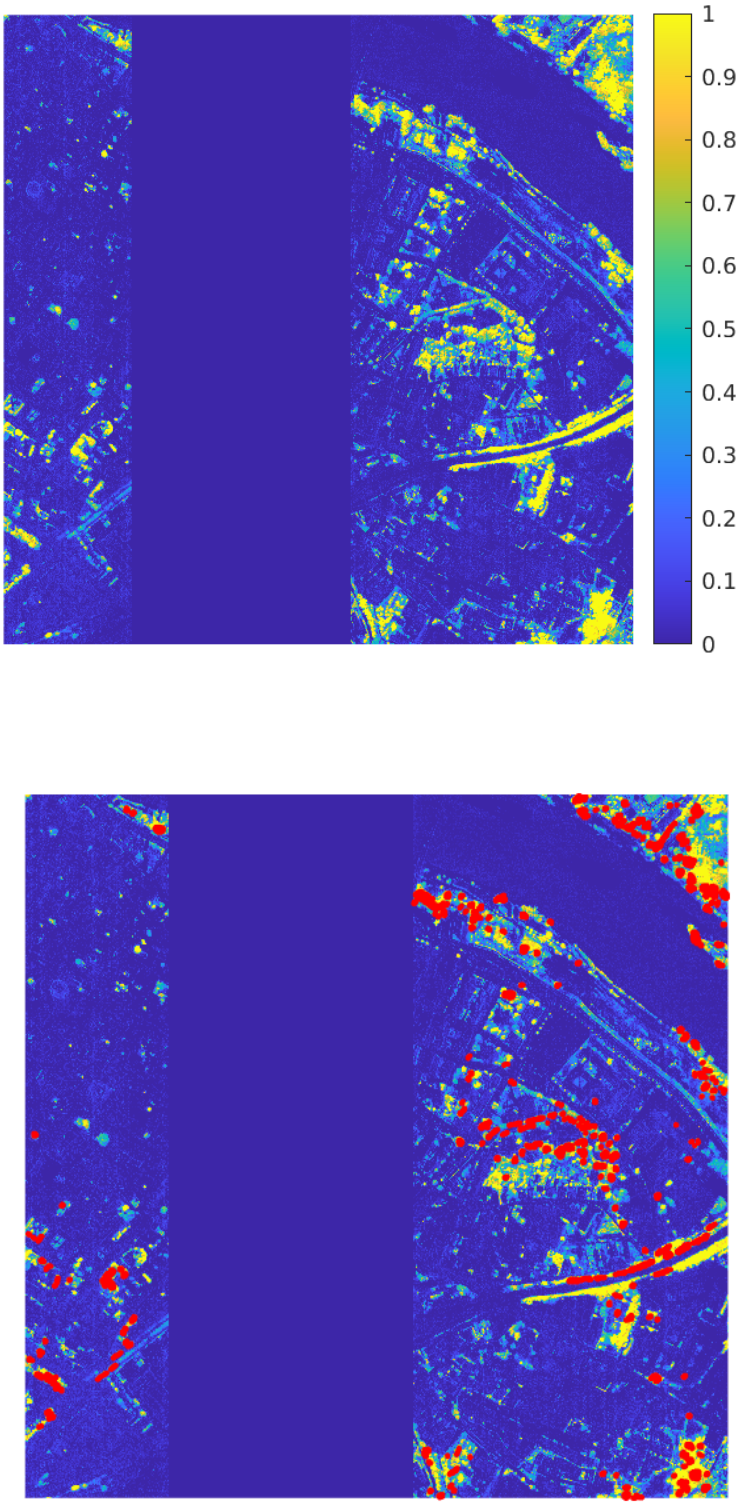

Figure A.2: Abundance of Trees, Pavia center

A.1.3 Asphalt

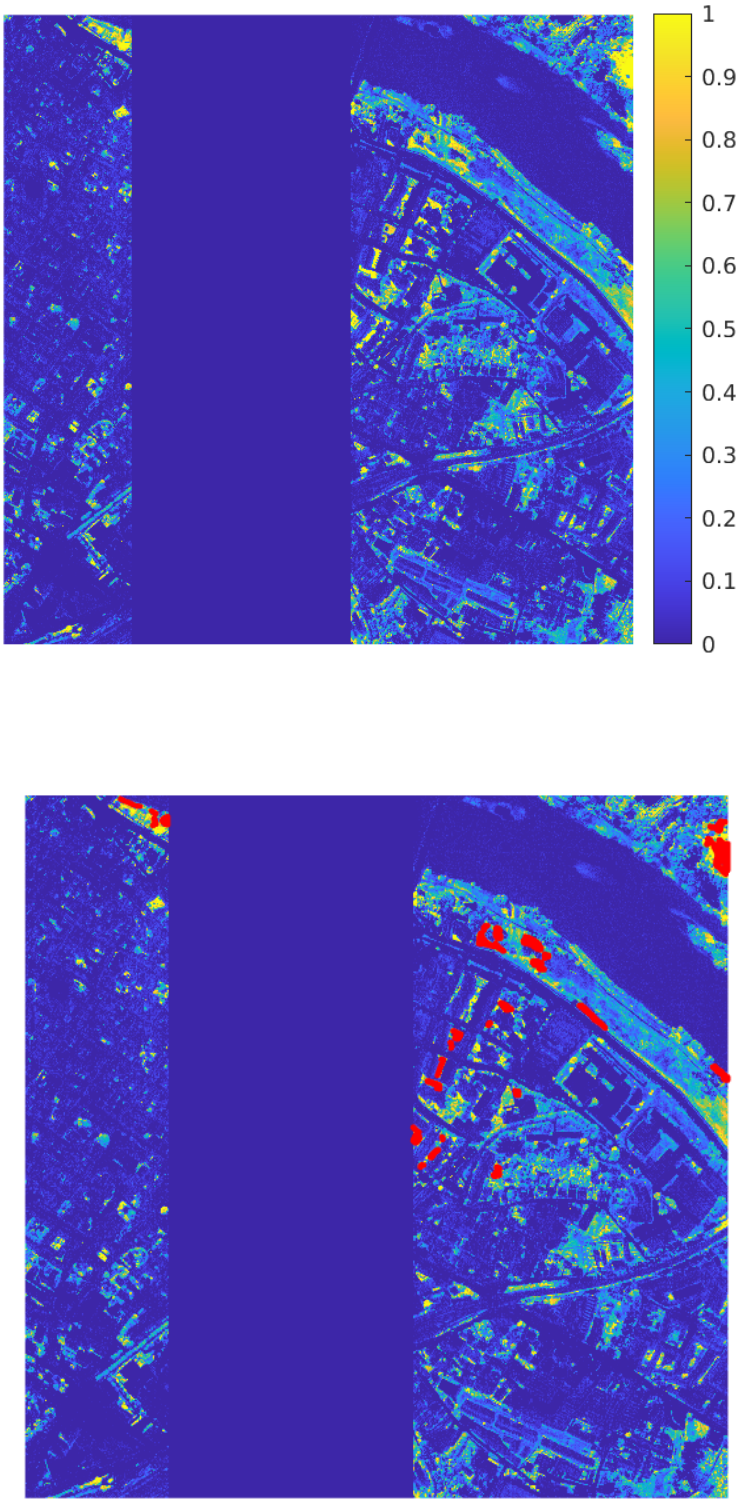

Figure A.3: Abundance of Asphalt, Pavia center

A.1.4 Self-blocking bricks

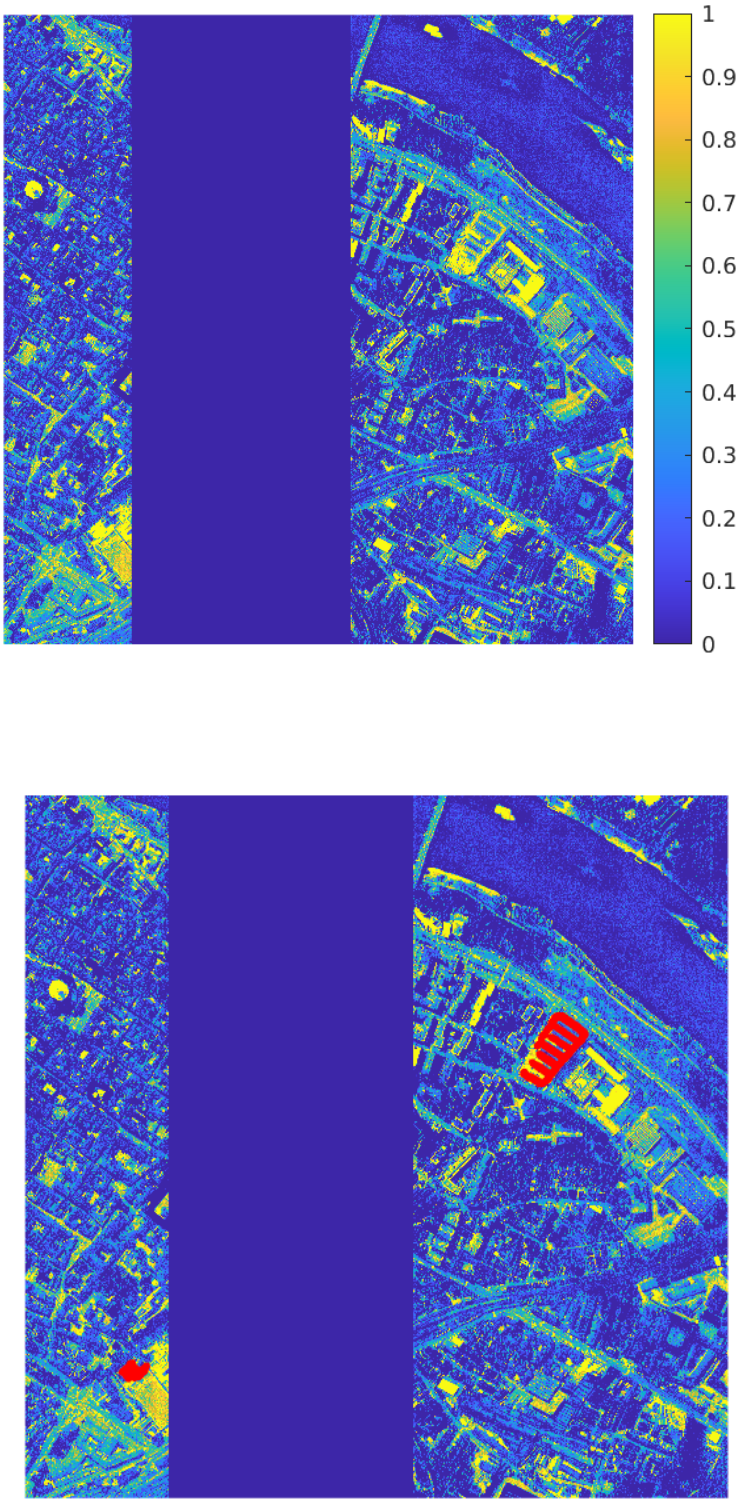

Figure A.4: Abundance of Self-blocking bricks, Pavia center

A.1.5 Bitumen

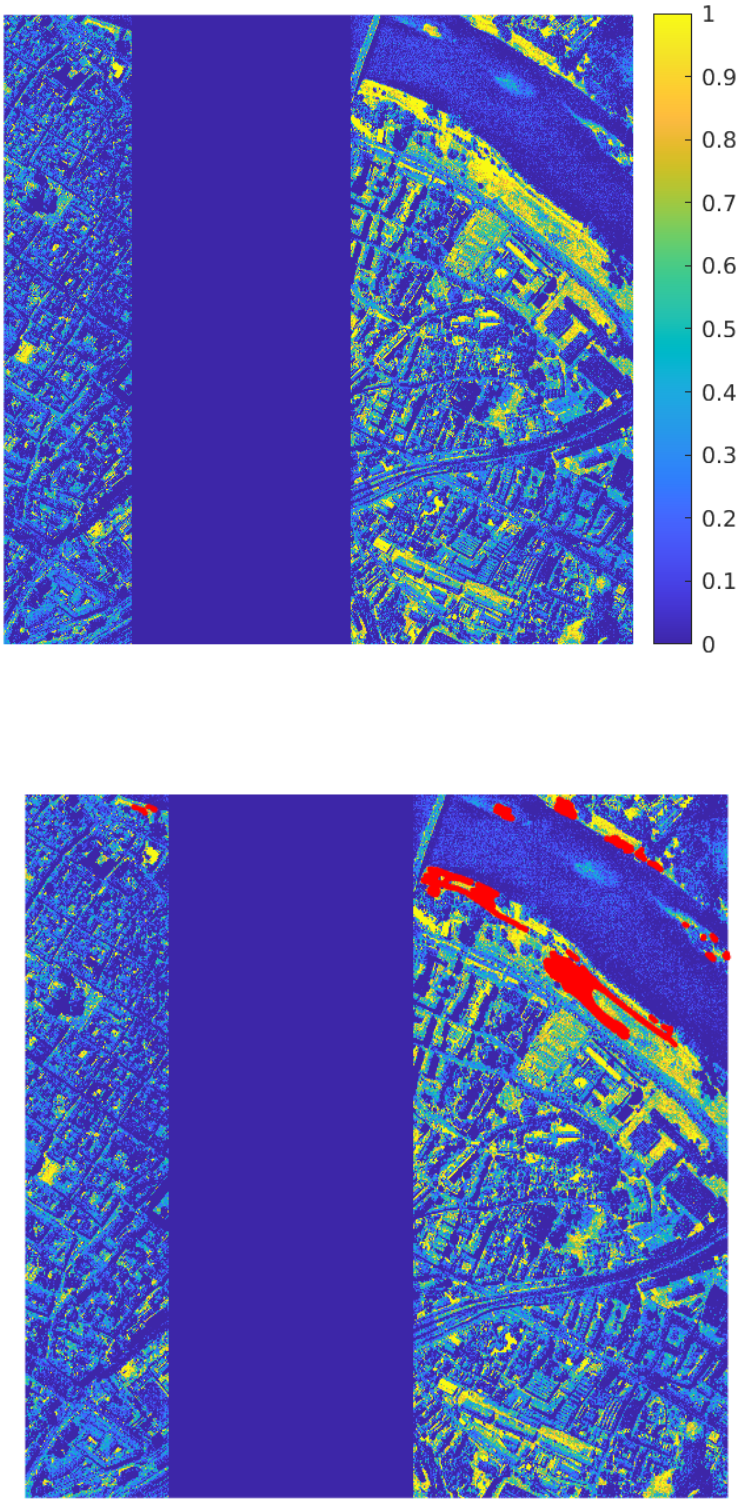

Figure A.5: Abundance of Bitumen, Pavia center

A.1.6 Tiles

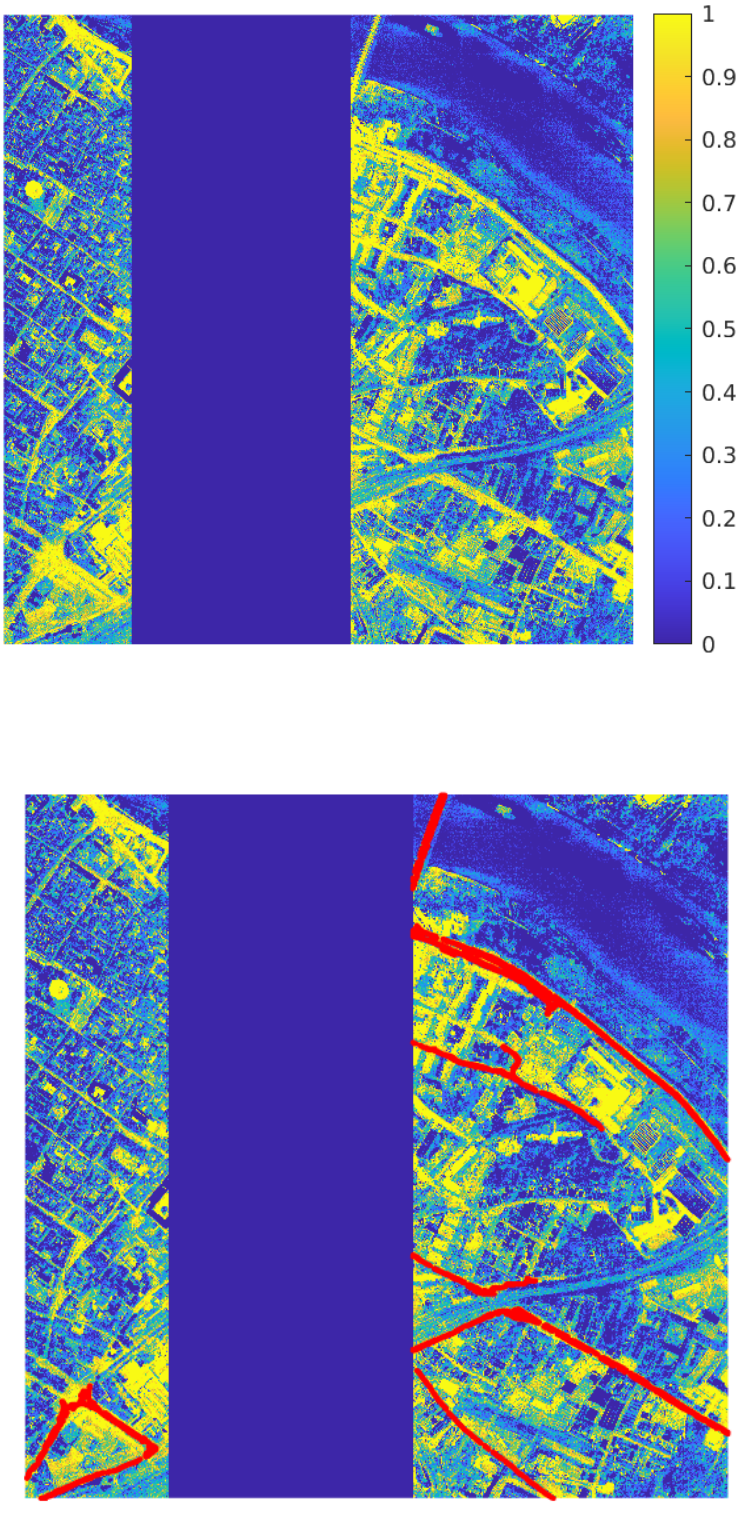

Figure A.6: Abundance of Tiles, Pavia center

A.1.7 Shadows

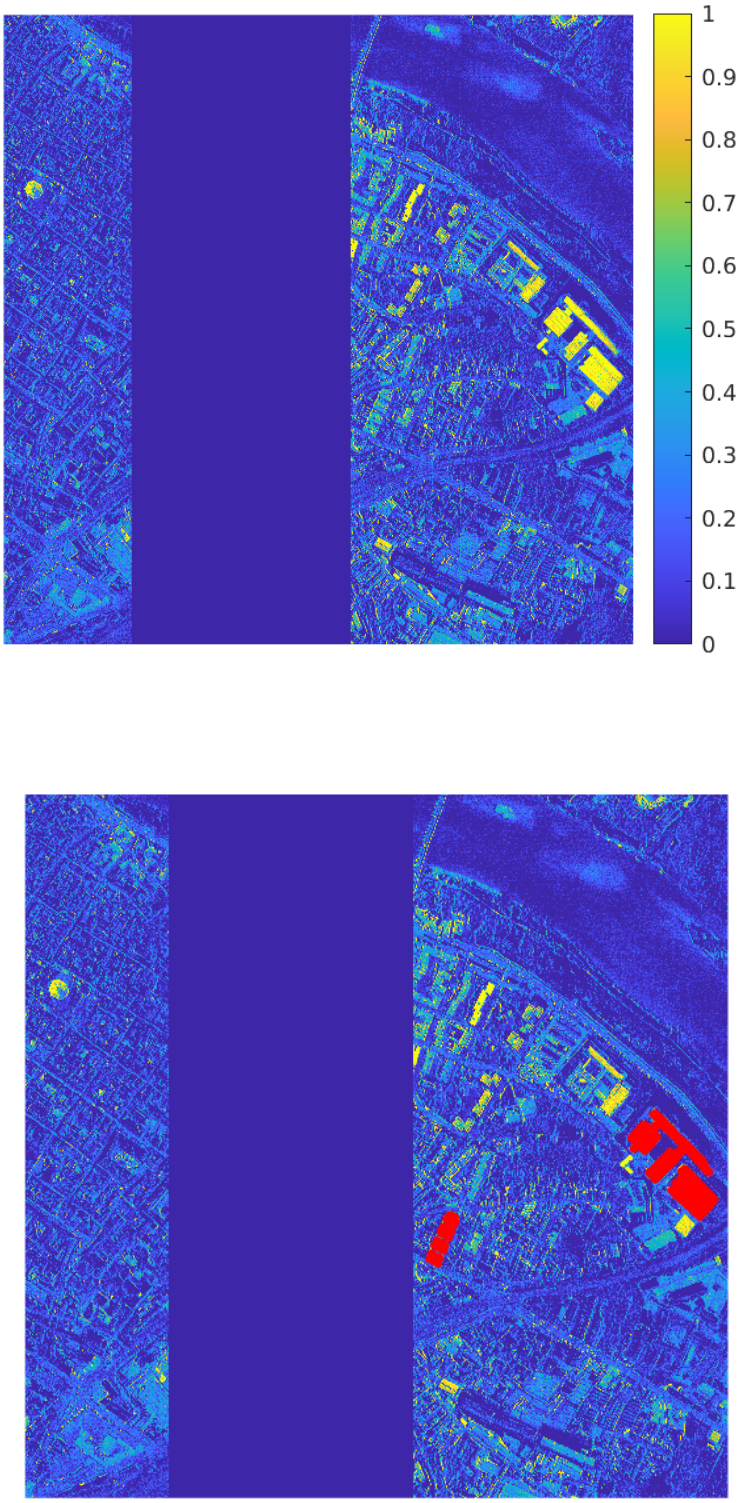

Figure A.7: Abundance of Shadows, Pavia center

A.1.8 Meadows

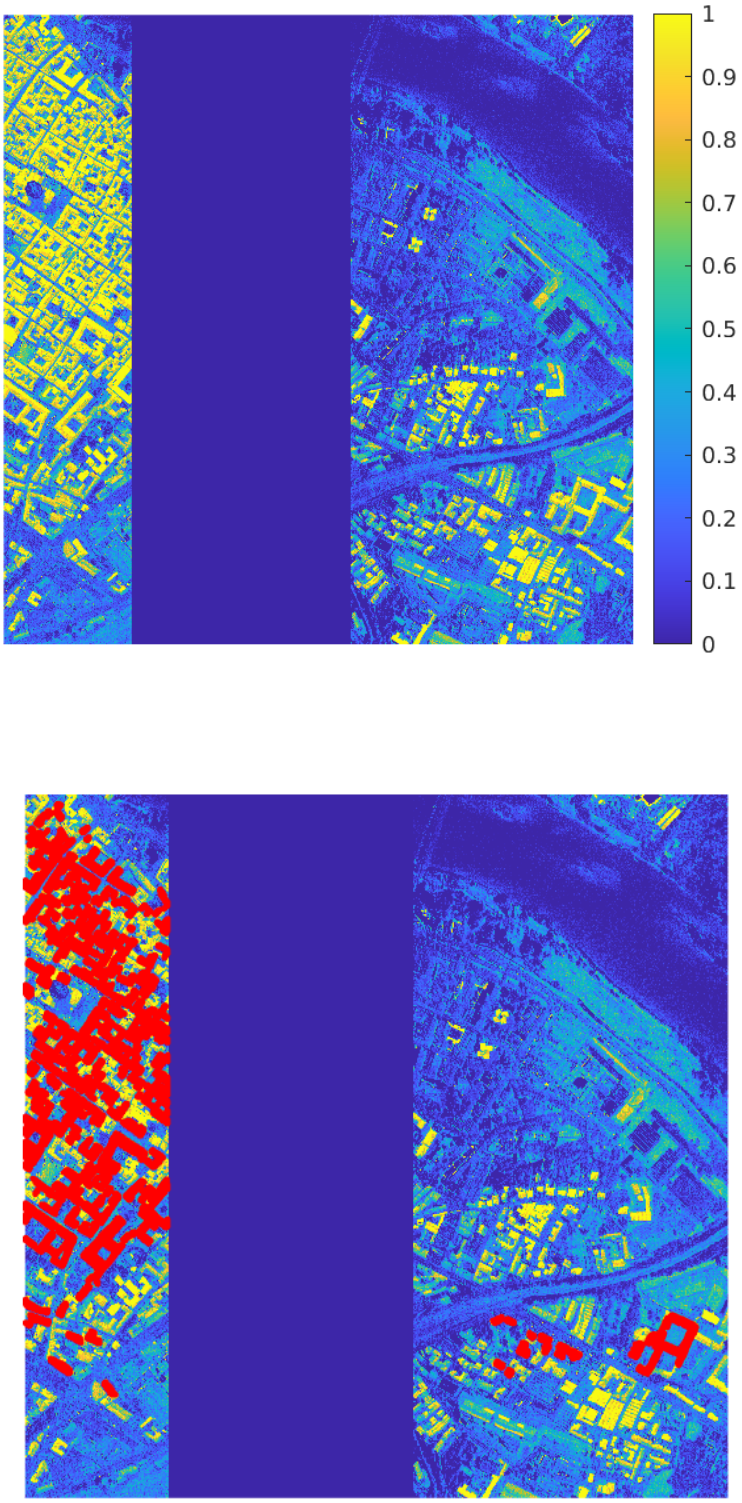

Figure A.8: Abundance of Meadows, Pavia center

A.1.9 Bare soil

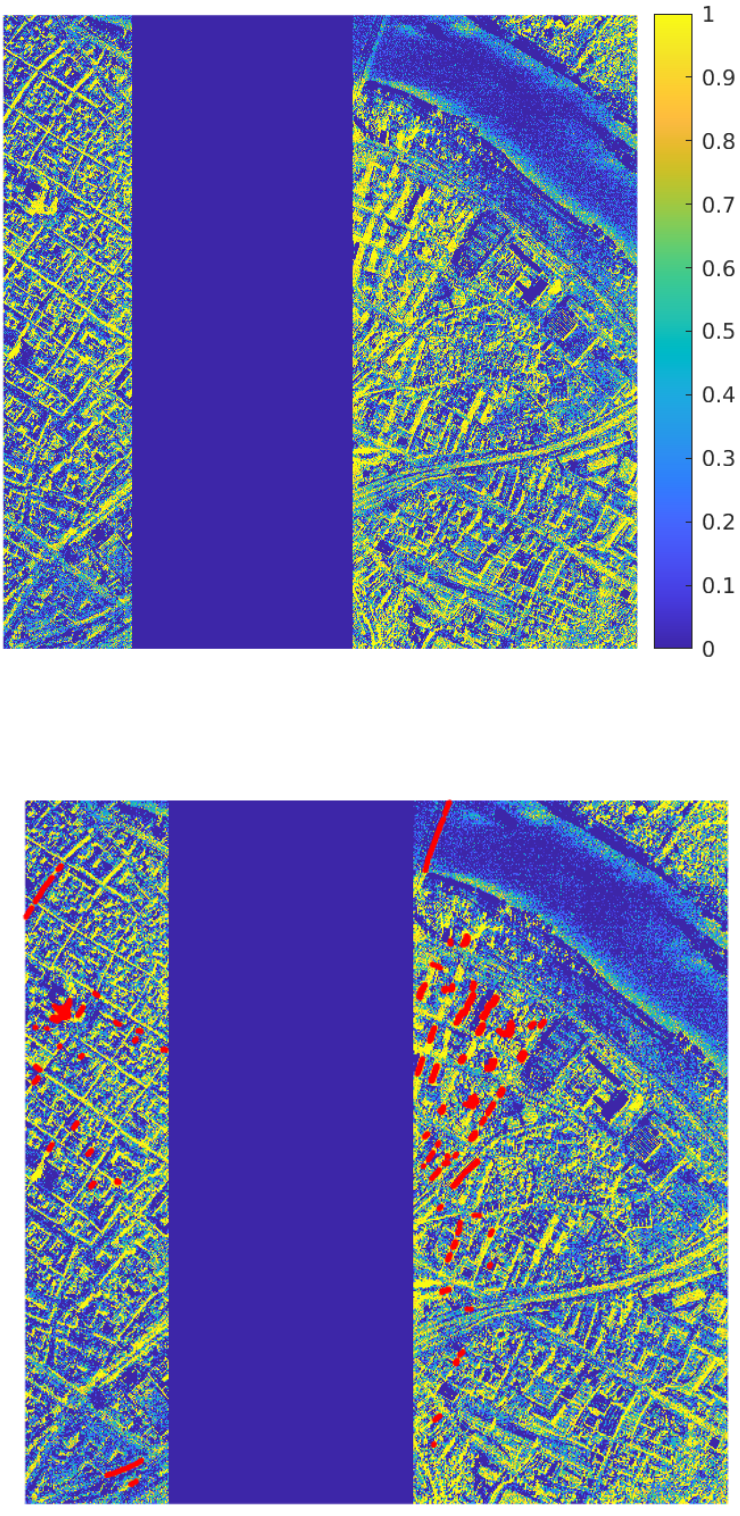

Figure A.9: Abundance of Bare soil, Pavia center

A.2 Pavia University

A.2.1 Asphalt

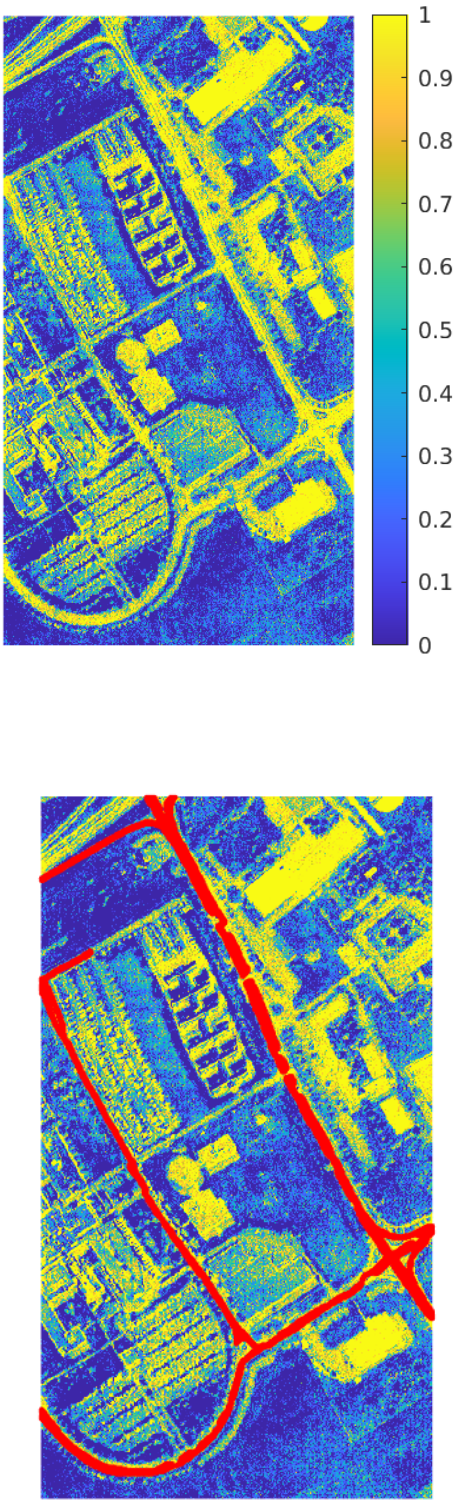

Figure A.10: Abundance of Asphalt, Pavia university

A.2.2 Meadows

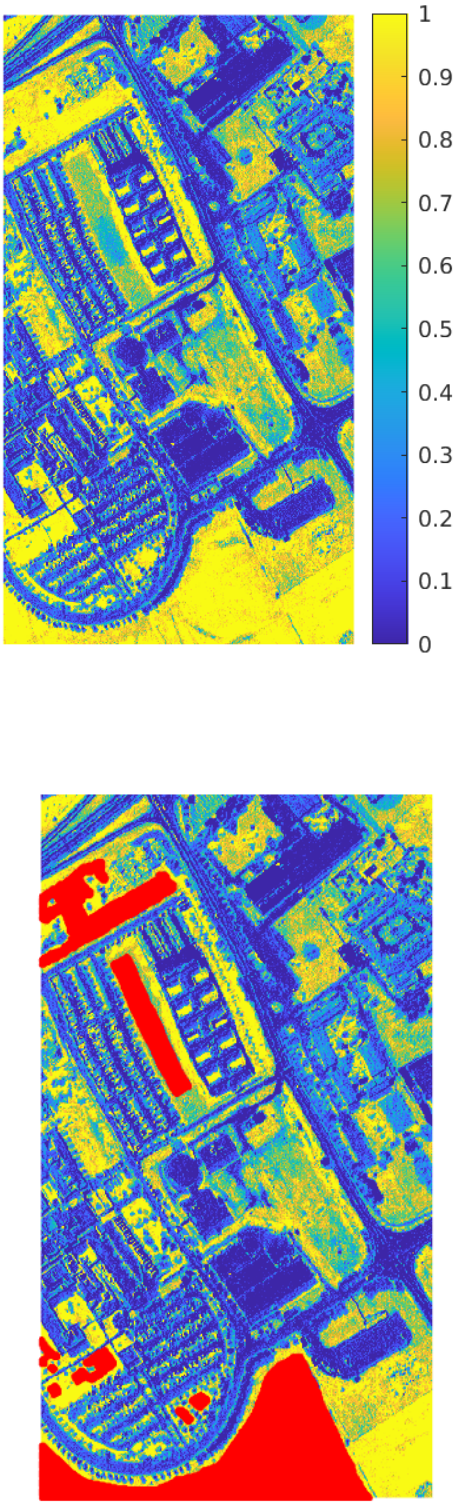

Figure A.11: Abundance of Meadows, Pavia university

A.2.3 Gravel

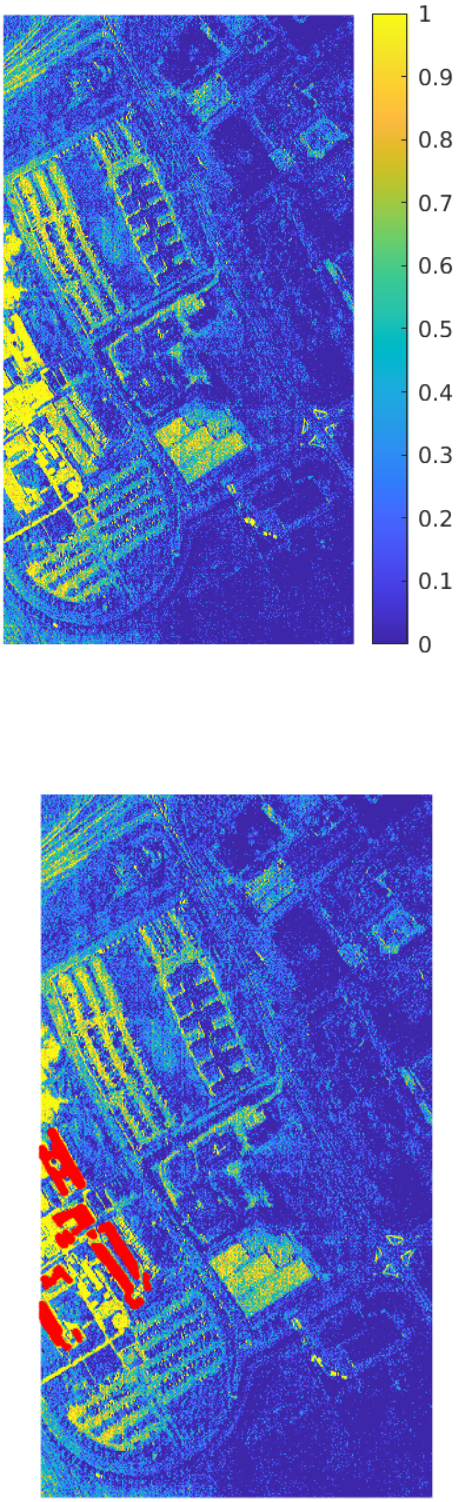

Figure A.12: Abundance of Gravel, Pavia university

A.2.4 Trees

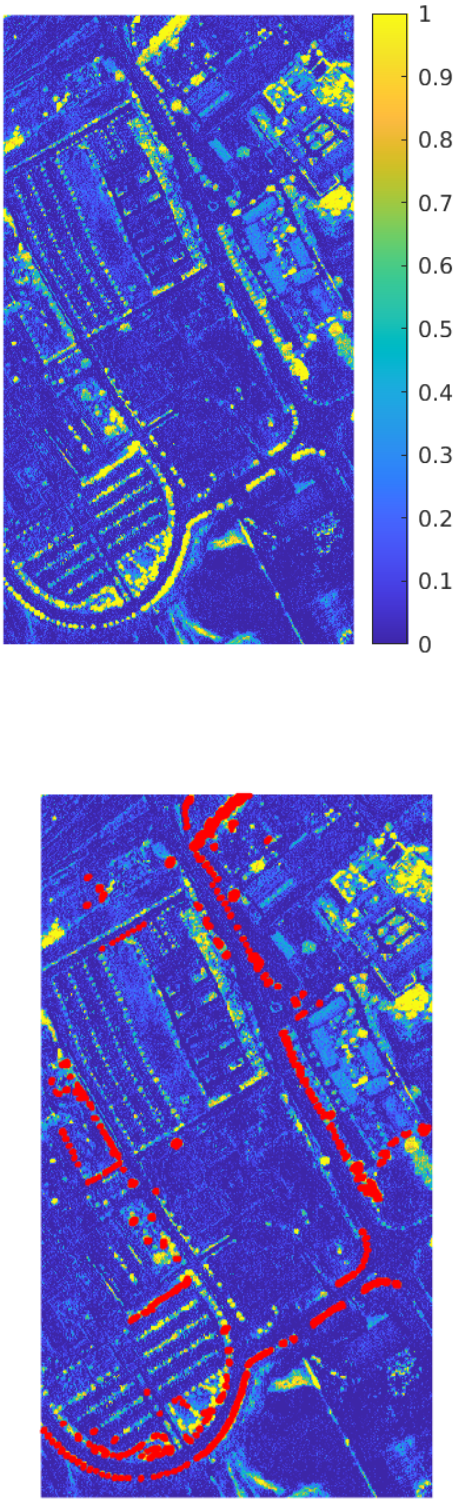

Figure A.13: Abundance of Trees, Pavia university

A.2.5 Painted metal sheets

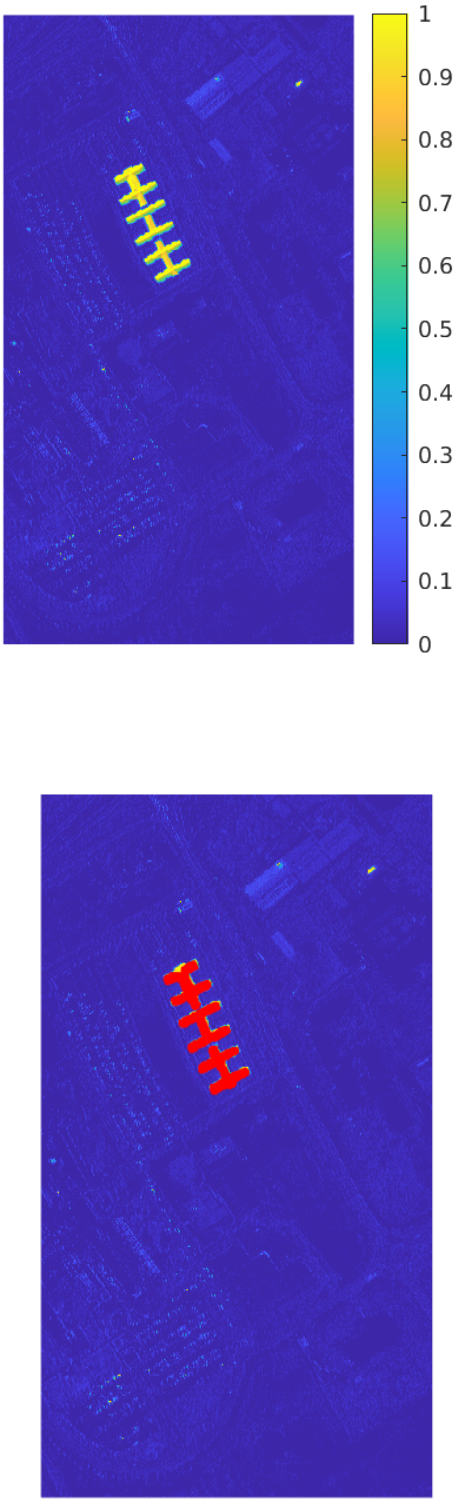

Figure A.14: Abundance of Painted metal sheets, Pavia university

A.2.6 Bare soil

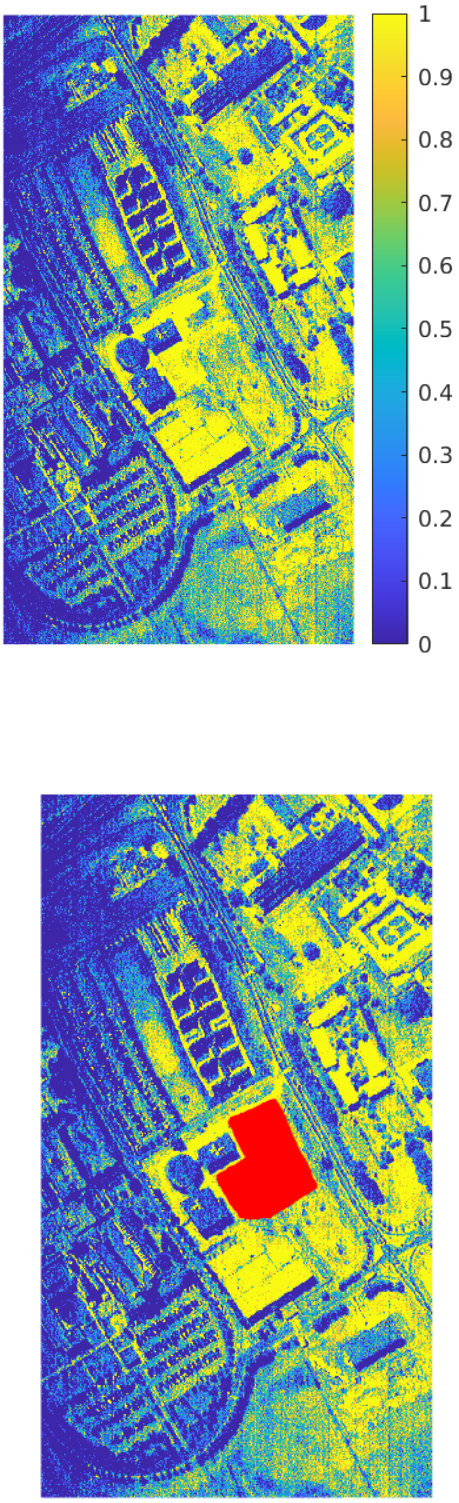

Figure A.15: Abundance of Bare soil, Pavia university

A.2.7 Bitumen

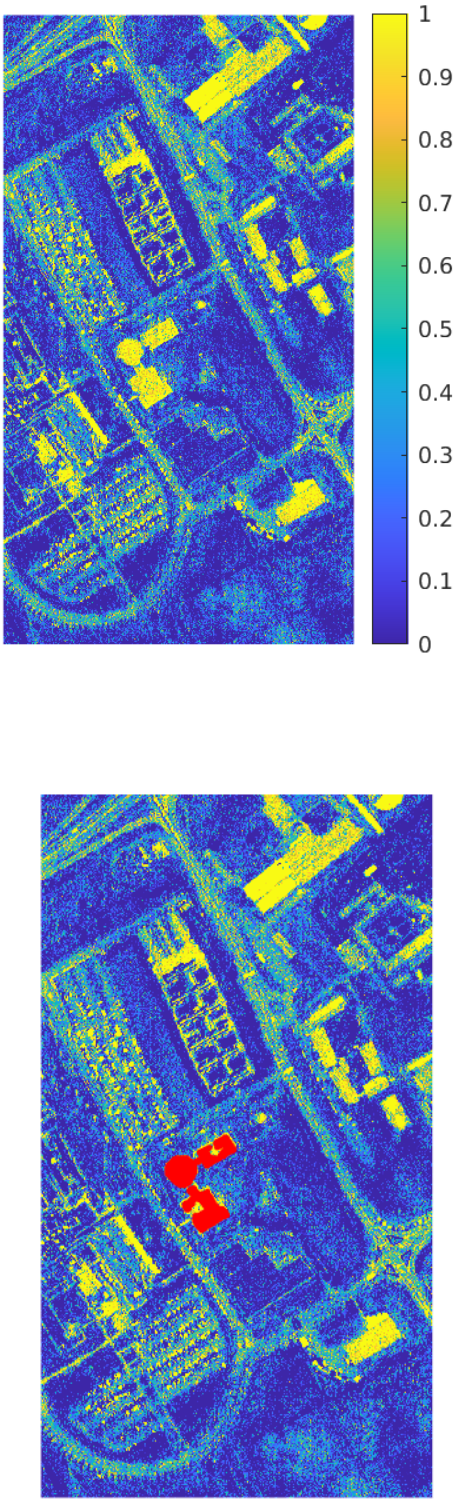

Figure A.16: Abundance of Bitumen, Pavia university

A.2.8 Self-blocking bricks

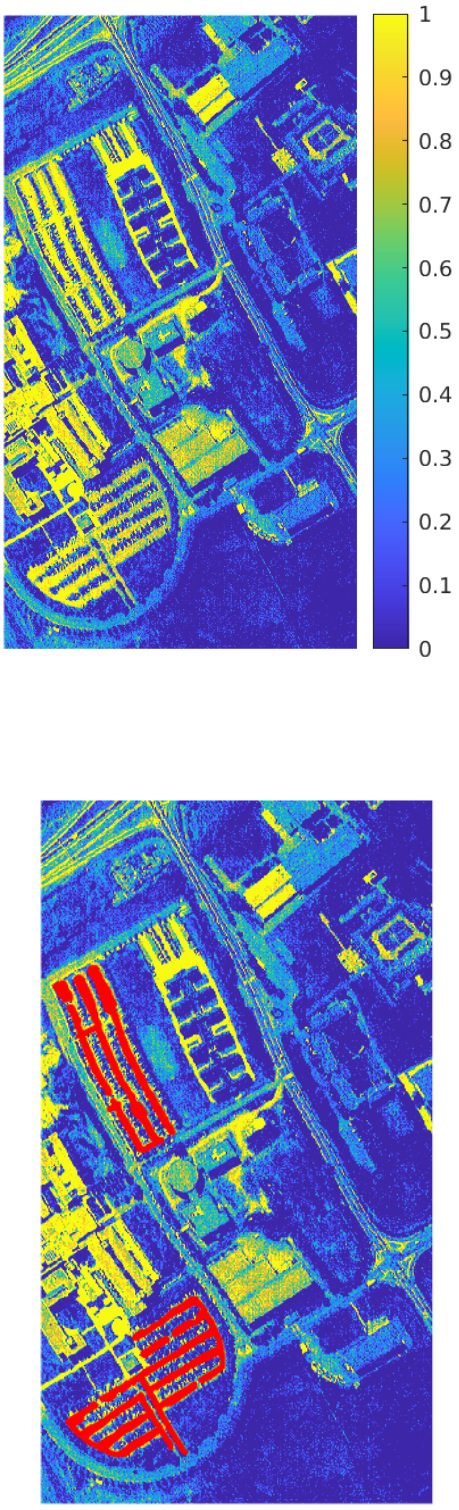

Figure A.17: Abundance of Self-blocking bricks, Pavia university

A.2.9 Shadows

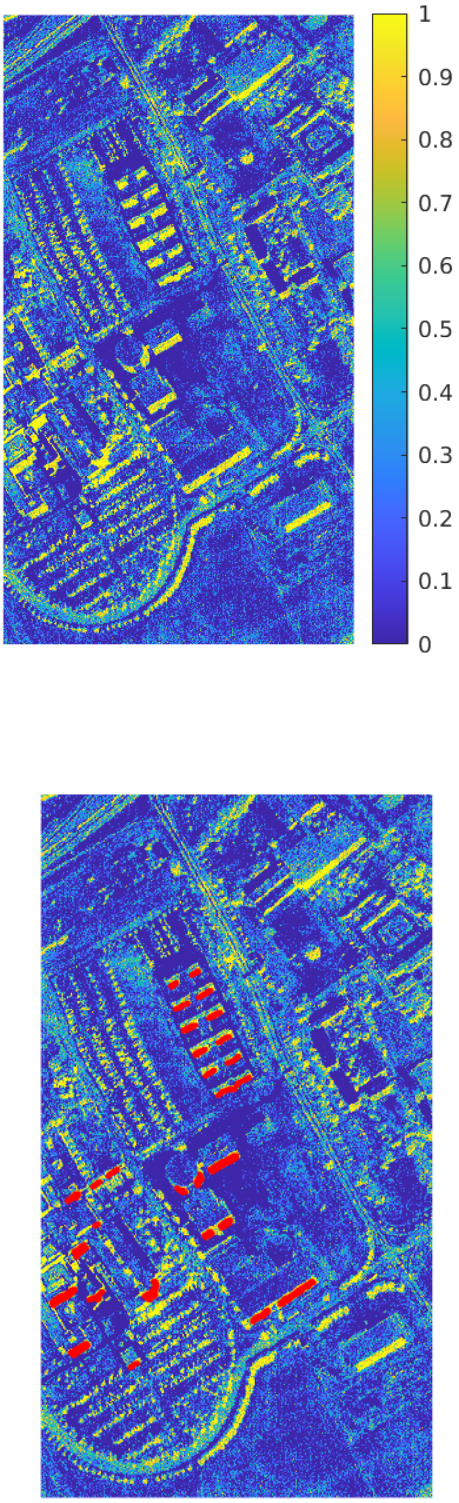

Figure A.18: Abundance of Shadows, Pavia university

B. Matlab Code

The code is also available on the GitHub repository of this paper[3].

B.1 Ground truth image generation

This Matlab code takes the raw ground truth images and generates the color images shown in chapter 2.

```
clc;clear;close all;


load PaviaU_gt.mat
load Pavia_gt.mat

gap = size(pavia_gt,1)-size(pavia_gt,2);
gap_pos = 223;



mycolors = [255,255,255; 0 0 128; 0,100,0; 134,154,172;214,41,41;80,80,80;...
            100,0,0;0,0,0;50,205,50;128,128,0]/255;
colormap(mycolors);
contourf(flipud([pavia_gt(:,1:gap_pos),zeros(size(pavia_gt,1),gap),...
    pavia_gt(:,gap_pos+1:end)]))
colorbar('Ticks',1:9,...
         'TickLabels',{'Water','Trees','Asphalt','Bricks',...
         'Bitumen','Tiles','Shadows','Meadows','Bare soil'})
axis equal
axis off
saveas(gcf,"images/Pavia/ground_truth.png")


swap = [3,4,5,8,7,6,1,2,9];
data = paviaU_gt;
for i = 1:9
    data(paviaU_gt==swap(i)) = i;
end

names = {'Asphalt','Meadows','Gravel','Trees',...
         'Painted metal sheets','Bare soil','Bitumen','Self-blocking bricks',...
         'Shadows'};


figure
mycolors = [255,255,255;0 0 128; 0,100,0; 134,154,172;214,41,41;80,80,80;...
            100,0,0;0,0,0;50,205,50;128,128,0]/255;
% mycolors = mycolors(swap,:);
colormap(mycolors);
contourf(flipud(data))
colorbar('Ticks',1:9,...
         'TickLabels',names(swap))
```

```
axis equal
axis off
saveas(gcf,"images/PaviaU/ground_truth.png")
```

B.2 Pavia Center

```
clc;clear;close all;

load Pavia_gt.mat;
load Pavia.mat;

data = pavia;
gt = pavia_gt;
clear paviaU paviaU_gt

gap = size(gt,1)-size(gt,2);
gap_pos = 223;

m = size(data,3);
n = max(gt,[],'all');

M = zeros(m,n);
freq = zeros(1,n);
R = zeros(m);
for i = 1:size(data,1)
    for j = 1:size(data,2)
        k = gt(i,j);
        if k>0
            M(:,k) = M(:,k) + reshape(data(i,j,:),[],1);
            freq(k) = freq(k) + 1;

            r = reshape(data(i,j,:),[],1);
            R = R + r*r';
        end
    end
end
M = M ./ freq;
R = R./(i*j);

%%
for k = 1:n
    alphas = zeros(size(gt));
    d = M(:,k);
    w = R\d/(d'/R*d);

    X = [];
    Y = [];

    error = 0;

    for i = 1:size(alphas,1)
        for j = 1:size(alphas,2)
            r = reshape(data(i,j,:),[],1);
            alphas(i,j) = max(0,min(1,w'*r));
            if gt(i,j) == k
                if (j>gap_pos)
                    X = [X,j+gap];
                else
                    X = [X,j];
                end
```

```
                Y = [Y,size(gt,1)-i];
                error = error + 1 - alphas(i,j);
            end
        end
    end
%     alphas = max(0,min(1,alphas));

    error = error / length(X);
    disp("error = " + string(error))

    figure
    alphas = [alphas(:,1:gap_pos),zeros(size(alphas,1),gap),alphas(:,gap_pos+1:end)];
    image(alphas,'CDataMapping','scaled')
    colorbar
    axis equal
    axis([0 size(alphas,2) 0 size(alphas,1)])
    axis off
    pause(1e-2)
    saveas(gcf,'images/Pavia/k='+string(k)+'_unknown_U.png')

    figure
    hold on
    image(flipud(alphas),'CDataMapping','scaled')
    plot(X,Y,'r.')
    axis equal
    axis([0 size(alphas,2) 0 size(alphas,1)])
    axis off
    pause(1e-2)
    saveas(gcf,'images/Pavia/k='+string(k)+'_unknown_U(red).png')

    alphas = zeros(size(gt,1),size(gt,2),n);

    Minv = (M'*M)\M';
    threshold = .3;

    for i = 1:size(alphas,1)
        for j = 1:size(alphas,2)
            r = reshape(data(i,j,:),[],1);
            A = Minv*r;

            A = max(0,min(1,A));
%             A = A/sum(A);
            idx = A > threshold;
%             [¬ ,idx] = max(A);

%             A = zeros(size(A));
%             A(idx) = 1;

            alphas(i,j,:) = A;
        end
    end

    figure
    alphas = alphas(:,:,k);
    alphas = [alphas(:,1:gap_pos),zeros(size(alphas,1),gap),alphas(:,gap_pos+1:end)];
    image(alphas,'CDataMapping','scaled')
    colorbar
    axis equal
    axis([0 size(alphas,2) 0 size(alphas,1)])
    axis off
    pause(1e-2)
    saveas(gcf,'images/Pavia/k='+string(k)+'_pseudo_inverse.png')

```

```
    figure
    hold on
    image(flipud(alphas),'CDataMapping','scaled')
    plot(X,Y,'r.')
    axis equal
    axis([0 size(alphas,2) 0 size(alphas,1)])
    axis off
    pause(1e-2)
    saveas(gcf,'images/Pavia/k='+string(k)+'_pseudo_inverse(red).png')


    alphas = zeros(size(gt));
    d = M(:,k);
    U = M;
    U(:,k) = [];

    Pu = ones(m) - U/(U'*U)*U';

    for i = 1:size(alphas,1)
        for j = 1:size(alphas,2)
            r = reshape(data(i,j,:),[],1);
            A = d'*Pu*r/(d'*Pu*d);
            alphas(i,j) = max(0,min(1,A));
        end
    end

    figure
    alphas = [alphas(:,1:gap_pos),zeros(size(alphas,1),gap),alphas(:,gap_pos+1:end)];
    image(alphas,'CDataMapping','scaled')
    colorbar
    axis equal
    axis([0 size(alphas,2) 0 size(alphas,1)])
    axis off
    pause(1e-2)
    saveas(gcf,'images/Pavia/k='+string(k)+'_optimum_detection.png')

    figure
    hold on
    image(flipud(alphas),'CDataMapping','scaled')
    plot(X,Y,'r.')
    axis equal
    axis([0 size(alphas,2) 0 size(alphas,1)])
    axis off
    pause(1e-2)
    saveas(gcf,'images/Pavia/k='+string(k)+'_optimum_detection(red).png')

end

pause(1)
close all
```

B.3 Pavia University

```
clc;clear;close all;

load PaviaU_gt.mat
load PaviaU.mat

data = paviaU;
gt = paviaU_gt;
```

```
clear paviaU paviaU_gt


m = size(data,3);
n = max(gt,[],'all');

M = zeros(m,n);
freq = zeros(1,n);
R = zeros(m);
for i = 1:size(data,1)
    for j = 1:size(data,2)
        k = gt(i,j);
        if k>0
            M(:,k) = M(:,k) + reshape(data(i,j,:),[],1);
            freq(k) = freq(k) + 1;

            r = reshape(data(i,j,:),[],1);
            R = R + r*r';
        end
    end
end
M = M ./ freq;
R = R./(i*j);

% w = zeros(m,n);
% for k = 1:n
%     d = M(:,k);
%     w(:,k) = R\d/(d'/R*d);
% end

%%
for k = 1:n
    alphas = zeros(size(gt));
    d = M(:,k);
    w = R\d/(d'/R*d);

    X = [];
    Y = [];

    error = 0;
    for i = 1:size(alphas,1)
        for j = 1:size(alphas,2)
            r = reshape(data(i,j,:),[],1);
            alphas(i,j) = min(1,max(0,w'*r));
            if gt(i,j) == k
                X = [X,j];
                Y = [Y,size(gt,1)-i];
                error = error + 1 - alphas(i,j);
            end
        end
    end
%     alphas = max(0,min(1,alphas));

    error = error / length(X);
    disp("error = " + string(error))

   figure
    image(alphas,'CDataMapping','scaled')
    colorbar
    axis equal
    axis([0 size(alphas,2) 0 size(alphas,1)])
    axis off
    pause(1e-2)
```

```
71      saveas(gcf,'images/PaviaU/k='+string(k)+'_unknown_U.png')
72
73      figure
74      hold on
75      image(flipud(alphas),'CDataMapping','scaled')
76      plot(X,Y,'r.')
77      axis equal
78      axis([0 size(alphas,2) 0 size(alphas,1)])
79      axis off
80      pause(1e-2)
81      saveas(gcf,'images/PaviaU/k='+string(k)+'_unknown_U(red).png')
82
83      alphas = zeros(size(gt,1),size(gt,2),n);
84
85      Minv = (M'*M)\M';
86      threshold = .3;
87
88      for i = 1:size(alphas,1)
89          for j = 1:size(alphas,2)
90              r = reshape(data(i,j,:),[],1);
91              A = Minv*r;
92
93              A = max(0,min(1,A));
94      %         A = A/sum(A);
95              idx = A > threshold;
96      %         [¬ ,idx] = max(A);
97
98      %         A = zeros(size(A));
99      %         A(idx) = 1;
100
101             alphas(i,j,:) = A;
102         end
103     end
104
105     figure
106     alphas = alphas(:,:,k);
107     image(alphas,'CDataMapping','scaled')
108     colorbar
109     axis equal
110     axis([0 size(alphas,2) 0 size(alphas,1)])
111     axis off
112     pause(1e-2)
113     saveas(gcf,'images/PaviaU/k='+string(k)+'_pseudo_inverse.png')
114
115     figure
116     hold on
117     image(flipud(alphas),'CDataMapping','scaled')
118     plot(X,Y,'r.')
119     axis equal
120     axis([0 size(alphas,2) 0 size(alphas,1)])
121     axis off
122     pause(1e-2)
123     saveas(gcf,'images/PaviaU/k='+string(k)+'_pseudo_inverse(red).png')
124
125     alphas = zeros(size(gt));
126     d = M(:,k);
127     U = M;
128     U(:,k) = [];
129
130     Pu = ones(m) - U/(U'*U)*U';
131
132     for i = 1:size(alphas,1)
133         for j = 1:size(alphas,2)
```

```
134             r = reshape(data(i,j,:),[],1);
135             A = d'*Pu*r/(d'*Pu*d);
136             alphas(i,j) = max(0,min(1,A));
137         end
138     end
139 
140     figure
141     image(alphas,'CDataMapping','scaled')
142     colorbar
143     axis equal
144     axis([0 size(alphas,2) 0 size(alphas,1)])
145     axis off
146     pause(1e-2)
147     saveas(gcf,'images/PaviaU/k='+string(k)+'_optimum_detection.png')
148 
149     figure
150     hold on
151     image(flipud(alphas),'CDataMapping','scaled')
152     plot(X,Y,'r.')
153     axis equal
154     axis([0 size(alphas,2) 0 size(alphas,1)])
155     axis off
156     pause(1e-2)
157     saveas(gcf,'images/PaviaU/k='+string(k)+'_optimum_detection(red).png')
158 
159 end
160 
161 pause(2)
162 close all
```

Bibliography

[1] CRS. Rosis | center for remote sensing (crs). https://crs.hi.is/?page_id=877.

[2] Grupo de Inteligencia Computacional (GIC). Hyperspectral remote sensing scenes.

[3] Maximilien Remacle. Github/hyperspcetral-unmixing. https://github.com/Sp0utn1k/Hyperspectral-unmixing.

[4] Wikipedia contributors. Light — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Light&oldid=1052992605, 2021.