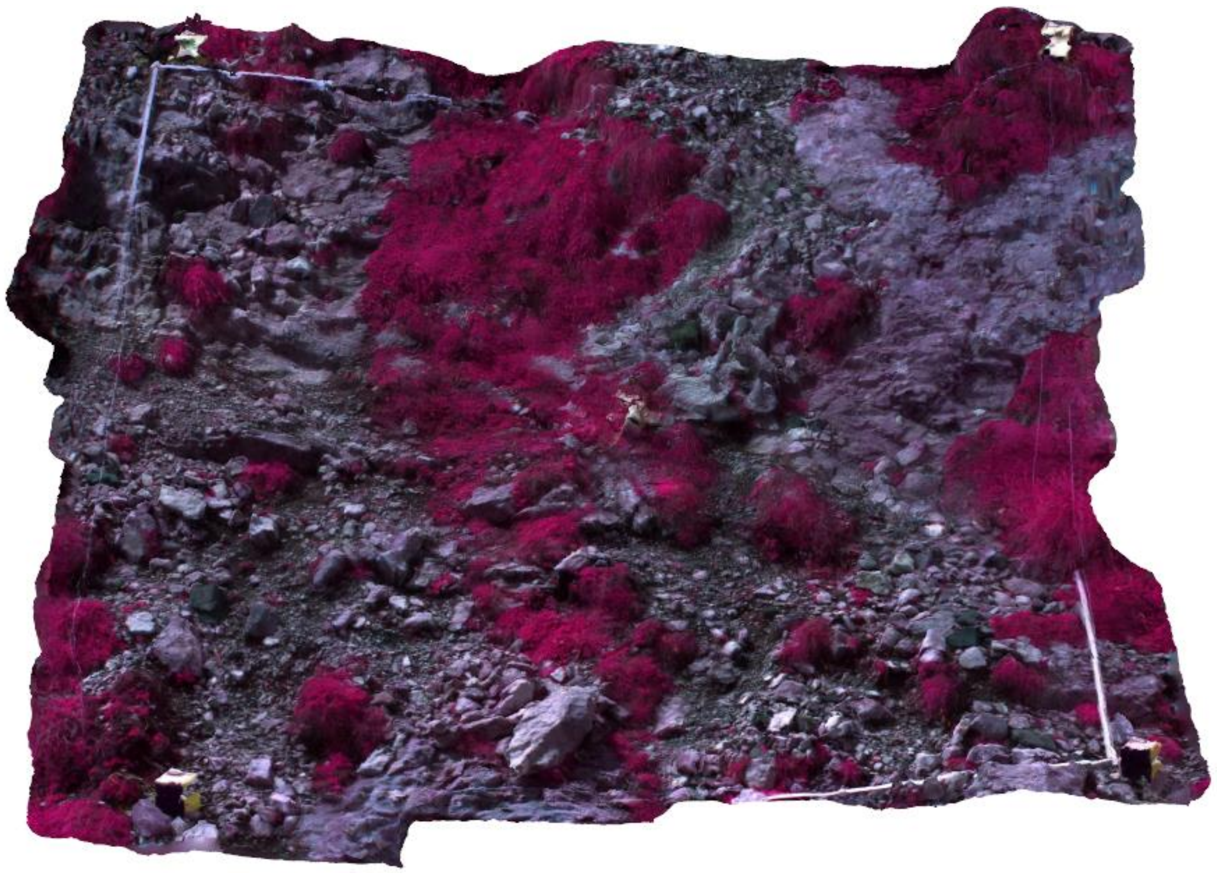
Vegetation detection in Structure-from-Motion derived 3D-models

J. (Jurjen) C. Kamphuis
4030230



BSc Student at Delft University of Technology,
Delft, The Netherlands
J.C.Kamphuis@student.tudelft.nl
Supervisors: Dr. R.C. Lindenbergh (TU Delft)
Mag. Dr. Martin Rutzinger (IGF)

This report was written by Jurjen Kamphuis as a part of the Bachelor of Science degree in Applied earth sciences at the Delft University of Technology.

 The process took place at the Institute of Mountain Research in Innsbruck, Australia under the direct supervision of Dr. R.C. Lindenbergh and Mag. Dr. Martin Rutzinger.

I would like to thank MSc. Robert Niederheiser for incorporating me in his PhD fieldwork.

Figure 1 (on the cover): False colour 3D model of plot PNL_E45.

Abstract

A multispectral camera has been used to make NIR, RED, GRN images of GLORIA plots. These are modelled with Agisoft Photoscan (Agisoft, 2015) following a Structure from Motion approach to get orthophotos and texture atlases. Pixels of these images are classified on their NDVI response, classifying them as vegetation or rock. The resulting 3D models seem to have local errors in the order of centimetres. Global errors cannot be assessed because of a lack of reverence.

Contents

1. Introduction

This thesis will guide the reader through the process of making three dimensional (3D) models of mountain slopes for the use of micro topography analysis and vegetation classification. Square patches of 3 by 3 meter called 'plots' (see figure 2) will be captured using a multispectral camera, capturing Near infra-red (NIR), green (GRN) and red (RED) light (example picture is given in figure 27, chapter 2.4)

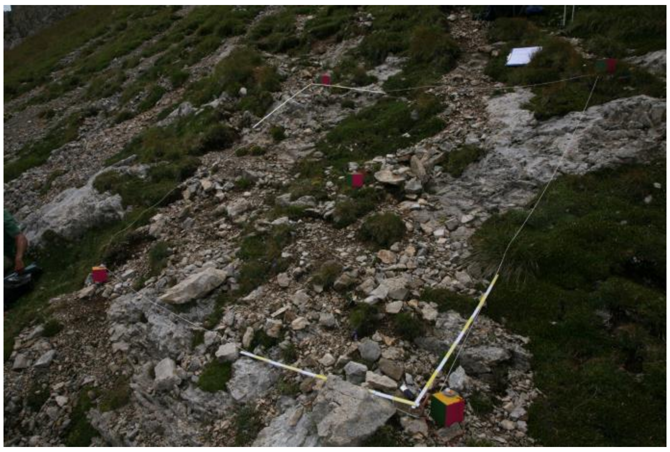


Figure 2. One of the 3-by-3 meter marked quadrants.

A total of 48 plots are captured as a complementary dataset for possible further use in the PhD thesis of MSc Robert Niederheiser at the institute of mountain research in Innsbruck, Austria. The further use and the social relevance of these models will be discussed in the chapter '2.1 MEDIALPS'

The plots are located on 3 alpine summits on two locations shown in figure 3. The summits do not have an official name so it is best to name them according to their code in this report. 'BRU' is located in the Valais area in Switzerland (figure 4) and 'RNK' and 'PNL' are located in the Latemar mountain group in the Dolomites, Italy (figure 5).

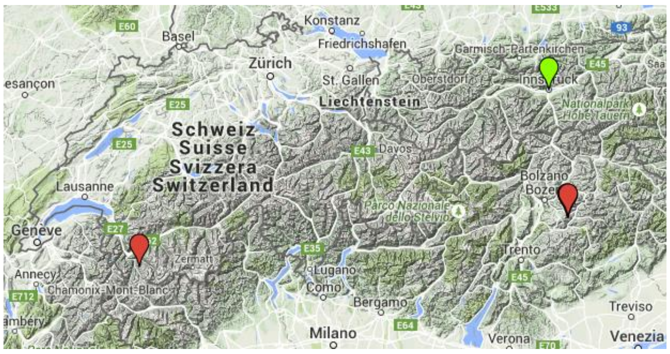


Figure 3. Two sample locations (red) and Innsbruck (green)

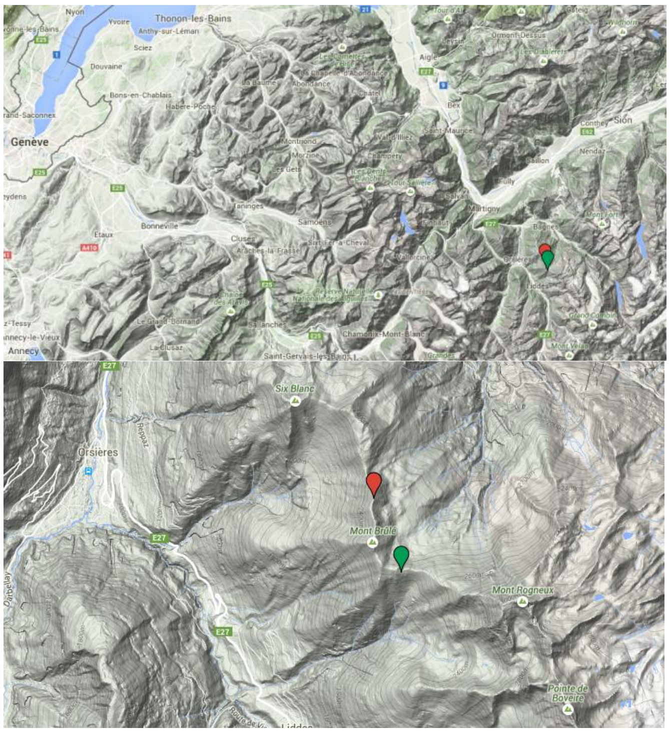


Figure 4. Swiss location with summit BRU (red) and our hut (green)

Summit BRU has a height of about 2500m and RNK and PNL 2700m and 2410 respectively. All these summits are well beyond the tree line but within reach for low vegetation.

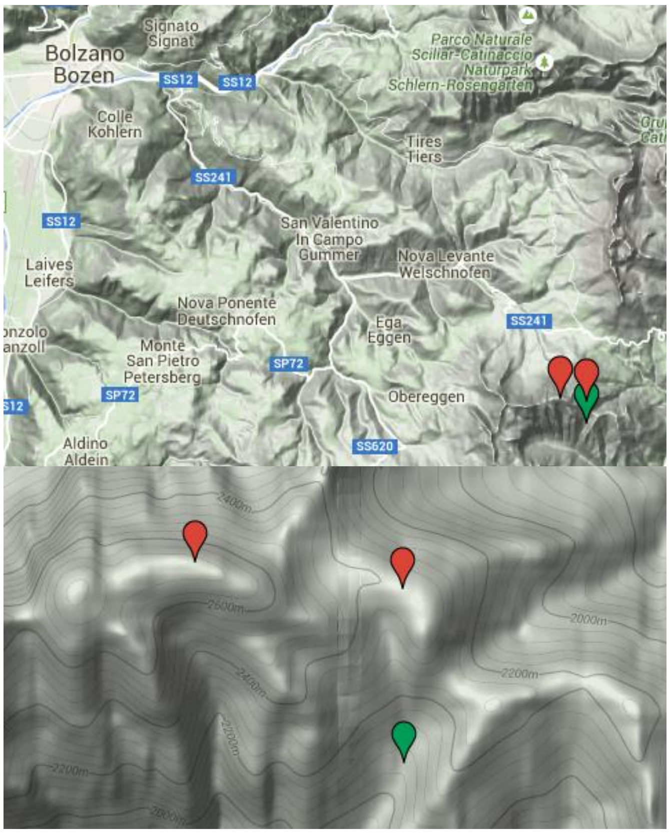
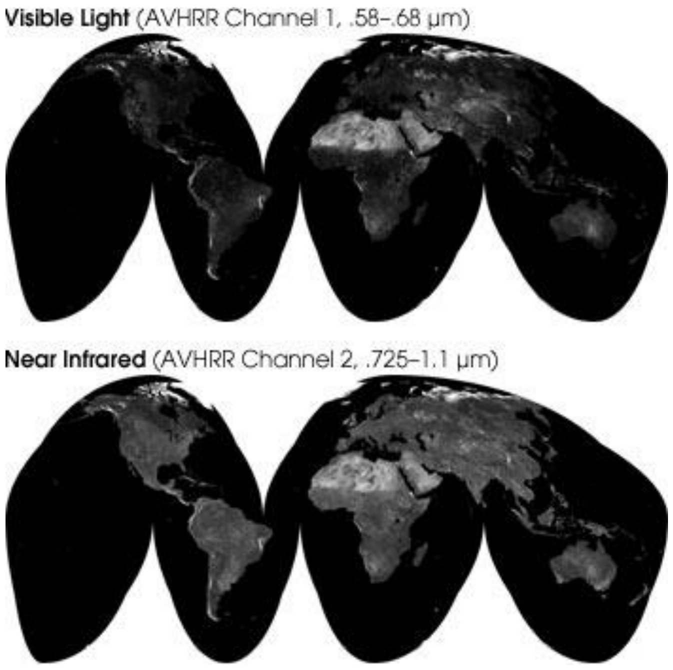


Figure 5. Italian location with summit RNK (left red) PNL (right red) and tent (green)

Within the MEDIALPS project, plots will be modelled with a 'Structure-from-Motion' approach which is explained in chapter 2.2. In this approach a software package can extract 3D models from a group of overlapping pictures. These pictures can be made by relatively lightweight consumer grade cameras making the data acquisition simple (Westoby et al, 2012). Due to the location of the plots, this was an important factor in choosing this approach.



Visible Light (AVHRR Channel 1, .58–.68 μm)
Near Infrared (AVHRR Channel 2, .725–1.1 μm)

Figure 6. Reflectance of the earth surface in two different spectra (taken from: Earth observatory, NASA, 2015)

Pictures are taken with a multispectral camera which records NIR, RED and GRN instead of the normal red, green and blue (RGB) colours. This is done because of the following effect. Vegetation has the property of absorbing high amounts of visible light for photosynthesis while the cell structures reflecting lots of NIR light. Others substances on earth surface do not have this property making it possible to use this data to indicate vegetated areas from bare lands with the use of (aerial) pictures made with these multispectral cameras. Imagery taken from space illustrates this effect very well. In figure 6 you see two images of the world. High densely vegetated areas such as equatorial rainforests and boreal forests in the northern hemisphere show very dark in the visible light spectrum, meaning they do not reflect much of this light. However, these same areas show bright colours in the NIR spectre indicating a high reflectivity. This has a big contrast with the bare lands of the African Sahara and the Arabian Peninsula, where both visible and NIR light is reflected to a high degree. *(Earth Observatory, NASA, 2015)*

1.1 Problem description

The aim for this report is to explore the potential of the multispectral camera for the use of detecting vegetation cover on the rock face. As the rock face is given by a 3D structure, I aim to detect the vegetation in a 3D model. This will give the opportunity to do the microanalysis of the morphology at the same time, but mostly it may give a more realistic view on the ratio vegetation/bare rock than a 2D picture would.

The main question of this report is:

- How can the multispectral camera be used to detect vegetation on a rocky slope?

In order to answer this question, several sub questions will be addressed:

- How to classify vegetation from NIR-, GRN- and RED light-intensity?
- How should you gather data on the site for easy processing and accurate models?
- Are the results of the camera suitable for 3D Structure-from-Motion modelling?
- Is the resolution of the 3D model high enough to make a micro topography analysis?

The first two sub questions will be answered in the methodology chapter, the other two questions will be answered while discussing the results.

2. Background

To get a better understanding of the methods used in this report, there are some chapters on the background of important aspects of the report. When a subject is known by the reader, it may be skipped trough instead of thoroughly read.

2.1 Background on MEDIALPS

This bachelor thesis has been made as a side track of the MEDIALPS project. In this project, researchers try to disentangle anthropogenic drivers of global change impacts on alpine plant species composition. It is a comparison study between the Alps and Mediterranean mountains (MEDIALPS, 2015). In less scientific words this means they will monitor alpine plant species composition together with impacts effected by climate such as rainfall, solar radiation and temperature. Besides, social scientists map the change in anthropogenic activities such as changes in grazing, tourism and land use.

To gain a better understanding in why I performed certain tests, I will explain the background and general workflow of the MEDIALS project in this chapter.

Due to the industrialization of the modern world, ecosystems have been put under pressure by drivers such as climate change, nitrogen deposition (the effect of large amounts of anthropogenic nitrogen (N) in the N-cycle) and land-use changes. Remote mountain areas are not excluded from these global processes. On the large scale, species tend to move towards areas with suitable living conditions. As warming takes place on a global scale, the climate regimes shift towards the poles, taking flora and fauna with it. Mountains however, are like little islands in this process. Because of the local climate that mountain ranges provide, climate regimes do not tend to move towards the poles, but towards the mountain tops; following the direction of decreasing temperature (Lenoir et al., 2008).

If cryophilic (low temperature favoring) vegetation is not able to adapt to the different climate, there will eventually be a moment in which it reaches the mountain top and cannot shift its habitat anymore. Eventually species will go (locally) extinct. However, as these regions are generally covered with heavy rock fall, microclimates created in small pockets of air which are shaded from direct sunlight like in figure 7 may provide shelter for some. In this bachelor thesis I will join the work of Robert Niederheiser in modeling designated areas for an analysis of the morphology (shape, form and texture) and thus microclimate potential. Apart from his work, I will focus on mapping the existing vegetation cover.

These models, together with the help of detailed information gathered by biologists and socials scientist, create a diverse data set which can be used to gather insight in the way alpine vegetation reacts to current climate change.

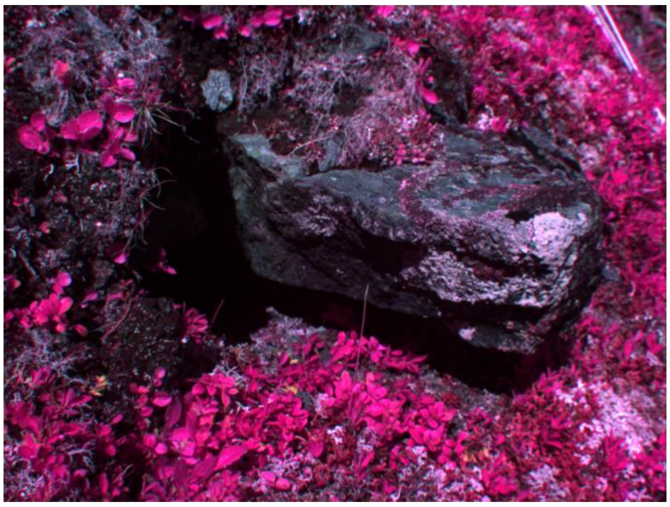


Figure 7. Overhanging rock with possible microclimate below (False colour image shot, taken with the multispectral camera)

2.2 Structure from motion

For the morphology analysis we use digital elevation models (DEM). There are numerous ways to get these, which will be described in the chapter Methodologies. As we only used the Structure from motion (SfM) approach, I will only elaborate on this method.

2.2.1 Main principle of stereo photogrammetry

SfM is a form of stereo photogrammetry. In stereo photogrammetry, two or more photos are used to make a three dimensional model out of two dimensional images; just like your brain is capable of making a three dimensional representation of the two dimensional images made by your own eyes. To gain some understanding in this process, we first have to focus on one picture alone.

Modern day cameras are equipped with a lens that focusses the light from a scene outside the camera onto a sensor inside the camera. The first cameras only had a screen with a very small hole in front of their light sensitive image receivers. As modern day cameras with near perfect lenses have the same geometric model as seen in this kind of camera, it is used often in computer vision science.

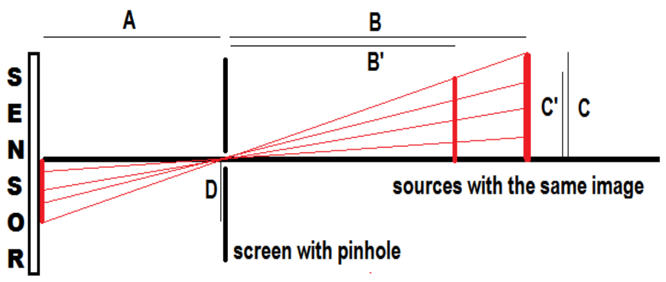


Figure 8. pinhole camera model

Figure 8 shows a pinhole camera model. As all light travels in straight lines and has to pass through the same point (the pinhole, also named principal point), the following formula holds by geometrical proofs

$$\frac{|A|}{|D|} = \frac{|B|}{|C|} = \frac{|B'|}{|C'|}$$

Figure 9. basis geometric formula used in the pinhole camera model

Since A is a given as a camera parameter (focal length), and D can be measured from the picture, we have some information about the photographed object which can help us to transform the image back to a 3D model. However, from the picture we cannot see any difference between the red bar in front and the one in the back.

This is where the second camera comes in. It makes a different picture from a different angle. Like the first camera, it only knows the ratio between the distance to the object, and its vertical height.

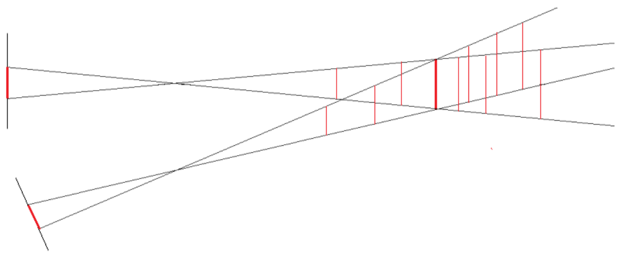


Figure 10. Two pictures combined

In figure 10 you see two pictures made from a red bar. With a single camera you are not able to know which red bar represents the real world position and size of the red bar. With the second camera combined there is only one position and size which suits both cameras. However, in this model, it is important to know the position of the camera. Without fixed camera positions, it would be possible to align any red bar in the field of one camera to another red bar with the same length in the view of the other camera. This problem is overcome by making pictures in stereo pairs of which the cameras differ with a known rotation and translation.

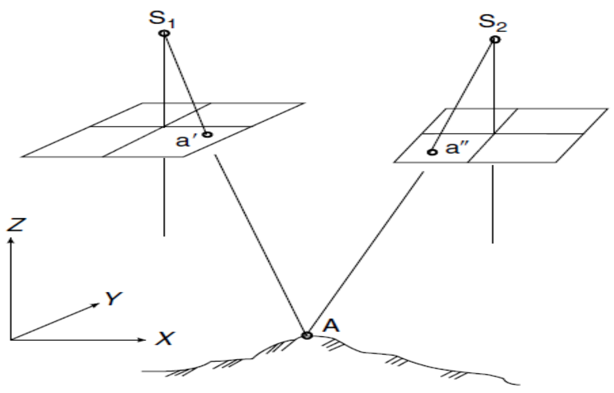


Figure 11. creation of elevation model (taken from Zhilin Li, 2004)

2.2.2 Elevation model

With this camera model we can look at the process of making a 3D model out of multiple pictures. Figure 11 shows the two cameras lens centres S1 and S2, and a terrain with point A (tie point). The planes in between represent the taken pictures with at a' and a'' the projection of tie point A on each picture. From the previous camera model we know that light travels in a straight line across the lens from object to sensor. This line represents the spatial relation between the three points. This relation is called the collinearity condition. This is given by the flowing formulas;

$$x = -f\frac{a_1(X_A - X_S) + b_1(Y_A - Y_S) + c_1(Z_A - Z_S)}{a_3(X_A - X_S) + b_3(Y_A - Y_S) + c_3(Z_A - Z_S)}$$

$$y = -f\frac{a_2(X_A - X_S) + b_2(Y_A - Y_S) + c_2(Z_A - Z_S)}{a_3(X_A - X_S) + b_3(Y_A - Y_S) + c_3(Z_A - Z_S)}$$

Figure 12. Collinearity functions. (taken from Zhilin Li, 2004)

With:

$$\begin{aligned}
a_1 &= \cos\phi\cos\kappa + \sin\phi\sin\omega\sin\kappa \\
b_1 &= \cos\phi\sin\kappa + \sin\phi\sin\omega\cos\kappa \\
c_1 &= \sin\phi\cos\omega \\
a_2 &= -\cos\omega\sin\kappa \\
b_2 &= \cos\omega\cos\kappa \\
c_2 &= \sin\omega \\
a_3 &= \sin\phi\cos\kappa + \cos\phi\sin\omega\sin\kappa \\
b_3 &= \sin\phi\sin\kappa - \cos\phi\sin\omega\cos\kappa \\
c_3 &= \cos\phi\cos\omega
\end{aligned}$$

Figure 13. Angular functions. (taken from Zhilin Li, 2004)

[x y] correspond to the coordinates within the picture frame, [f] is the distance between the S (camera lens centre) and the picture plane known as the focal length of the camera, [X_A Y_A Z_A] are the geodesic coordinates of point A, [X_S Y_S Z_S] are the geodesic coordinates of point S. [A_1 … … C_3] are factors which correct for camera roll, pitch and yaw [ϕ ω κ].

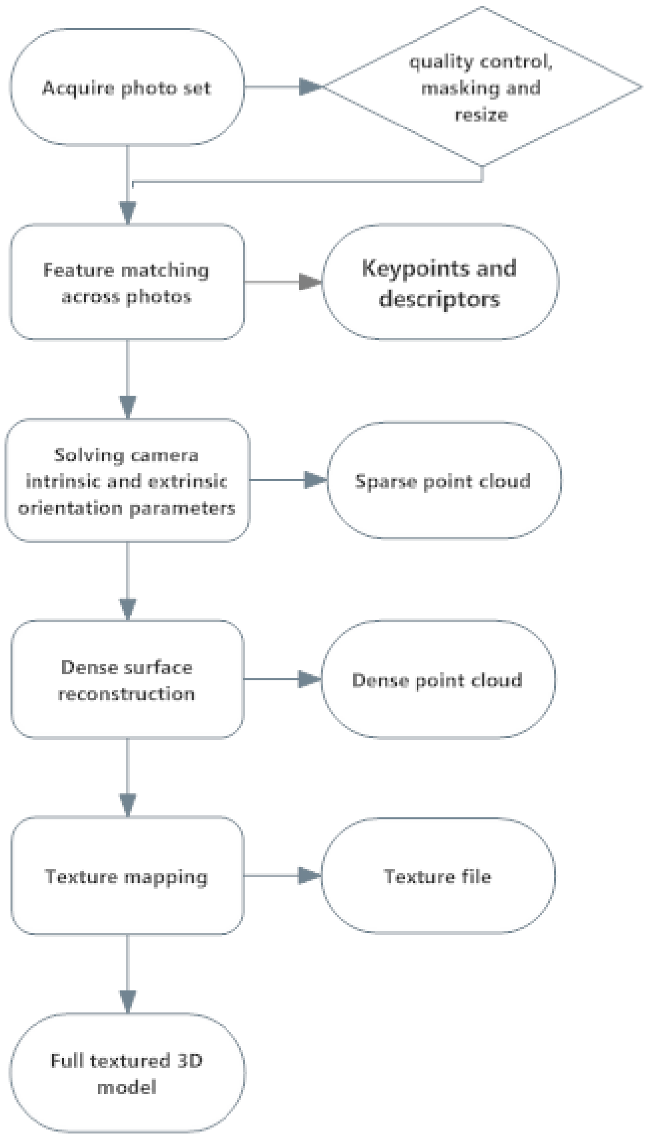
When the 6 degrees of freedom are known of the camera two camera's and x and y are measured on both photos, we can calculate [X_A Y_A Z_A] with the assumption that these is the same for both cameras. This assumption holds if point A is correctly matched between the two pictures.

2.2.3 Structure-from-Motion pipe-line

What makes structure from motion different from other stereo photogrammetry is that the 6 degrees of freedom are not known. By having redundant photos, these camera positions can be approached by looking at matching points in several photos. The program undergoes an iterative process of placing the cameras and constructing a the geometry of the scene. Because of slight errors and deviations of the matched points, there are more solutions for this geometry. This is solved by using a non-linear least squares minimisation. Shortly said it chooses the geometry which has the smallest value for the sum of all squared errors. (Westoby et al, 2012.) (Further explanation of least squares will be out the scope of this thesis) This process above takes place in the 3rd row of figure 14. The intrinsic parameters describe the inside of the camera such as focal length, sensor size and principal point. Extrinsic parameters describe the 6 camera orientation parameters.

We will now take two steps back in the process. It starts with acquiring a photo set. Either by making one, or downloading the whole search result of online photo server Flickr for the keywords "Rome" or "Coliseum" (Aragwal et al, 2011). This set may undergo a quality control, removing pictures that are out of focus or shot at a different subject etcetera. It may be downsized in pixel size or photo number to speed up the process. Finally, you could manually mask features that you are not interested in. However, this may cost a lot of time if you have hundreds of photos.

Next, a really important step, is the feature detection and matching. If we take figure 11 as an example, the algorithm has to detect a feature in picture one (a') and a feature in picture two (a'') and link them together to get to a realistic representation of point A. One of the most used algorithms to do this is called the Scale Invariant Feature Transform (SIFT) (Lowe, 2004).



Acquire photo set
quality control, masking and resize
Feature matching across photos
Keypoints and descriptors
Solving camera intrinsic and extrinsic orientation parameters
Sparse point cloud
Dense surface reconstruction
Dense point cloud
Texture mapping
Texture file
Full textured 3D model

Figure 14. Structure-from-motion workflow

SIFT works in two steps to find points which could potentially be interesting for matching photos. The first part consists of a detector that detects areas that contain abrupt changes which may be present in different pictures too. The second part is the descriptor, describing the point in such a way that it is recognised in a different picture which may be tilted, scaled or translated.

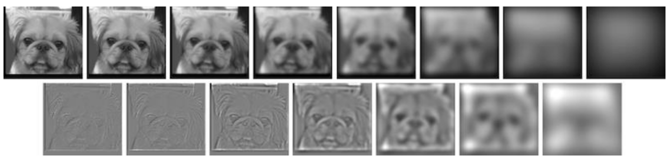


Figure 15. First line: One octave of Gaussians. Second line: Difference of Gaussians (taken from: Evan Wallis , 2011)

The detector starts by making the picture blurry by filtering out the highest frequencies (first line figure 15). Think of applying a Fourier transfer with a band pass filter in the frequency domain. For the novice reader: a Gaussian transform is applied. This is repeated several times creating a series of blurred images with in the end only the lowest frequencies in place. This first series is a so called octave. Now the original picture is sampled down form a new octave. 4 pixels are replaced by one new average pixel and the blurring process starts over again. After a few iterations you get a so called Gaussian pyramid (left side of figure 15). It is called a pyramid as the down sampled picture gets smaller each iteration. Next, within each octave, the blurred pictures are subtracted from each other, resulting in the difference of Gaussians(bottom row figure 15, right row figure 16). If we think of the Fourier transfer again, subtracting one bandwidth from the other, leaves out the response of the frequencies in between. So does the Difference of Gaussian (DOG). It holds information of the picture in a small part of the frequency domain. For example, it can be seen that the fourth picture of the second row of figure 15 has the most distinctive features. This means that this frequency domain is most present . Now with this information, the SIFT algorithm continues.

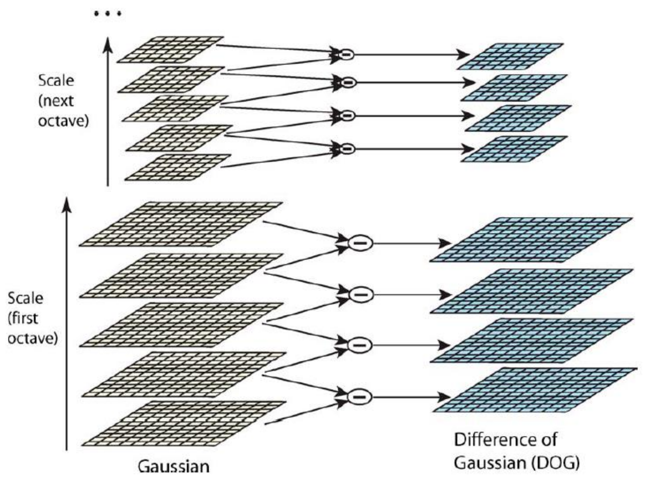


Figure 16. graphical representation of the pyramid of Gaussians and the Difference of Gaussians (taken from Lowe, 2004)

The algorithm looks for local minima and maxima. It finds these by looking at the 8 surrounding pixels within its DOG image, but also with 9 pixels in the DOG of the same frequency domain one octave below and 9 pixels one octave above (figure 17). It selects point to be a keypoint when a point is the largest or smallest value. Computational power (amount of calculations the processor has to make) is saved by stopping the check after one bigger and one smaller value is found.

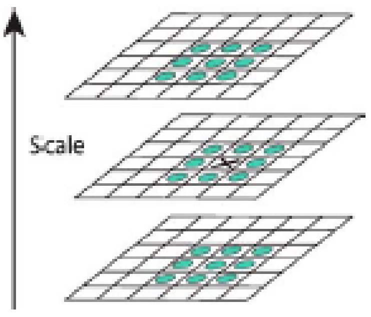


Figure 17. The pixel in the cross will be marked as possible tie point when it is the largest or smallest value compared to the pixels marked with the blue dots (taken from Lowe, 2004)

Once the keypoints have been found they undergo a selection procedure on quality.

The next step will be the descriptor. A 16 by 16 pixel square is selected around the area of interest. In each pixel the gradient is described by a vector. Each 4 by 4 pixel square is summarised by a 8 bin vector with the average vector direction of the 16 pixels inside its region. A 8 by 4 by 4 vector arises which acts as the descriptor. Two keypoints are a match when the two descriptors are nearest neighbour and the Euclidian distance in the 8 by 4 by 4 vector space (similar to Pythagoras theorem for a 2D vector) is minimal.

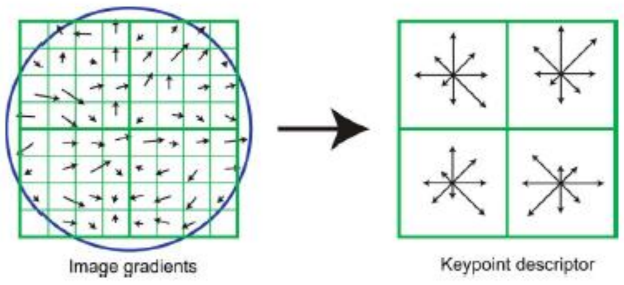


Figure 18. small version of the actual keypoint descriptor (taken from (Lowe, 2004))

Next step in figure in figure 14 is solving the camera parameters which is described above. After this you can build a dense surface model. This is done with algorithms that will not be dealt with in detail. Texture mapping is done by blending source images pixels with the same position on the model to a new pixel in a texture file which is made as overlay of a mesh made from the connecting points form the dense point cloud. Full textured 3D models like the one shown in figure 19 are the end result

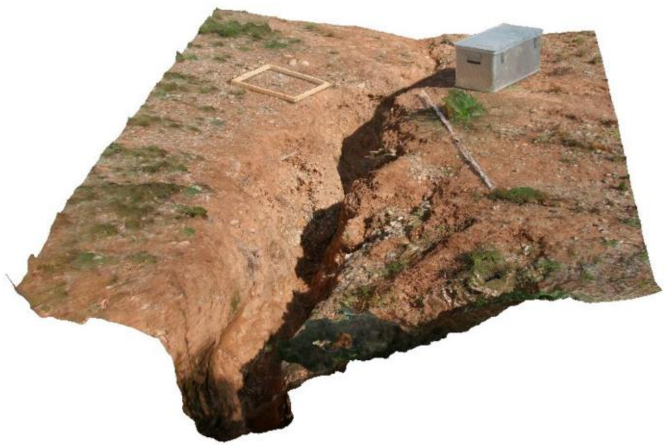


Figure 19. Example result of a SfM Workflow (taken from (Kaiser, 2014))

2.3 Vegetation classification; NDVI and False color imagery

When acquiring a new skill Is think it is important to play around. So before starting to read a lot of papers on vegetation classification, which I did on Structure from motion, which would send me too deep in the theory, it tried to come up with my own solutions from scratch for this part of the project. In this chapter I will guide the reader through my process learning what data I had within my reach and how I eventually used it in a vegetation classification tool made by Matlab (Matlab, 2011B)(see appendix).

As a start, I had the explore the camera that was given to me. As it is a multispectral camera, I had to figure out what kind of data I got out of it and why it looked that way. This made me understand False colour imagery again after it had sunken down a bit from the second year course on data processing and remote sensing within my bachelors. As an image which is displayed on a computer screen normally consists out of red, green and blue colours, a basic image is saved as a raster file with 3 layers in which each layer corresponds to the intensity of one colour. The combination of these colours covers the whole palate shown in figure x.

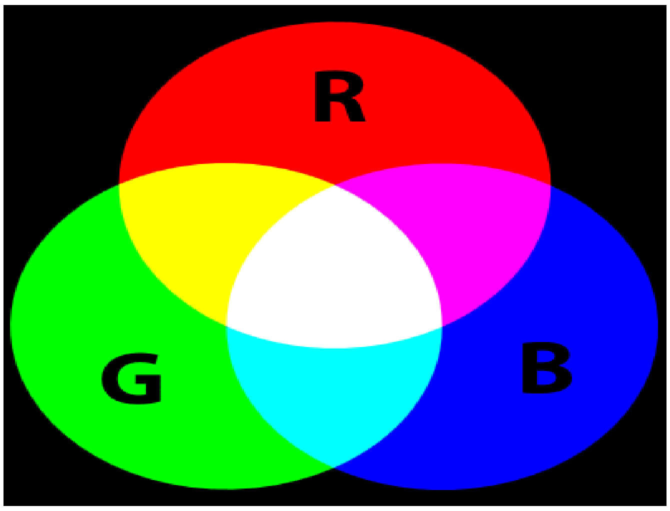


Figure 20. Additive RGB Colours

But what happens if you put different light intensities in the normal RGB bands? Quite simple, the image processing program on your PC will think the first band is red, the second band is green and the third band is blue. When looking at the images from the multispectral camera, it appeared that vegetation is always bright pink. Looking at the colours, this means it must have a high response in the red, and the blue band. Knowing from every day experience vegetation has a high response in the green band since we see it as green, I knew either the first or second band had to be green. Then I checked the manual of the camera to find out that R -> NIR, G -> RED B -> GRN. So I learned vegetation must have a high response to Near infra-red light to produce the magenta in the pictures. Later I learned from papers that this is the case. (Earth Observatory, NASA, 2015)

Next, I wrote a Matlab script to see what all the pixels of an image would do in a 3D space spanned by its values in different bands. Figure x shows such a result. Each pixel has its own place in the 3D space.

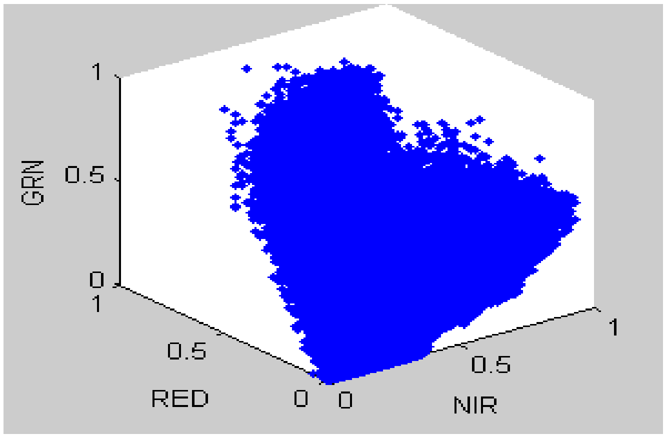


Figure 21. 3D space in which all the pixels of a photo are located. (Matlab, 2011B)

With some imagination you can see two clusters in this cloud, so it tried to cluster them and put them back in in the picture as two clusters.

First I tried an Euclidean distance clustering algorithm in the form of a k-means-method learned in the same second year course again. This did not work. Two clusters arose but one cluster was located at het origin and the other somewhere further It seemed that the pixels close by the origin have low values in any band, and thus they may be in the shade. So I did not want to cluster the pixels on the fact if they were shaded and dark or well lit. so I thought of not clustering 3D coordinates, but transforming the coordinates to spherical coordinates, and then leaving out the radius of the coordinate as this kind of represents the amount of light reflected. So I only clustered on the vectors on their angle with the x-axis and the z=0 plane. This worked quite well. But it takes some time to cluster all the pixels so I looked for a faster method.

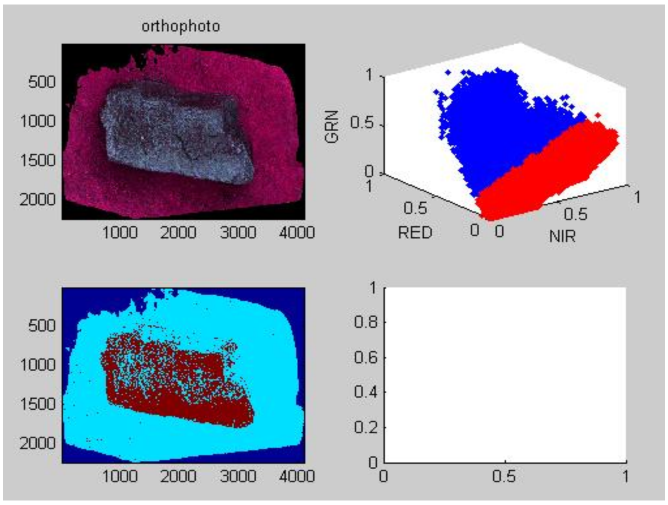


Figure 22. Classification of top left figure. Top right showing 2 clusters made with a K-means algorithm on Euclidian distance. Bottom left is the end result. (Matlab 2011B)

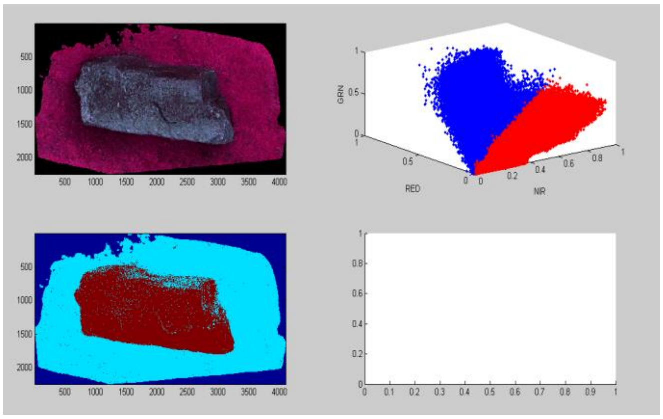


Figure 23. top left is the photo to be classified. top right are the two clusters based on their angular coordinates, bottom left is the resulting classification. (Matlab, 2011B)

The clustering process takes quite long, but it is seen in figure 23 that with the classification on angular coordinates the classification performs well on the sample. If we look at the cluster however, it looks like there is a plane going through the cluster, of which each pixel that is above this plane belongs to one class, and every pixel below this plane belongs to the other. Knowing the parameters of this plane would let us classify next images much faster as only a logical test has to be performed for each pixel. This is where NDVI comes in. It stands for Normalized Difference Vegetation index and it represents the ratio of intensity of visible light over the intensity Near infra-red light. Intensity of visible light is given as the red light only shown on the RED axis. The intensity of near infra-red light is given on the NIR pixel axis. so it turns out it is not even a plane in 3D space to split the data, but only a line in a 2D space spanned by NIR and red light. This makes the classifying method much faster. (It took me to the very end to realise the results were much better when I did not include the green light in the visible spectrum).

NDVI is given by NDVI = (NIR — VIS)/(NIR + VIS). VIS stands for visible light. The possible values that come out of this formula are between -1 (NIR = 0, VIS =1) and 1 (NIR = 1, VIS=0). Healthy vegetation will give a value of around 0.3 with grasslands around 0.2/0.3 till 0.8 for dense rain forests. .(NASA Earth Observatory, 2015)(Carslon and ripley, 1997)

However, these data is used from satellite images, so I did a small calibration to see which value would give me good results. This is shown in figure 24. It turned out that inserting a split of vegetation and rock with a NDVI value of 0.2 to 0.3 would give me good result. This split I will call 'cut-off'. Values above the cut-off are classified as vegetation, and values below as rock.

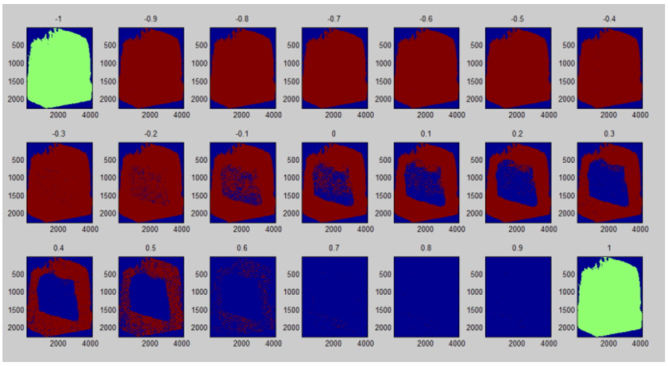


Figure 24. Classification with several NDVI values. 0.2 to 0.3 (middle row, two most right plots) gives the most distinctive results for this picture. (Matlab, 2011B)

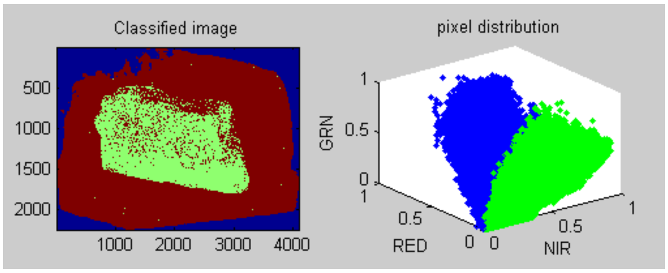


Figure 25. image classified with NDVI with a cut off at 0.25

2.4 The ADC Tetracam

The Agricultural Digital Camera (ADC) Tetracam is a camera specially developed for vegetation cover classification. The camera is a commercially developed by 'Tetracam Inc.' . and is used by scientists, agricultural businesses and educational programs on data acquisition and remote sensing (Tetracam, 2015).

2.4.1 Filter and sensor

The camera that is used to make the pictures has a 3.2 megapixel CMOS (Complementary metal–oxide–semiconductor) sensor with two filters attached. First, the light runs through a filter made from yellow glass, removing all the blue light from the spectrum. Second, the light runs through a checkerboard pattern Bayer RGB filter.

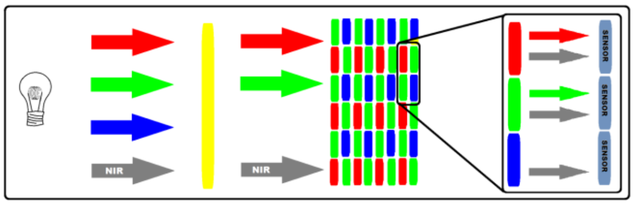


Figure 26 The camera setup with first a yellow glass blue filter, next to it the Bayer RGB filter.

The pixels that receive red and green light also receive near-infrared and add this to their response. Because the blue pixel only receives this NIR, the response of the green and red pixel values can be corrected with software by subtracting the NIR response.

The CMOS technology is used in all kinds of chips; it is basically the way to make an electric circuit on a small plate of semi conducting material. The CMOS sensor is an active pixel sensor (APS). Such a sensor consists out of an array of small components, named photo detector, which sense the presence of light. In the ADC camera, this array is placed behind the filters described above in such a way that each component writes out either RED+NIR, GRN + NIR or only NIR data. A combination of this data is later converted to the pixels you see on the screen. More details on this is given in the chapter 3.2 Data processing.

2.4.2 Lens and aperture

The camera is mounted with computer 8-mm fixed lens. Aperture (the mechanical iris of the camera) can be fixed anywhere between f/1.4 and f/16, although it can be fully closed too.

Focusing the photos is no easy job. The LCD-screen does not tell you if the picture is really sharp, and there is no view through a viewfinder. By experience in the field, the pictures that are made around 2 meters are sharp when the focus is in between the two stripes next to the infinity mark on the lens.

2.4.3 Pixel size and field-of-view

The pixel size depends of the distance between the camera and the object. From the manual we find that the field of view at a distance of 122 meter is 100 by 75 meter. Since we can scale these values as we assume light travels in a straight path, we can make a table of handy values which will be used later on to determine the desired height of the camera.

Distance (m)	Long edge (m)	Short edge (m)	Pixel size (mm)
1	0.82	0.61	0.40
1.2	1	0.75	0.49
1.63	1.33	1	0.65
2	1.64	1.23	0.80

Table 1. Handy values to keep in mind while taking shots from above.

2.4.4 Output

The output of the camera is a raw file which has to be processed with Pixelwrench 2.0 (Pixelwrench 2.0, 2015) software that comes with the camera. After processing, NIR, red and green light responses are stored In the first, second and third band respectively creating a false color image like figure 27

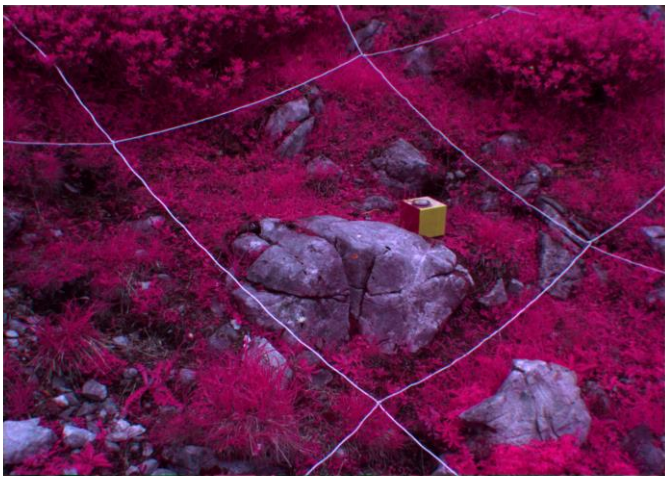


Figure 27. Example picture made with the ADC Tetracam

3. Methodologies

In this chapter I will elaborate on the way I gathered and processed the data with the help of a clear and easy data set. This data set is shot at the parking lot without the data acquisition protocol; however, it can be used in the processing part.

3.1 Data acquisition

The data acquisition was done together with Robert Niederheiser for later use in the MEDIALPS project. As this project is part of a bigger structure called GLORIA, MEDIALPS uses its method as a base of their workflow. Before explaining my own workflow, I will describe the GLORIA method

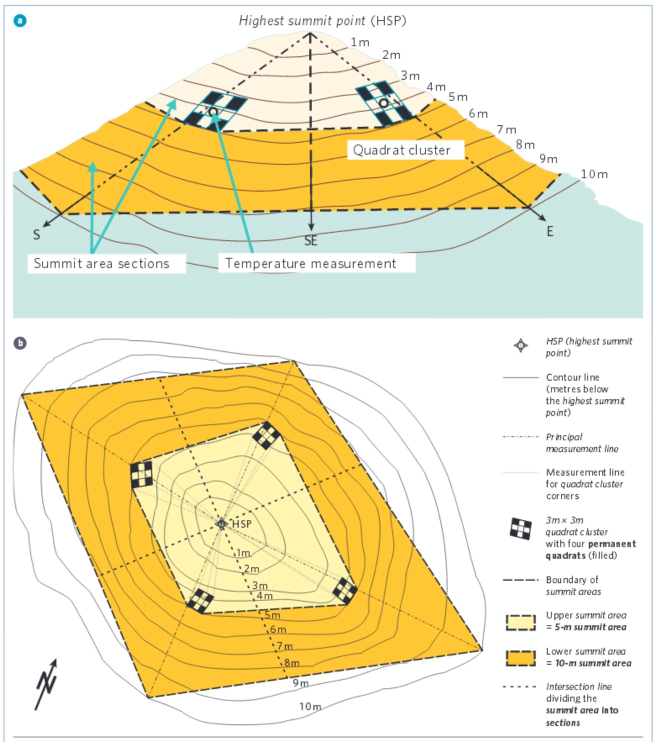

a
Highest summit point (HSP)
1m
2m
3m
4m
5m
6m
7m
8m
9m
10m
Quadrat cluster
S
SE
E
Summit area sections
Temperature measurement
b
HSP
HSP (highest summit point)
Contour line (metres below the highest summit point)
Principal measurement line
Measurement line for quadrat cluster corners
3m × 3m quadrat cluster with four permanent quadrats (filled)
Boundary of summit areas
Upper summit area = 5-m summit area
Lower summit area = 10-m summit area
Intersection line dividing the summit area into sections

Figure 28. Gloria method. (taken from GLORIA, 2015)

3.1.1 GLORIA and MEDIALPS workflow

'' The purpose of GLORIA is to establish and maintain a world-wide long-term observation network in alpine environments. --- *Vegetation and temperature data collected at the GLORIA sites will be used for discerning trends in species diversity and temperature. The data will be used to assess and predict losses in biodiversity and other threats to these fragile alpine ecosystems which are under accelerating climate change pressures*'' (GLORIA, 2015)

The method ensures researchers around the world to follow the same protocol while simplicity, comparability and low costs are ensured even in expedition conditions. After a suitable summit is chosen, the layout of figure X is made on the slopes. From the highest summit point (HSP) the four directions in between the cardinal directions are set out. These separate sections which are observed individually. Within these areas, on the line of the cardinal directions and at a height of approximately 5 meters below HSP, there is one 3-by-3-meter quadrat which is divided into 9 1-by-1-meter quadrants. The corner quadrants are sampled more thoroughly and should not be stepped in under any condition. Further explanation will not be of influence of my own workflow.

The MEDIALPS project stretches down some more. There are 3 more quadrants in each cardinal direction at the 25-, 45- and 65 meter elevation line. These are not monitored by any other Gloria associated group.

The workflow of capturing a plot started with building up the plot. As a start we place a rope around the outer edge of the plot. Next we place 2 rulers in two opposite corners for scaling. Then we placed an orientated cube (see figure 29) in each corner of the rope for model orientation. They are placed in such a way that one side always faces north and a bubble under a curved surface gives feedback on the levelling of the horizontal planes. Finally we add a sign with time and place specific information.

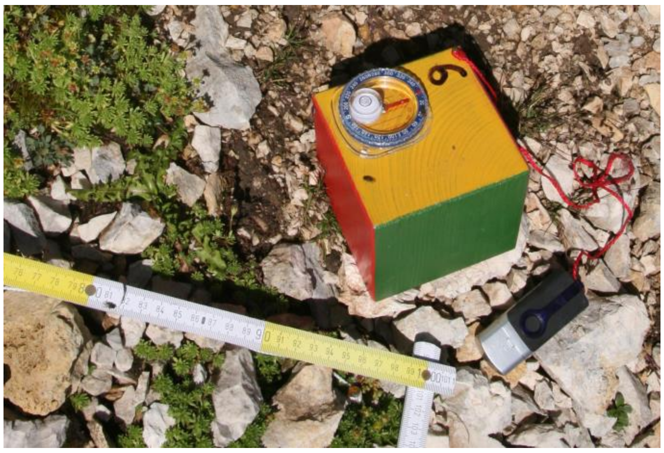


Figure 29. Cube with GPS, compass and level for model orientation.

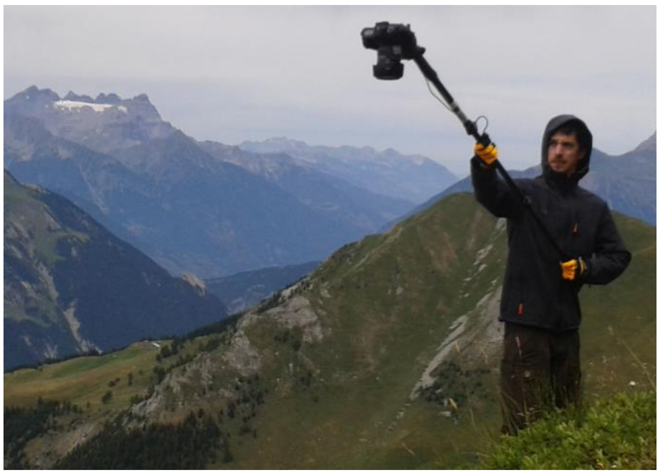


Figure 30. Robert taking a nadir (shot directly down) picture.

For the MEDIALPS project Robert would take around 300 pictures with a professional grade digital single lens-reflex (DSLR) camera (Canon 5D). Starting off with some scenery shots he would make a number of rounds making shots of the plot and the cubes in detail, followed by nadir-shots (shot in the direction of the centre of the earth) with the help of a rod and remote for the shutter creating a dome structure all around the plot.

3.1.2 Tetracam-workflow

The Tetracam-workflow is a short version of Robert his workflow. I only made the nadir shots with additional detail shots of interesting 3D structures to save time, battery and storage capacity. I made a small Matlab script helping me in making decisions and gathering insight how camera elevation and orientation influenced the amount of pictures needed for a certain overlap. This can be found in the appendix.

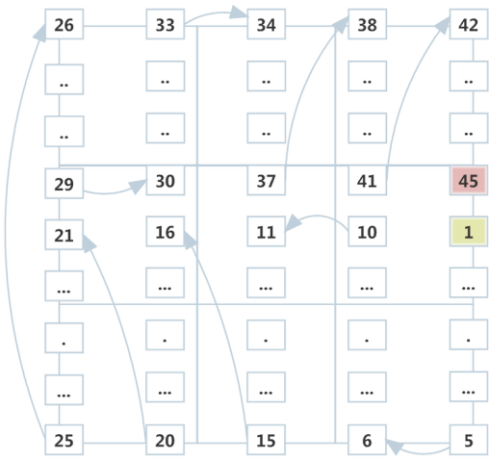


Figure 31. Possible trajectory of the camera with first (green) and last picture (red) indicated.

As a guide I took values from literature about aerial photography and aimed for at least 60 % overlap in one direction, and 30 % overlap in the other direction (G.C. Dickinson, 1969). With the help of the Matlab script I choose for a 60% overlap on the short edge of the photo, and 40% overlap on the long edge. This resulted in a minimum of 45 pictures at a height of 2 meter. The sequence of taking the pictures was planned, but eventually the terrain had a larger influence. Depending on the stability of my feet, a pattern like figure 31 was used to capture the photos.

During the whole workflow notes are taken on a designed paper for further information and convenient information redundancy at the desk. An example of such a paper is found in the appendix.

3.1.3 Logistics

As the plots are on local summits, logistics are a bit more difficult than a walk in the park. The first trip to Swiss we slept and ate in a hut, leaving out the need for camping gear. The second trip however, we slept in between sheep in a tent. Camping gear, food for a week and battery supplies had to be carried up the mountain by foot which took us a day, and started to hurt the knees. Unfortunately the work lasted only the short version of the schedule, which meant we had to carry all the uneaten food down again.

Figure 32. Accommodation during Italy trip

Planning and preparation of such trips will take time and energy and should not be under estimated by planning an arrival and departure within 24 hours without regarding it as one trip.

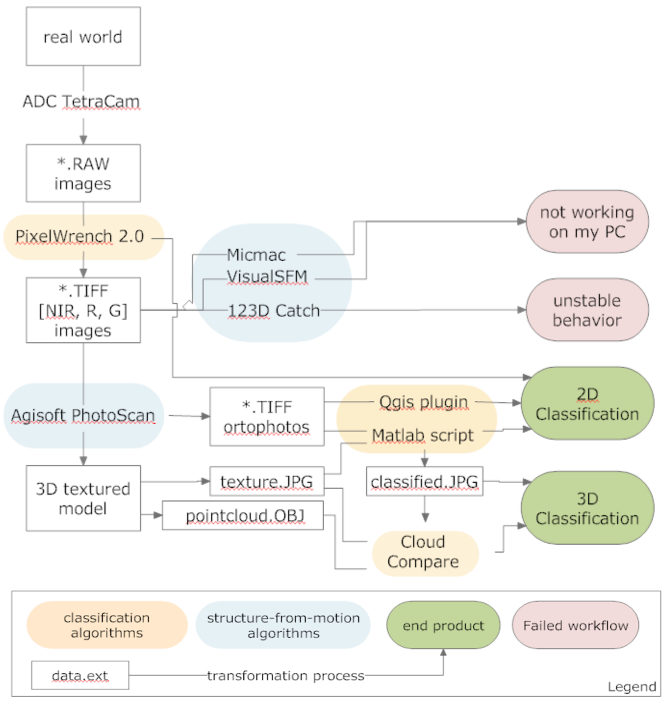


Figure 33. Workflow from picture to classification

3.2 Data processing

In this part of the chapter, I will manly focus on the software tools that I used and process steps that I came up with. The full flow chart in figure 26 is a rough guide to this chapter. The 27 test photos of that I shot of a rock in the parking lot will be used to explain the process. Besides, the clear difference in rock and vegetation makes is perfectly suited for a classification.

3.2.1 Preparing the pictures

The pictures were shot at a .RAW format, unreadable by other software than the accompanying software (PixelWrench 2.0) of the manufacturer of the camera. As a first step, all the pictures were converted to a .tiff format for further processing in a batch process. As the flow chart shows, the software could also perform a vegetation classification on the pictures. However, my objective is to make the classification on the entire plot in one go instead of using the single pictures. This will not be explored beyond this point.

When the pictures are converted they have to be sorted so they can be put in a database for further use. This is one big puzzle as some metadata was missing due to conditions on the mountain. This is where the redundancy of information on plot name etcetera comes in handy.

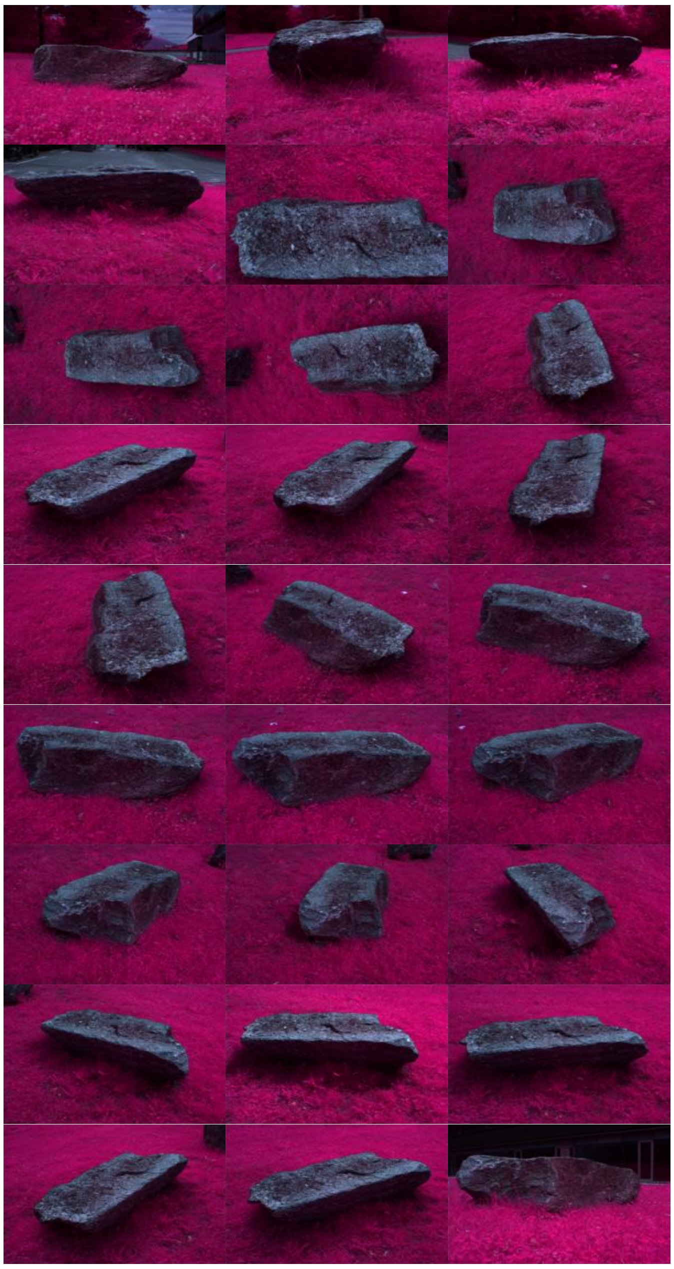


Figure 34. Test photos of the rock used for testing of the workflow

As a last step, the pictures which are used for further modelling must be checked on errors and focus.

3.2.2. Structure from motion modelling

To get the 3D model, I use the software Agisoft Photoscan (Agisoft, 2015) as for me it turned out to be the most reliable, easy-to-use package with reasonable results. This finding is backed up by (Remondino et al, 2014). The program follows the steps described in the chapter on SfM. First it makes the sparse point cloud and estimates the camera orientation at the same time. This model of a rock with 27 cameras has 5,663 keypoints in its sparse point cloud.

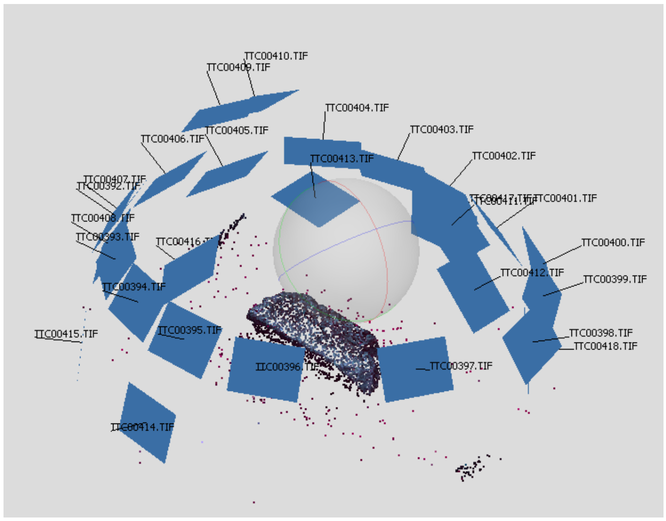


Figure 35. camera orientation and sparse point cloud (Agisoft)

Next, it makes the dense point cloud by finding more matches from the photos while they do not change their orientation anymore. Outliers from a smooth surface are filtered out at a certain degree by user's choice. As the model would not correspond to the actual shape of vegetation due to movement and too much detail, I choose the smoothest method to filter out misinterpreted vegetation cover. The resulting dense point cloud is shown in figure 36 and consists of 1.4 million points.

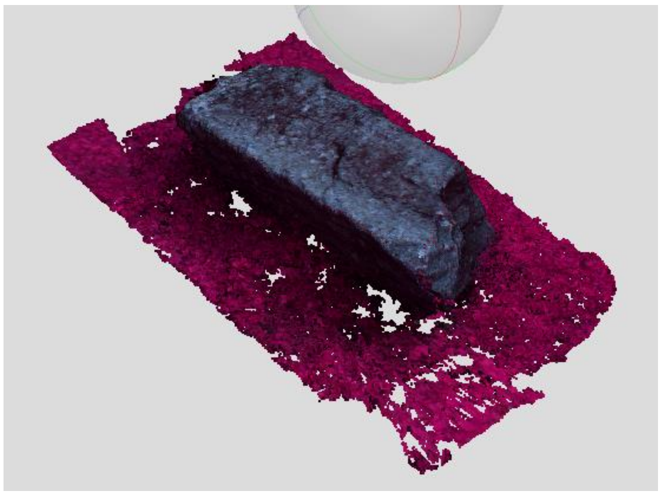


Figure 36. Dense point cloud (Agisoft)

In this model, the dense point cloud is so dense it almost looks like the finished model, but there are two last steps to be done. First an actual surface, or a mesh, will be generated by making triangles with (a down sampled amount of) points from the dense point cloud as corners. This is shown in figure 37. Next, a texture file can be created to fit this surface with more photo-like quality, shown in figure 38. This texture file, also called a ‘texture atlas’ can also be represented as a undistorted 2D image. This is shown in figure 39 and can be used for classification.

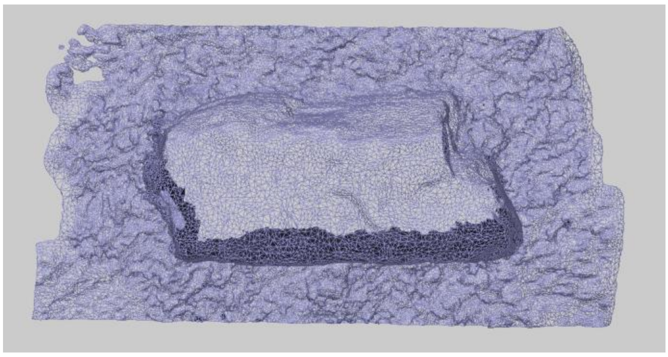


Figure 37. Mesh of traingles between keypoints (Agisoft)

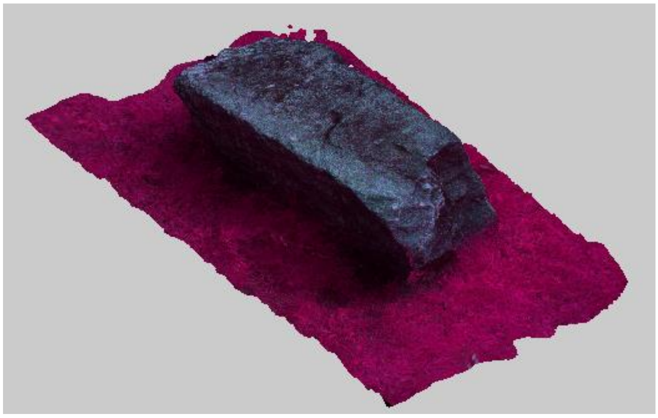


Figure 38. Textured model (The alert reader may have seen that this point of view is not given by a single photograph and thus shows the capabilities of SfM modelling) (Agisoft)

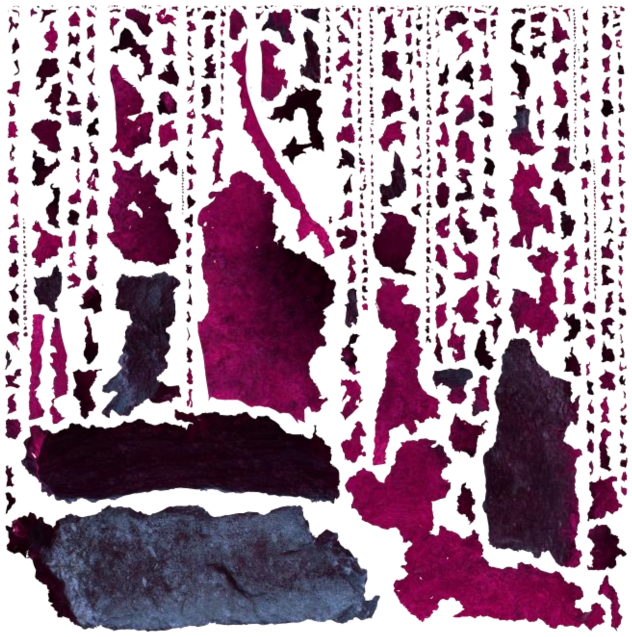


Figure 39. Texture atlas, a jig saw like assembly of all the faces of the model.

3.2.3 Vegetation classification

To perform the vegetation classification I followed a more experimental approach, and came up with my own guided user interface(GUI) made with Matlab.

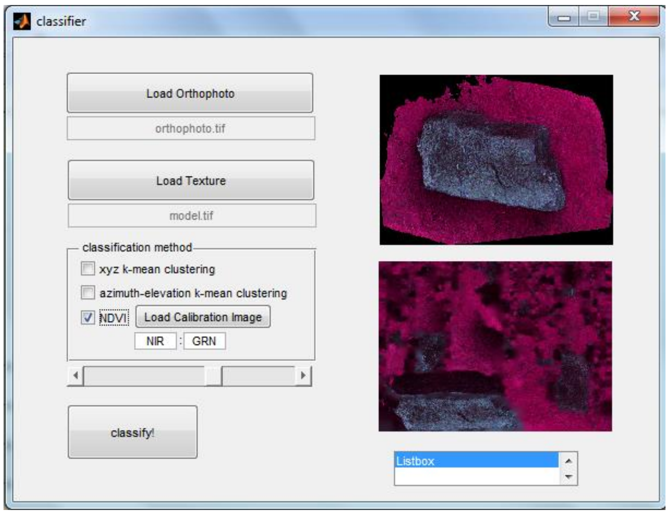


Figure 40. Guided user interface (GUI) for classification

It loads one orthophoto and one texture file. Both in tiff file format. You can choose a classification method, but only the NDVI is fully worked out as it gave the best results. The slider is used to set the cut-off NDVI value you want, but it is standardly set on 0,25. When you press ‘classify!’ it classifies both pictures. The result is shown in figure 41. The first column shows the mask which shows which pixels are classified.

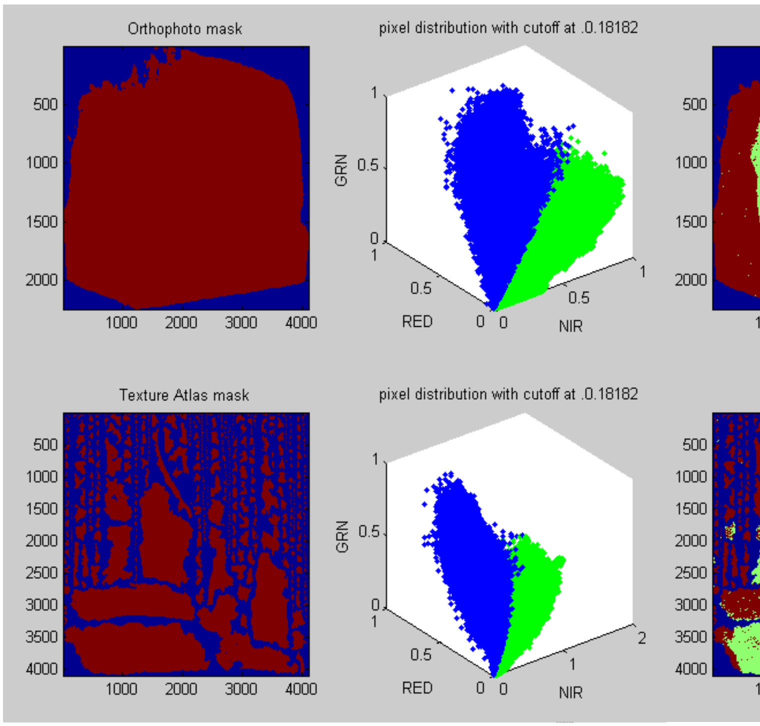


Figure 41. Results of the classification GUI

Especially for the texture file this is important as there are a lot of fake pixels in between the real data. The next column shows the distribution of pixels and their cluster. It seems there are two clusters, and by changing the cut off value a bit, you can try to get a nice split between the clusters. The third column shows the classified image with two classes. One rock, and one vegetation. The top right corner shows the ratio vegetation/rock as a function of their area for both the orthophoto as the texture image. In the right lower corner it shows the sensitivity for a different cut-off. By chancing the cut off, the ratio changes as well. This graph shows their relationship for a the given cut-off ± 0.05. Finally, the texture file is saved so it can be applied to the model again to get a classified 3d model. this is viewed with the software Cloud Compare, which simply loads the files without further hassle.

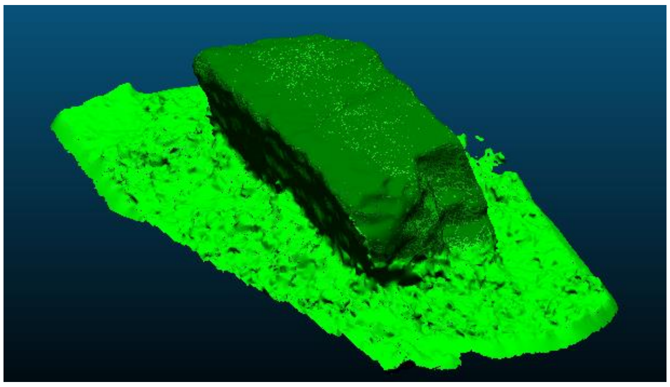


Figure 42. classified model shown in Cloud Compare

4. Results

In this chapter, I will describe the results from the whole data acquisition in the part 'Database'. Next, 3 plots are highlighted and processed as described in the methodology chapter.

4.1 Database

The database gotten from al the fieldwork is made viewable in figure 43. Each figure is named after one top, each circle within a figure corresponds to a certain elevation line and the four clusters of squares correspond to the cardinal directions (North, South, West and East). Each square corresponds to a 3 by 3 meter plot and the number inside the square represents the number of photos of that plot. It could be seen as a schematically overlay of the topographical location on the summit

A plot is indicated red if it is not captured or has theoretical too little photos to make a full overview orthophoto out of the pictures as the plot is not fully captured. Yellow indicates a poor coverage of photos. It may be enough to make an orthophoto when a corresponding DEM is provided. Green indicates a high amount of photos, possibly enough to make a 3D model.

The photos are stored within folders of their own plot name which is a combination of its cardinal direction and its elevation, e.g. 'N65' or 'S05'. These folders are sorted according to their summit and country code.

4.1.1 Statistics

Some basic total results can be summed up looking at this figure. A total of 3 summits where recorded with a planned amount of 48 plots.

8 out of 48 plots were not captured. All 8 because of a breakdown of the camera lens after a tumble down the mountain. The first 4 were not captured because it could not be fixed on the mountain, the other four because I had to wait for the new lens to arrive before I could join Robert in his fieldwork.

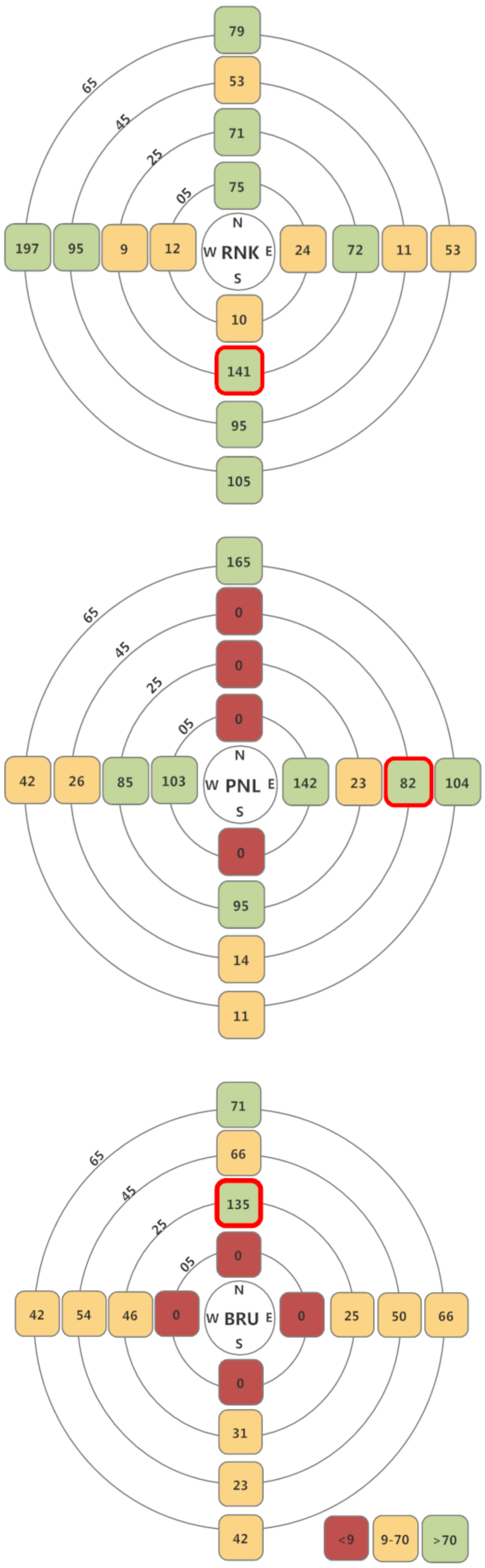


Figure 43. The three summits with their plots

22 out of 48 plots were captured with little coverage. This was mostly due to a large vegetation cover, which is thought to interfere with the SfM workflow. A faster protocol was followed for efficiency (detailed shots were left out). At some plots it was simply too difficult to perform a full protocol due to the bad maneuverability in the terrain, which forced me to follow a shorter protocol skipping some angles. Results will depend of picture quality which is not evaluated in this step yet.

18 out of 48 plots were captured with reasonable to good coverage for SfM modelling. Again, picture quality is not evaluated yet.

A total of 2715 pictures are taken of 40 plots. The raw pictures have a combined size of 7.94 GB.

Below, I will zoom in into only a few plots of which the photos will be evaluated.

4.1.2 'BRU_N25', 'PNL_E45' and 'RNK_S25'

'BRU_N25', 'PNL_E45' and 'RNK_S25' are further tested and modeled, they are selected to have at least one model a summit, the number of photos is redundant and a quick look shows good quality photos and some had interesting remarks in the information form or within the plot itself.

plot	Camera malfunction	Unfocused	Total errors	percentage
NL_E45	3	4	7/82	8.5%
RNK_S25	3	10	13/141	9.2%
BRU_N25	6	2	8/135	5.9%

Table 2. Information on picture quality of the chosen plots

Before modeling I looked at the datasets to look for errors and unfocused pictures such as seen in figure 37 and 38. Errors due to the camera would include skipped lines and insertion of base colors. Unfocussed pictures were mostly moved too much during capture. the rolling shutter effect (the effect of curvature in the picture due to movement of the camera during capture) did not came up in these sets. The outcome is shown in table 2.

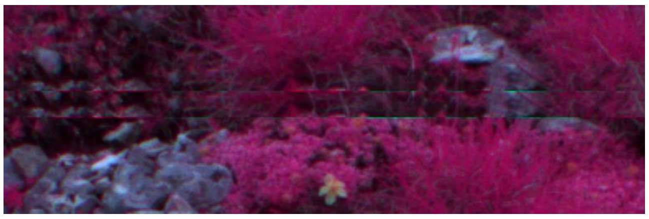


Figure 44. Camera errors

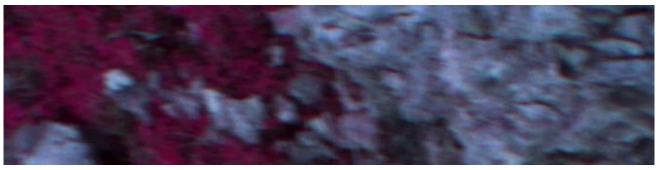


Figure 45. Unfocused picture

4.2 results of the Structure from motion workflow

Table 3 shows the SfM properties of the three plots.

Plot name	Photos aligned from total	Sparse point cloud (points)	Dense point cloud (points)
PNL_E45	78/78	6.828	7.412.834
RNK_S25	139/139	10.430	8.281.982
BRU_N25	131/131	22.751	7.628.776

Table 3. Information about point cloud sizes

Just to get an idea of the resolution, we can make a small calculation. We assume the points are about evenly spread, so we can get the spacing of points by dividing the length of one side, 300 cm, with the square root of the size of the dense point cloud. As all three models have around 8 million points, we take this for a quick calculation. It turns out the points are spaced only 1 mm apart from each other. This corresponds to the pixel size of the pictures shot at 2 meter, which I did. A higher resolution of the dense point cloud than this cannot really be expected.

The general accuracy is difficult to assess without going in much detail of the separate plots but figure 46 suggest an local error within a few centimetres as one edge of the cube is 10 cm.

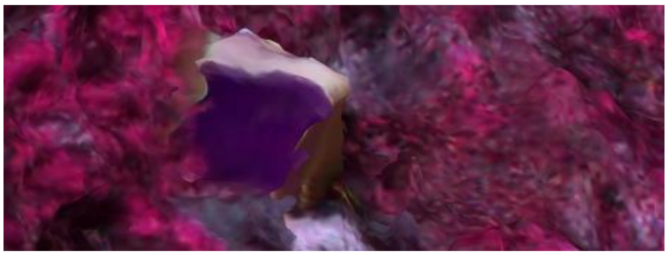


Figure 46. Results of a cube with 10 cm long edges in the middle of plot BRU_N25

4.2.1 Plots in detail

An overview picture of the model of BRU_N25 in figure 47 shows that there is a lot of vegetation in this plot. Also, the white line on the left side is drawn multiple times, indicating an incorrect model in this area.

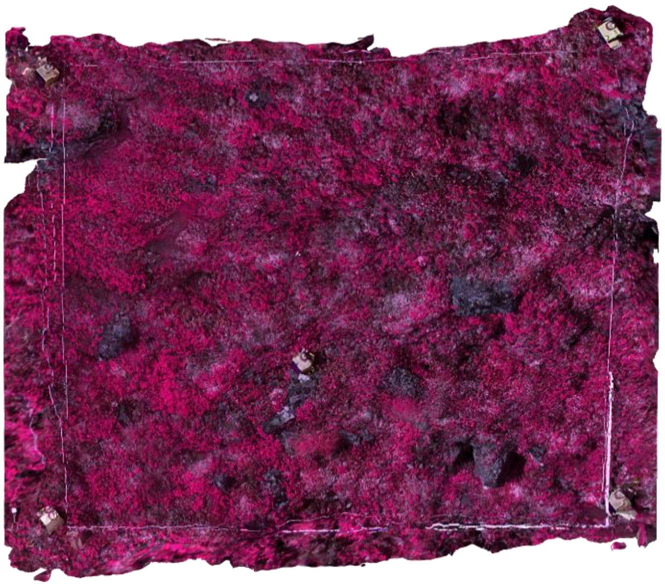


Figure 47. Orthophoto of 'BRU_N25'

This plot would potentially be difficult to model, indeed it can be seen that global dome structures are drawn over the vegetated areas instead of all the separate branches. Again it seems to be locally accurate in the cm scale.(figure 48). Larger error scales ranging from one side of the plot to the other are difficult to assess.

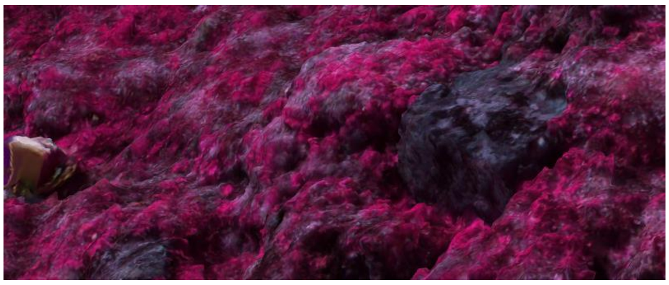


Figure 48. detail shot of smooth vegetation in contrast to a sharper rock

Figure 49 shows a big difference in precision between the overhanging rock in the left and the cube in the right. As the rock got my attention, I made detailed shots of this feature. The cube however, is only captured on the standard nadir shots. As a big difference in quality can be seen, it stresses the fact that redundancy of overlapping picture did help here.

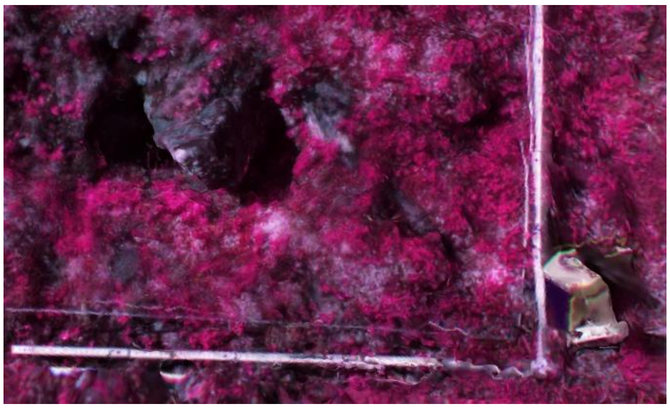


Figure 49. Difference in accuracy between a fully captured rock and a sparsely captured cube.

This model would not really be accurate to analyse the morphology of the rocks and soil, as the vegetation cover blocks to much. The only interesting parts are rocks that stick out which may provide shelter.

The 3D model and the orthophoto give a similar view on the classification. Some bigger rock structures are shown in figure 44. they are covered with lots of light green spots which indicate vegetation. It does not get clear if these may be mosses of that is has a different reason

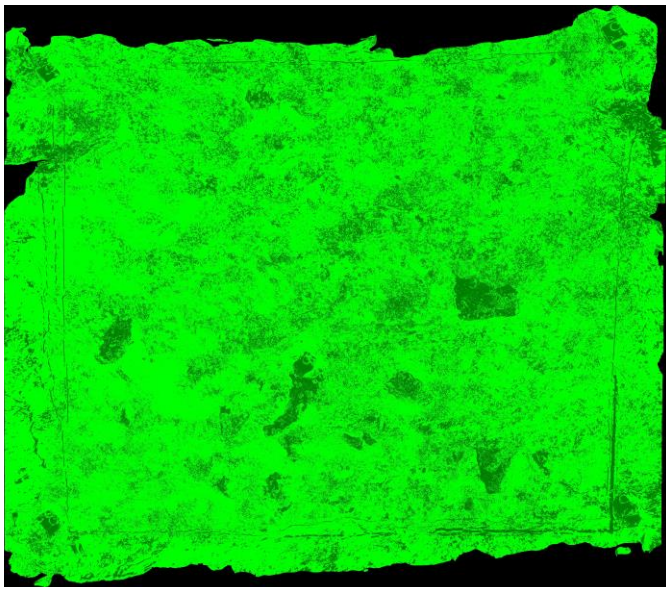


Figure 50. Classified orthophoto.

PNL_E45 has much less vegetation if we look at the orthophoto in figure 51. Just as the previous one it has some mistakes at the side, in this case the right side does not line up in the middle which happens to be a difficult to match vegetation part.

.

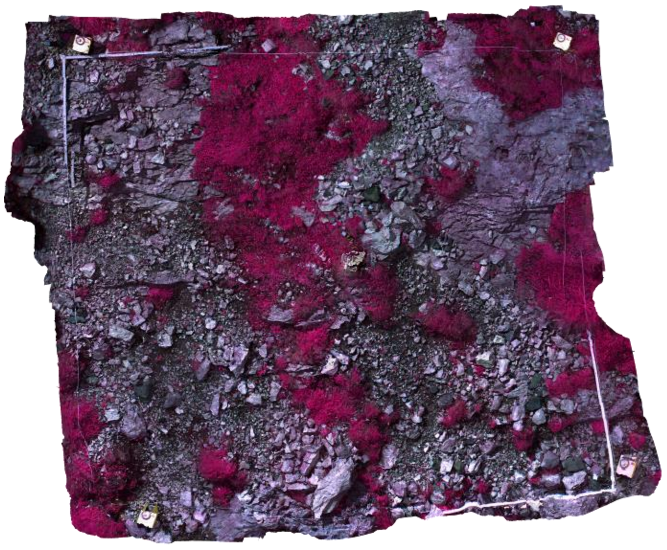


Figure 51. Orthophoto PNL_E45

The rocks show some good relief in the model expecting it to be accurate. However, if we look at the known shape of the cube in the middle of the plot we see that it is not accurate at all. The cubes at the corners look just as bad. This makes is really difficult to trust the model, as we do not know the shape of the rocks. It may look reasonable, but maybe it is not representative at all. It may also be a problem that the cubes have to little keypoints and too sharp edges in contrast with the rocks, making them harder to model.

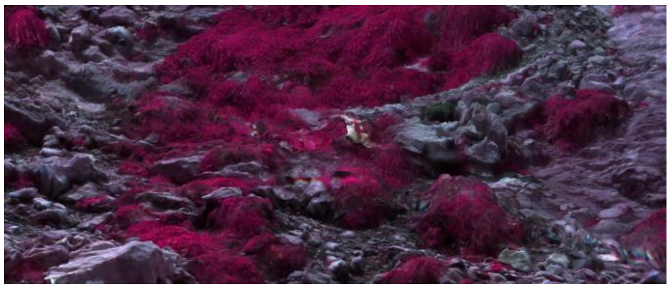


Figure 52. Detail of the cube in the middle.

The classification however, gives a really distinctive difference between the two classes. The rocks are fresh rocks with little to no algae and mosses on them, which may make the process easier.

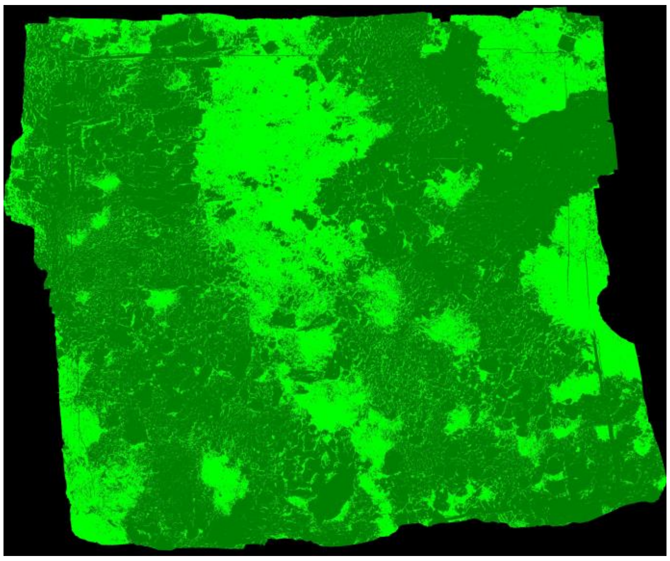


Figure 53. Classified orthophoto PNL_E45.

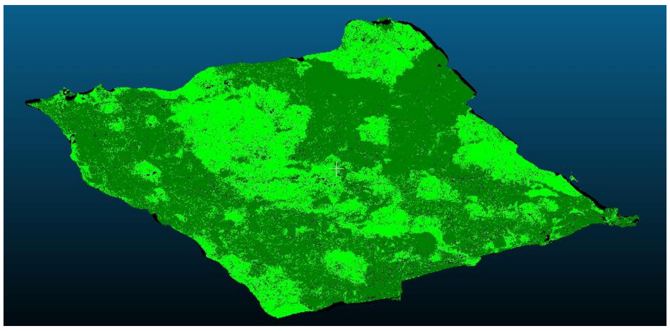


Figure 54. Classified model PNL_E45 (Shown in Cloud Compare)

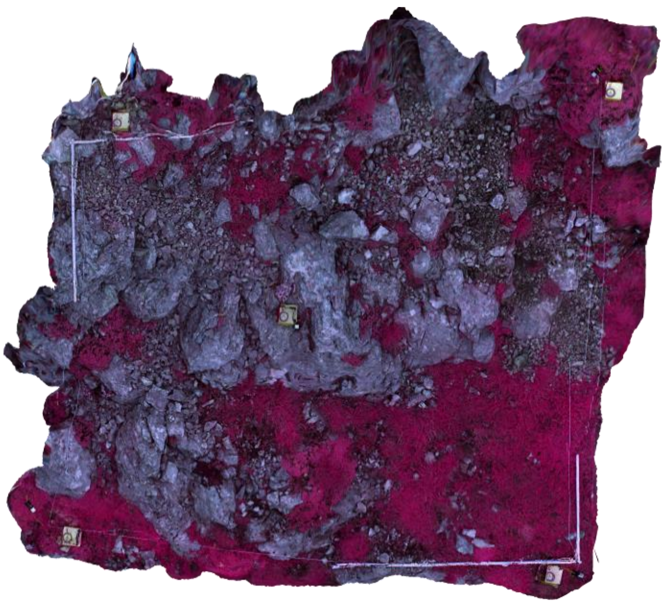


Figure 55. Orthophoto RNK_S25

Last be not least, RNK_S25. It can be seen in the orthophoto that this is the steepest plot (orientation and slope are not dealt with on purpose) as is seems a bit rectangular instead of square. Indeed, the front photo can be seen in figure 56 showing big differences in elevation

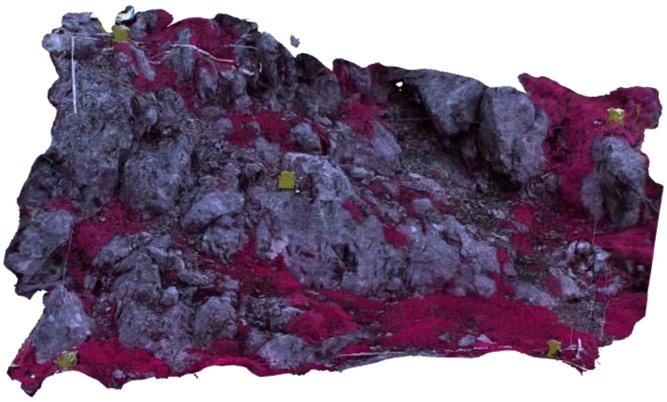


Figure 56. frontal 'orthophoto' RNK_S25

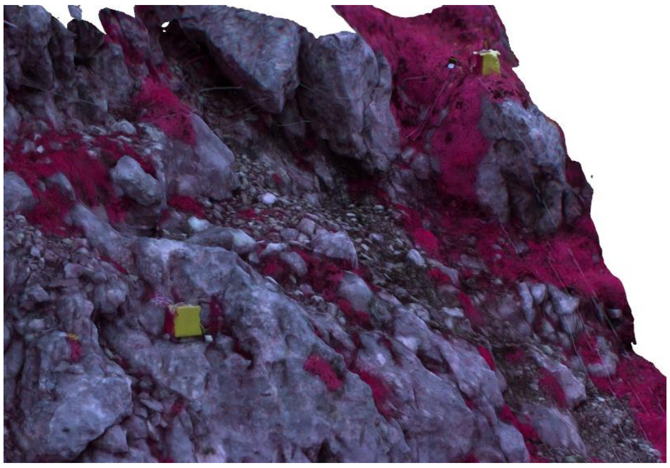


Figure 57. Section of RNKS25 with clear 3D structures

Figure 57 shows large 3D structures which are not really seen on the orthophoto. This may lead to interesting differences in the ratio of vegetation/rock within this plot. Also figure 57 shows sharp edges at the cubes, indicating a possible accurate model.

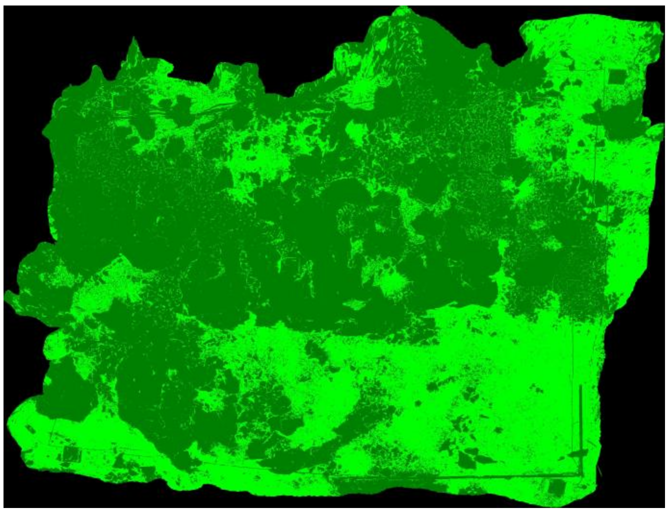


Figure 58. Classified orthophoto RNK_S25

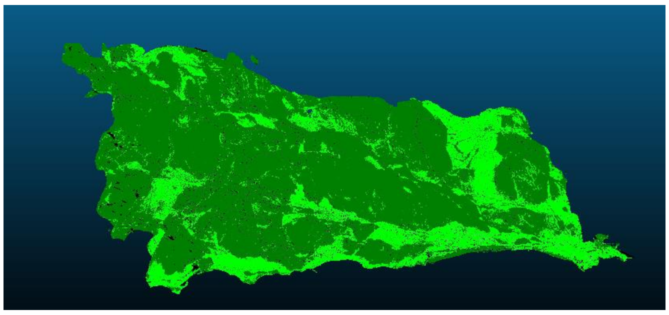


Figure 59. 3D classified model (Cloud Compare)

The edges between vegetation and rock seem sharp, there is only few scatter of the other class within large regions and it corresponds well to the unclassified image. This model is really good classifiable.

4.2.2 Vegetation/rock ratio difference for 2- and 3D

What is interesting know is whether it makes any difference to classify the textured model or just the orthophoto. Table 4 gives these values

Plot name	Orthophoto (V/R area)	Textured model (V/R area)	Relative difference (Orth/Text)
PNL_E45	0.48	0.51	0.941
RNK_S25	0.60	0.47	1,277
BRU_N25	3.44	3.17	1.08

Table 4. differences between Orthophoto and textured model

Indeed, RNK_S25 has the biggest difference in vegetation cover for the two techniques as we expected. For the other 2 the difference is twice as small. So you would say it does not really matter. However if we look further than the scope of this thesis, a 3D textured classification is much more convenient when comparison studies in different timescales would to be made. If we would repeat this process of data acquisition in a couple of years you would be able to do a change detection within the 3D model, making it possible to gather information about erosion, plant growth in the 3D domain and soil sedimentation. Such information would be more difficult to extract from a orthophoto.

5. Conclusion and Recommondations

In the conclusion we come back at the main research question

- How can the multispectral camera be used to detect vegetation on a rocky slope?

By answering the sub questions we deal with the main research question.

- *How to classify vegetation from NIR-, GRN- and RED light-intensity?*
  For classifying vegetation we saw that the NDVI index works really well in making a classification decision on light intensities of the radiation in the red and NIR spectrum.

- *How should you gather data on the site for easy processing and accurate models?*
  With care and consistency, as we saw in the data base description only a few were suitable for 3D modelling as too few pictures were taken from a large amount of plots. Plots that have more overlap and have more than 100 pictures including detail shots of outstanding features give promising results. Multiple photos at almost the same place give bad results. A distance of about 20 to 30 centimetres apart gives more accurate models.

- *Are the results of the camera suitable for 3D Structure-from-Motion modelling?*
  Yes, we see in all the results that are presented that there is no problem with making 3D models with the camera because of the false colour image, low resolution or what so else. You do have to be careful to keep the camera still while taking pictures, as blurry pictures occur easily.

- *Is the resolution of the 3D model high enough to make a micro topography analysis?*
  this depends on what scale you want to analyse the data. Rocks smaller than 5cm seem to be evened out by the algorithm. Larger features do stick out. For the scale and plots that are made in this thesis, I think analysis of the micro topography would not give you much more information than analysing a set of detail pictures by hand would as no automated procedure would be able to get the small scale topography from the model that is not captured by the structure from motion. or you would have to revise the whole structure from motion algorithms which would give you new models on which you can do you analysis.

5.1 Recommendations or further research

To make this extend this research to a further extend it would make sense to include work on scaling and orientating the model in the real world. This would be possible with the dataset as the cubes are used for this function.

Besides, quality control is left out for a large extent in this report. Accuracy of the pictures to begin with and quality of the 3D models is not accurately defined. Vegetation classification is only assessed on base of the pictures itself.

The data acquisition took quite long as we had to do everything twice as this was complementary to the Medialps project. Purely looking at this subject, bypassing the actual research question of Medialps, I would try to find a camera that can record RGB and NIR response at the same time so you do not have to do the acquisition twice. Mounting a lightweight stereo pair (or a device such as a Microsoft Kinect) on the rod instead of a single camera would be interesting to see if it boosts up the quality compared to one high quality sensor.

Finally incorporating drones in the image acquisition part would be interesting to develop, as precise objects for localisation (the cubes) are within the scene all the time. The hard winds in the mountain are a reason why current drone systems are not used in this study, but with new technologies come new possibilities.

Appendix

List of used software

Pixelwrench 2.0	http://www.tetracam.com/Products_PixelWrench2.htm
Agisoft Photoscan	http://www.agisoft.com/
Matlab (multiple versions)	http://nl.mathworks.com/products/matlab/
Cloud Compare V2.5.5.2	http://www.danielgm.net/cc/

References

Agisoft, 2014: Agisoft, "About Agisoft", 2015, http://www.agisoft.com/about/ (last visited: 28-8-2015)

Aragwal et al, 2011: Agarwala, S., Furukawaa, Y., Snavely, N., Simonb, I., Curless, B., .Seitz, S.M. , Szeliski R., 2011. '*Building Rome in a Day*' . Communications of the ACM vol. 54 no. 10 pp 105-112.

Carslon and ripley, 1997: Bechtel, A.; Puttmann, W.; Carlson, T.N.; Ripley, D.A. 1997 '*On the Relation between NDVI, Fractional Vegetation Cover, and Leaf Area Index*' Remote Sensing of Environment, Vol. 62, No. 3, pp. 241-252

Earth Observatory, NASA, 2015: John weier, David Herring, "Measuring Vegetation (NDVI & EVI)", 2000 http://earthobservatory.nasa.gov/Features/MeasuringVegetation/ (last visited: 28-8-2015)

Evan Wallice , 2011: Wallice E., **"**CS 143: Hybrid Images", 2011, *http://cs.brown.edu/courses/cs143/2011/results/proj1/edwallac/* (last visited: 28-8-2015)

G.C. Dickinson, 1969: G. C. Dickinson, 1969. **Maps and Air Photographs**, Hodder & Stoughton Educational.

GLORIA, 2015: Gloria **"**The purpose of GLORIA", 2015, http://www.gloria.ac.at/?a=2 (last visited: 28-8-2015)

Kaiser, 2014: Kaiser, A.; Neugirg, F.; Rock, G.; Müller, C.; Haas, F.; Ries, J.; Schmidt, J. 2014 '*Small-Scale Surface Reconstruction and Volume Calculation of Soil Erosion in Complex Moroccan Gully Morphology Using Structure from Motion*'. Remote Sensing. Vol *6*, pp 7050-7080.

Lenoir et al, 2008: Lenoir, J., Gégout, J. C., Marquet, P. A., de Ruffray, P., Brisse H., 2008. *A Significant Upward Shift in Plant Species Optimum Elevation During the 20th Century.* Science Vol. 320 no. 5884 pp. 1768-1771.

Lowe, 2004: Lowe D., 2004 '*Distinctive image features from scale-invariant keypoint*' International. Journal on Computer Vision . vol 60 no. 2 pp 91–110.

Medialps, 2015: Medialps, "Medialps", 2015 *http://www.mountainresearch.at/index.php/en/projects/ongoing-projects/406-Medialps.html* (last visited: 28-8-2015)

Remondino et al, 2014) Remondino F., Spera M. G., Nocerino E., Menna F., Nex F. 2014**. '***State of the art in high density image matching'.* The Photogrammetric Record vol. 29 no. 146 pp. 144–166 (June 2014)

Tetracam, 2015: Tetracam, "About Tetracam", 2015, http://www.tetracam.com/Company.htm (last visited: 28-8-2015)

Westoby et al, 2012: Westoby, M.J., Brasington, J., Glasser, N.F., Hambrey, M.J., Reynolds, J.M., 2012. *'Structure-from-Motion' photogrammetry: A low-cost, effective tool for geoscience applications.'* Geomorphology Vol 179 pp 300-314.

Zhilin Li, 2004: Zhilin Li, Christopher Zhu, Chris Gold, 2004. **Digital Terrain Modeling: Principles and Methodology.** CRC Press.

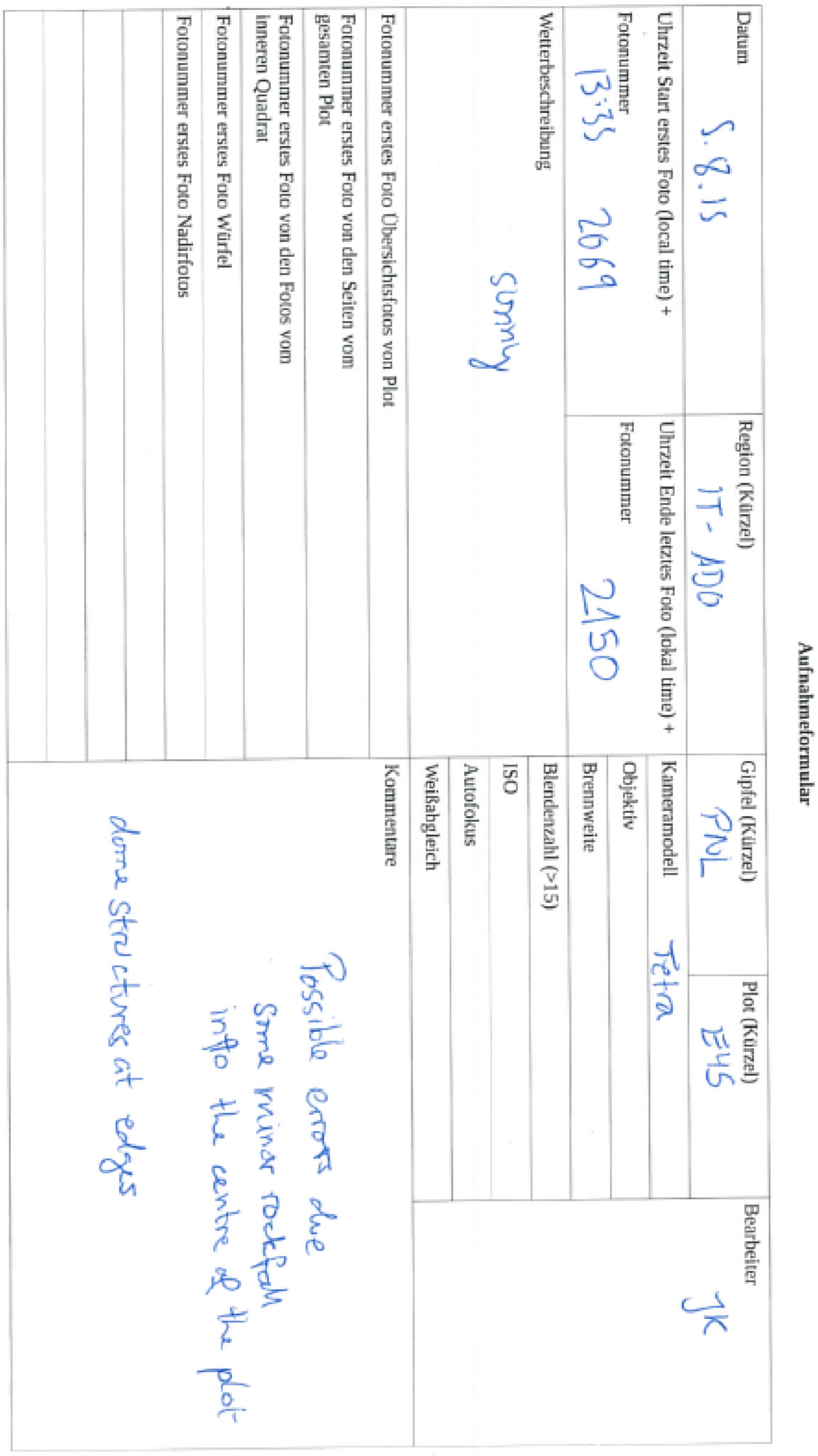
Wordlist and abbreviations

2D	Two dimensional
3D	Three dimensional
ADC	Agricultural Digital Camera
Aperture	Iris of the camera
Base colors	red, green and blue
BRU, RNK, PNL	Codes for the summit locations
Cardinal directions	North, east, south, west.
CMOS	Complementary metal–oxide–semiconductor
Collinearity condition	relation between camera parameters and feature location within image and real world coordinates
Cryophilic	Having an affinity for or thriving at low temperatures.
DEM	Digital elevation model
DOG	Difference Of Gaussians
DSLR	Digital single lens reflex
Focal length	A lens focusses parallel light at a plane which is the focal length away from the lens center
Gaussian pyramid	sequence of images convoluted with a Gaussian kernel in different down sampled scales
GLORIA	see Methodologies/Data acquisition
GRN	greenlight
GUI	Guided User Interface
IGF	Institute for Interdisciplinary Mountain Research
Keypoint	feature of interest in image matching
MEDIALPS	see Background/MEDIALPS
Morphology	The form and structure of anything
Nadir shots	down facing shots.
NDVI	Normalized differential vegetation index

NIR	Near infra-red
Orthophoto	Photo corrected for topographic relief, lens distortion, and camera tilt so it has the same property as a map. It can be used for measuring true horizontal distance.
Pinhole camera model	the model of ancient cameras with only a pinhole as lens which still works for modern cameras
Plot	3 by 3 meter region of interest
Principal point	middle of the lens, model dals the pinhole location of a pinhole camera
RED	Red light
RGB	red, green and blue
SfM	Structure from motion
SIFT	Scale invariant feature transform
Stereo pair	two cameras with known orientation difference used for stereography
Texture Atlas	image file in which is used to texture a 3D model
Tiepoint	Used keypoint for camera calibration

List of Figures

Note paper of fieldwork



Aufnahmeformular

Datum 5.8.15	Region (Kürzel) IT-ADO	Gipfel (Kürzel) PNL	Plot (Kürzel) E45	Bearbeiter JK
Uhrzeit Start erstes Foto (local time) + Fotonummer 13:35 2069	Uhrzeit Ende letztes Foto (lokal time) + Fotonummer 2150	Kameramodell Tetra		
		Objektiv		
		Brennweite		
		Blendenzahl (>15)		
		ISO		
		Autofokus		
Wetterbeschreibung sunny		Weißabgleich		
Fotonummer erstes Foto Übersichtsfotos von Plot		Kommentare Possible errors due some minor rockfall into the centre of the plot dome structures at edges		
Fotonummer erstes Foto von den Seiten vom gesamten Plot				
Fotonummer erstes Foto von den Fotos vom inneren Quadrat				
Fotonummer erstes Foto Würfel				
Fotonummer erstes Foto Nadirfotos				

Matlab scripts

Calculate amount of pictures:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Jurjen Kamphuis                           %
% script to calculate amount of pictures    %
% at certain overlap at certain height      %
%                                           %
% modified: 19-08-2015                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear
clc
close all

%% divine camera height
height = 2                                  % heigt of camera in meter
photo = [height*0.82 height*0.61];          %size of the resulting picture

%% divine overlap and plot size
sizeplot = 3;                               %size of plot
overlapshort = 0.6;                             %overlap in first direction
overlaplong = 0.4;                             %overlap in second direction

%% calculation of photos
shortedge = (sizeplot+photo(2))/(photo(2)*(1-overlapshort))      %number of photos in
a line from one edge to its opposite with half a photo overlap at each side
longedge = (sizeplot+photo(1))/(photo(1)*(1-overlaplong))     %number of lines to
cover the plot

numberofphotos = ceil(shortedge)*ceil(longedge)         %number of photos
```

Classification tool: (small lettertype for un important stuff)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Jurjen Kamphuis                           %
% script to to make a tool for              %
% classification of orthophotos and         %
% texture files                             %
%                                           %
% modified: 28-08-2015                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function varargout = classifier(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @classifier_OpeningFcn, ...
                   'gui_OutputFcn',  @classifier_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
```

```
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before classifier is made visible.
function classifier_OpeningFcn(hObject, eventdata, handles, varargin)
% Choose default command line output for classifier
handles.output = hObject;

%add extra handles
handles.orthophoto = 0;
handles.transparency = 0;
handles.texture = 0;
handles.trans_texture = 0;
handles.class.xyz = false;
handles.class.rad = false;
handles.class.NDVI = false;
handles.ratio = 1;
handles.list = '';
handles.filename.orthophoto = '';
handles.filename.texture = '';
handles.cutoff = 0.25;
% Update handles structure
guidata(hObject, handles);



% --- Outputs from this function are returned to the command line.
function varargout = classifier_OutputFcn(hObject, eventdata, handles)
% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

[filename, pathname]=uigetfile('.tif')
Path=fullfile(pathname,filename)
handles.filename.orthophoto = filename;
orthophoto = importdata(Path);
handles.orthophoto = orthophoto(:,:,1:3);
handles.transparency = orthophoto(:,:,4);
set(handles.feedback, 'String', filename)
axes(handles.preview);

image(handles.orthophoto)
axis off
% guidata
guidata(hObject, handles)



% --- Executes on button press in calibtrate.
function calibtrate_Callback(hObject, eventdata, handles)
% hObject    handle to calibtrate (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[filename, pathname]=uigetfile('.tif')
Path=fullfile(pathname,filename)
calibrate_image = importdata(Path);
[n,m] = size(calibrate_image(:,:,1));
```

```
%mid sample (highest change of right calibration
 calibrate_image = calibrate_image(:,:,1:3);
%(floor(0.25*n):floor(0.75*n),floor(0.25*m):floor(0.75*m),1:3);
 axes(handles.preview);

 image(calibrate_image)
 NIR = sum(sum(calibrate_image(:,:,1)));
 RED = sum(sum(calibrate_image(:,:,2)));
 GRN = sum(sum(calibrate_image(:,:,3)));

 %amount of visible light
 VIS = RED/2 + GRN/2;

 %ratio between NIR and GRN
 RATIORED = num2str(2-RED/NIR);
 RATIOGRN = num2str(2-GRN/NIR);
 %set values
 set(handles.NIR, 'String', RATIORED);
 set(handles.GRN, 'String', RATIOGRN);

  handles.list = [RATIORED, ':' ,RATIOGRN, char(10), handles.list];


 set(handles.listbox2, 'String', handles.list)

 handles.list


 handles.ratio = VIS/NIR;
 guidata(hObject, handles);

% --- Executes on button press in classifybutton.
function classifybutton_Callback(hObject, eventdata, handles)

%prepare data
orthophoto = im2double(handles.orthophoto);
transparency = im2double(handles.transparency);
texture = im2double(handles.texture);
trans_texture = im2double(handles.trans_texture);

%all pixels which should not be classyfied are [0,0,0]
orthophoto(:,:,1) = (orthophoto(:,:,1)+0.0001) .* transparency ;
orthophoto(:,:,2) = (orthophoto(:,:,2)+0.0001) .* transparency ;
orthophoto(:,:,3) = (orthophoto(:,:,3)+0.0001) .* transparency ;

texture(:,:,1) = (texture(:,:,1)+0.0001) .* trans_texture ;
texture(:,:,2) = (texture(:,:,2)+0.0001) .* trans_texture ;
texture(:,:,3) = (texture(:,:,3)+0.0001) .* trans_texture ;

%% seperating bands into vectors
[R,C] = size(orthophoto(:,:,1));
[NIR_1, NIR_2 NIR] = find(double(reshape(orthophoto(:,:,1),R*C,1)));
[RED_1, RED_2 RED] = find(double(reshape(orthophoto(:,:,2),R*C,1)));
[GRN_1, GRN_2 GRN] = find(double(reshape(orthophoto(:,:,3),R*C,1)));

LENGTH = numel(double(reshape(orthophoto(:,:,1),R*C,1)));

[R_T,C_T] = size(texture(:,:,1));
[NIR_1_T, NIR_2_T NIR_T] = find(double(reshape(texture(:,:,1),R_T*C_T,1)));
```

```
[RED_1, RED_2 RED_T] = find(double(reshape(texture(:,:,2),R_T*C_T,1)));
[GRN_1, GRN_2 GRN_T] = find(double(reshape(texture(:,:,3),R_T*C_T,1)));

LENGTH_T = numel(double(reshape(texture(:,:,1),R_T*C_T,1)));

%only execute when marked
if handles.class.rad == true
    handles.list = ['rad', char(10), handles.list];
    set(handles.listbox2, 'String', handles.list);

%k-means on manual radial separation by converting to sperical coordinates
[azimuth,elevation,r] = cart2sph(NIR,RED,GRN);
[idx] = kmeans([azimuth elevation],2);

figure
%subplot of orthophoto
subplot(2,2,1)
imagesc(handles.orthophoto)
subplot(2,2,4)

%subplot of clusterd pixels
subplot(2,2,2)
    plot3(NIR(idx==1),RED(idx==1),GRN(idx==1),'b.', ...
              NIR(idx==2),RED(idx==2),GRN(idx==2),'b.')
    xlabel('NIR')
    ylabel('RED')
    zlabel('GRN')

%subplot of classified image
subplot(2,2,3)
    A = full(sparse(NIR_1,NIR_2,idx-0.5,LENGTH,1));

    classified_image = reshape(A,R,C);
    imagesc(classified_image)
    classified_image(:,:,2)=classified_image;
    classified_image(:,:,3)=zeros(R,C);
    imwrite(classified_image,'classified_rad.jpg','JPEG')


end


%only executes when marked
if handles.class.xyz == true
    handles.list = ['xyz', char(10), handles.list];
    set(handles.listbox2, 'String', handles.list);



%k-means on Euclidian distance

[idx] = kmeans([NIR RED GRN],2);

figure
%subplot of orthophoto
subplot(2,2,1)
imagesc(handles.transparency)
title('texture atlas')
subplot(2,2,4)
```

```
%subplot of clusterd pixels
subplot(2,2,2)
title('xyz nearest neighbour')
    plot3(NIR(idx==1),RED(idx==1),GRN(idx==1),'r.', ...
              NIR(idx==2),RED(idx==2),GRN(idx==2),'b.')
    xlabel('NIR')
    ylabel('RED')
    zlabel('GRN')

%subplot of classified image
subplot(2,2,3)
    A = full(sparse(NIR_1,NIR_2,idx-0.5,LENGTH,1));

    classified_image = reshape(A,R,C);
    imagesc(classified_image)
    classified_image(:,:,2)=classified_image;
    classified_image(:,:,3)=zeros(R,C);
    imwrite(classified_image,'classified_xyz.jpg','JPEG')



end

%only execute when marked
if handles.class.NDVI == true
    handles.list = ['NDVI', char(10), handles.list];
    set(handles.listbox2, 'String', handles.list);

NDVI = (NIR-((RED+GRN)*.5))./ (NIR+((RED+GRN)*.5));
cut = handles.cutoff;

NDVI_T  = (NIR_T-((RED_T+GRN_T)*.5))./ (NIR_T+((RED_T+GRN_T)*.5));

figure

%subplot of orthophoto
subplot(2,4,1)
imagesc(handles.transparency)
title('Orthophoto mask')

%subplot of texture
subplot(2,4,5)
imagesc(handles.trans_texture)
title('Texture Atlas mask')

%subplot of clusterd pixels
subplot(2,4,2)

    plot3(NIR(NDVI>=cut),RED(NDVI>=cut),GRN(NDVI>=cut),'b.', ...
              NIR(NDVI<=cut),RED(NDVI<=cut),GRN(NDVI<=cut),'b.')
    title(strcat('pixel distribution with cutoff at . ',num2str(cut)))
    xlabel('NIR')
    ylabel('RED')
    zlabel('GRN')

 subplot(2,4,6)

    plot3(NIR_T(NDVI_T>=cut),RED_T(NDVI_T>=cut),GRN_T(NDVI_T>=cut),'g.', ...
              NIR_T(NDVI_T<=cut),RED_T(NDVI_T<=cut),GRN_T(NDVI_T<=cut),'b.')
    title(strcat('pixel distribution with cutoff at .',num2str(cut)))
    xlabel('NIR')
    ylabel('RED')
    zlabel('GRN')


%subplot and making of classified image
subplot(2,4,3)
NDVIarc = NDVI;
NDVI(NDVI>=cut) = 1;
NDVI(NDVI<=cut) = 0.5;

A = full(sparse(NIR_1,NIR_2,NDVI,LENGTH,1));

    classified_image = reshape(A,R,C);
    imagesc(classified_image)
    title('Classified image')
    classified_image(:,:,2)=classified_image;
    classified_image(:,:,3)=zeros(R,C);
    classified_image(:,:,1)=zeros(R,C);
    filename = strcat(handles.filename.orthophoto,'_NDVI.jpg');
    imwrite(classified_image,filename,'JPEG')

    subplot(2,4,7)
NDVI_Tarc = NDVI_T;
NDVI_T(NDVI_T>=cut) = 1;
NDVI_T(NDVI_T<=cut) = 0.5;

A_T = full(sparse(NIR_1_T,NIR_2_T,NDVI_T,LENGTH_T,1));

    classified_texture = reshape(A_T,R_T,C_T);
    imagesc(classified_texture)
    title('Classified texture')
    classified_texture(:,:,2)=classified_texture;
    classified_texture(:,:,3)=zeros(R_T,C_T);
    classified_texture(:,:,1)=zeros(R_T,C_T);
    filename = strcat(handles.filename.texture,'.jpg');
    imwrite(classified_texture,filename,'JPEG')

%classified statistics
subplot(2,4,4)
title('Surface Ratio vegetation/rock')
    tekst = num2str(numel(NDVI(NDVI==1))/numel(NDVI(NDVI==0.5)))
    axis off
    text(0.2,0.8,strcat('orthophoto -> ',tekst));
    tekst_T = num2str(numel(NDVI_T(NDVI_T==1))/numel(NDVI_T(NDVI_T==0.5)))

    text(0.2,0.6,strcat('texture ->',tekst_T));

%stability of cutoff
subplot(2,4,8)
    title('vegetation/rock ratio as a function of the cut off')
    legend('red = orthophoto','green = texture')
    xlabel('cutoff')
    ylabel('Vegetation/rock area')
    stability(3) =
numel(NDVIarc(NDVIarc>(cut+0.05)))/numel(NDVIarc(NDVIarc<(cut+0.05)));
    stability(2) = numel(NDVI(NDVI==1))/numel(NDVI(NDVI==0.5));
    stability(1) = numel(NDVIarc(NDVIarc>(cut-0.05)))/numel(NDVIarc(NDVIarc<(cut-
0.05)));
```

```
    stability_T(3) =
numel(NDVI_Tarc(NDVI_Tarc>(cut+0.05)))/numel(NDVI_Tarc(NDVI_Tarc<(cut+0.05)));
    stability_T(2) = numel(NDVI_T(NDVI_T==1))/numel(NDVI_T(NDVI_T==0.5));
    stability_T(1) = numel(NDVI_Tarc(NDVI_Tarc>(cut-
0.05)))/numel(NDVI_Tarc(NDVI_Tarc<(cut-0.05)));
    hold on
    plot([cut-0.05 cut cut+0.05],stability,'r')
    plot([cut-0.05 cut cut+0.05],stability_T,'g')
    hold off
end



% --- Executes during object creation, after setting all properties.
function feedback_CreateFcn(hObject, eventdata, handles)
% hObject    handle to feedback (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function feedback2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to feedback (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in rad.
function rad_Callback(hObject, eventdata, handles)
%assign logical
handles.class.rad = get(hObject,'Value');

%update gui data
guidata(hObject, handles);


% --- Executes on button press in xyz.
function xyz_Callback(hObject, eventdata, handles)

%assign logical
handles.class.xyz = get(hObject,'Value');

%update gui data
guidata(hObject, handles);

% --- Executes on button press in NDVI.
function NDVI_Callback(hObject, eventdata, handles)
%assign logical
handles.class.NDVI = get(hObject,'Value');

%update gui data
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function preview_CreateFcn(hObject, eventdata, handles)
% hObject    handle to preview (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
axis off
% Hint: place code in OpeningFcn to populate preview

% --- Executes during object creation, after setting all properties.
function NIR_CreateFcn(hObject, eventdata, handles)
% hObject    handle to NIR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function GRN_CreateFcn(hObject, eventdata, handles)
% hObject    handle to GRN (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in texturebutton.
function texturebutton_Callback(hObject, eventdata, handles)
% hObject    handle to texturebutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[filename, pathname]=uigetfile('.tif')
Path=fullfile(pathname,filename)
handles.filename.texture = filename;
texture = importdata(Path);
handles.texture = texture(:,:,1:3);
handles.trans_texture = texture(:,:,4);
set(handles.feedback2, 'String', filename)
axes(handles.preview2);

image(handles.texture)
axis off
% guidata
guidata(hObject, handles)

% --- Executes on slider movement.
function slider1_Callback(hObject, eventdata, handles)
handles.cutoff = get(hObject,'Value')
guidata(hObject, handles)
 %       get(hObject,'Min') and get(hObject,'Max') to determine range of slider


% --- Executes during object creation, after setting all properties.
function slider1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end
```