TECHNISCHE UNIVERSITÄT WIEN

AUTOMATION & CONTROL INSTITUTE
INSTITUT FÜR AUTOMATISIERUNGS- & REGELUNGSTECHNIK

Soil Cover Estimation in the Field of Agriculture Using Color and Near Infrared Information

MASTER THESIS

Conducted in partial fulfillment of the requirements for the degree of a

Diplom-Ingenieur (Dipl.-Ing.)

supervised by

Ao.Univ.-Prof. Dipl.-Ing. Dr. techn. M. Vincze
Dipl.-Ing. Dr. techn. J. Prankl

submitted at the

TU Wien

Faculty of Electrical Engineering and Information Technology
Automation and Control Institute

by

David Martin
Winkel 8
6741 Raggal
Austria

Vienna, 29th June, 2019

Vision for Robotics Group
A-1040 Wien, Gusshausstr. 27, Internet: http://www.acin.tuwien.ac.at

Preamble

First of all, I would like to thanks Dr. Johann Prankl for supporting me with his knowledge and experience throughout the whole project.
I would also like to sincerely thank Professor Dr. Markus Vincze for supervising this thesis and advising me whenever it was necessary.
Finally, I am very thankful for the support of my whole family and my friends which gave me a huge backing and accompany through the whole study.

David Martin
Vienna, 29th June, 2019

Abstract

This thesis presents a method for soil cover estimation in the field of agriculture based on semantic image segmentation under the usage of color, Near Infrared (NIR) and depth information. The main advantage of this approach against approaches using monocular camera systems is to increase the performance of the segmentation process by applying additional knowledge about green vegetation through NIR images. It is shown that the general accuracy of the segmentation process is sufficient good to estimate the soil composition for practical usage under the constraint of the rough environmental conditions and soil composition diversity. The presented method is evaluated against the monocular camera system approach by using the F1 accuracy score. Further, it is shown that the extended estimation process leads to a better F1 accuracy score of 5 for the segmentation of the residues class. Additionally, the implementation of a camera box for autonomously image capturing is presented.

Kurzzusammenfassung

In dieser Arbeit wird eine Methode zur Abschätzung der Bodenbedeckung in der Landwirtschaft vorgestellt, die auf semantischer Bildsegmentierung unter der Verwendung von Farb-, Near Infrared (NIR)- und Tiefeninformation basiert. Der Hauptvorteil dieses Ansatzes gegenüber Ansätzen mit monokularen Kamerasystemen besteht darin, die Genauigkeit des Segmentierungsprozesses zu steigern, indem zusätzliches Wissen über die grüne Vegetation durch die NIR-Bilder eingesetzt wird. Es wird gezeigt, dass die allgemeine Genauigkeit des Segmentierungsprozesses ausreicht, um die Bodenzusammensetzung für den praktischen Gebrauch, unter rauen Umgebungsbedingungen und der vielfältigen Zusammensetzung des Bodens, abzuschätzen. Die vorgestellte Methode wird mit Hilfe des F1-Genauigkeitsfaktors gegen den Ansatz des monokular Kamerasystems verglichen. Darüber hinaus kann gezeigt werden, dass der erweiterte Ansatz zu einem besseren F1-Genauigkeitswert von 5 für die Segmentierung der Rückstandsklasse führt. Zusätzlich wird die Implementierung einer Kamerabox zur autonomen Bilderfassung vorgestellt.

Contents

List of Figures

List of Tables

Acronyms

SVM	Support Vector Machine
NN	Neural Network
MRF	Markov Random Field
RF	Random Forest
RT	Random Tree
FCN	Fully Convolutional Network
RCNN	Regional Convolutional Neural Network
CNN	Convolutional Neural Network
DT	Decision Tree
NIR	Near Infrared
NDVI	Normalized Difference Vegetation Index
RGB	Red, Green and Blue
NDWI	Normalized Difference Water Index
EMR	Electromagnetic Radiation
HSI	Hyperspectral Image
SWIR	Short Wave Infrared
VIS	Visible Light
DOF	Depth of Field
DLT	Direct Linear Transformation
WLS	Weighted Least Squares
OSAVI	Optimized Soil-Adjusted Vegetation Index

1 Introduction

Smart and sustained farming is getting an increasingly important topic, considering that the world population is continuously growing and the food quality is decreasing, for example, by the widespread usage of herbicides. One approach to target these problems is to develop more intelligent machines with environmental sensors to optimize agricultural production. Thus the knowledge about the agriculture environment is essential to set actions against these developments. The first step to get the knowledge is to collect information from our environment. Several types of sensors can do this. For example the knowledge can be collected trough laser scanning, infrared scanning and many other techniques. In this thesis, we focus on images as information source. Figure 1.1 shows a classification result from soil covered by living plants, residues and stones based on images captured under real environmental conditions.

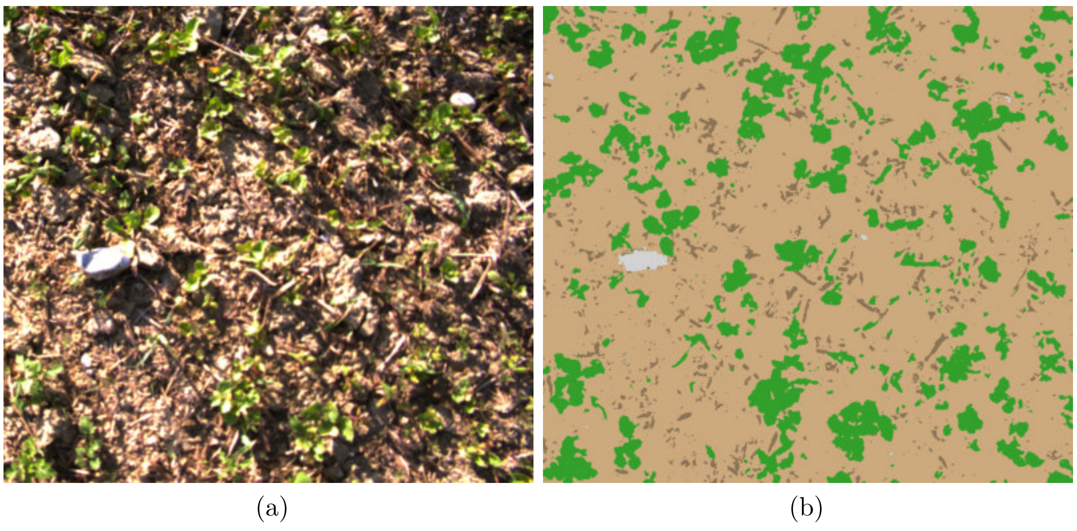


Figure 1.1: Soil covered with living plants and residues (a) and the classification result (b).

With the extracted information and the knowledge about soil coverage, it is possible to react on the current soil composition directly. Thus the main challenge is to find a way to collect this information and to build a system which can be used under rough environmental conditions. It is not an easy task to identify the possible best way to solve the soil cover estimation problems. This thesis scopes on several different approaches of semantic image segmentation as also on the hardware implementation of such a system. In the following two sections, a more detailed problem description is given, and our proposed solution is discussed.

1.1 Problem

The goal is to get information about soil coverage. In detail this means to get the percentage value of living vegetation, residues, stones and soil. With this information, we can prevent soil from erosion and gain knowledge about the uniformity of meddling to promote rotting.
In earlier work, the segmentation process was done via a monocular camera system. To improve the classification performance, we try to extend this system with additional feature sources. Further we try to build an autonomous camera system as also a casing to capture images directly from a farming vehicle, thus it can be used for practical applications. Section 1.2 briefly describes our proposed solution.

1.2 Proposed Solution

In our approach, we try to extend the monocular camera approach by adding additional Near Infrared (NIR) and depth information to the semantic image segmentation process. Further, we pursue to build an autonomous camera box which can be installed on a farming vehicle, to capture and stream the images to a computational unit. We assume a higher quality of the results by using NIR and depth information during the classification process, based on the fact that living organic plants can be differenced from not living in the NIR light spectrum. The depth information could also be helpful because through this additional information the process chain gets the knowledge about the different depths of the objects. To capture this information, we use a three-camera setup. We are using two RGB cameras for relative depth extraction trough stereo matching and a third specialized NIR camera. To project all three images to the same spatial coordinate system a very precisely calibration of every single camera as well as a stereo calibration between the cameras is required.
The camera box is needed to capture training data and evaluation data. Autonomous because thus the box is universally applicable. Additionally, the box should be resistant against rough environmental conditions. We decided

to use a Random Forest (RF) algorithm for semantic image segmentation because we want to evaluate the benefits of the extra information in a short time of development and this method gives us the possibility to investigate the impact to the model. Collecting information is one part which has to be done. The other one is to interpret it. This is colloquially known as semantic image segmentation. Semantic image segmentation deals with many different challenges like image acquisition, image preprocessing, feature extraction and classification as it is shown in Figure 1.2.

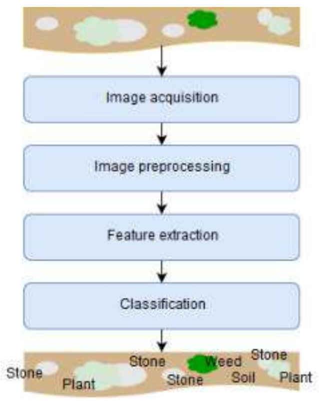


Figure 1.2: Semantic image segmentation process

The main challenge is to handle all these tasks to get accurate segmentation results based on the improved camera set up, which brings additional challenges in terms of image projection and comprehensive disparity estimation. The structure of this work is briefly described in Section 1.3.

1.3 Outline of Thesis

After a brief introduction which is given in Chapter 1, an overview of the related work in Section 2, including semantic image segmentation methods, image processing in agriculture and NIR, is given. The Chapter 3 mainly describes our approach by including the camera calibration, stereo matching and the projection of the images. It also includes the particularities of the algorithm implementation like the chosen features and layers and a brief description of the camera box implementation as also of the written software. Further, the made experiments and the evaluation of our method is discussed in Chapter 4. Chapter 5 summarizes the achieved results and gives a short outlook to improvements which can be done in future work. At the end of the thesis in Section 6, the training and evaluation dataset is listed.

2 Related Work

The basic problem underlying this work is to get information about the surface texture from agricultural images. In order to recognize the different components of an agricultural surface a semantic segmentation has to be carried out. There are many approaches to get semantic information from data sets. It is very actual and frequently researched field. To handle the problem statement, there is no way around semantic segmentation. The following Section 2.1 will discuss semantic segmentation.

2.1 Semantic Segmentation

Semantic segmentation is the base problem which has to be solved to get the information hold by features of a data set. In this work, the focus is primarily on semantic segmentation based on image data.
Before discussing the different approaches to reach this goal an overview of semantic image segmentation is given. Principal objects or components in images are formed out of linked pixels. In Figure 2.1 simple objects are shown, which easily can be recognized and clustered based on their colour. Thus all blue, orange, yellow and red pixels linked independently together are forming different objects. Clustering these objects is very simple based on the colour

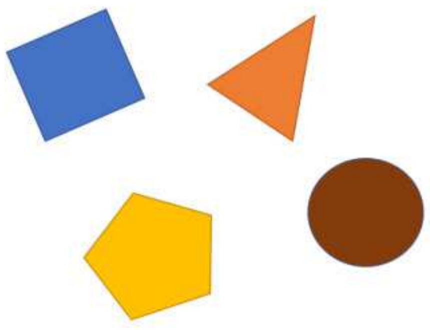


Figure 2.1: Simple objects in an image

feature. However, after the clustering, we still do not know what objects the individual pixel-groups are symbolizing. Giving clusters a meaning is the second task which has to be done. In a simple assumption, we could say that all blue pixel-groups are symbolizing rectangles and all orange pixel-groups are symbolizing triangles and so on. But this assignment of meaning would only hold in a world where the colour blue is reserved for rectangles the colour orange is reserved for triangles and so on. Because it is not true, we need better assumptions and methods to give pixel-groups a meaning. One approach would be to use additional features in addition to colour. For example, edges are compelling features. They can be identified with the help of partial derivatives of the image like it is done in canny edge detection algorithm [1]. When we have the information about edges in an image, we could say that four edges which have a 90-degree angle between each other and forming a closed loop are symbolizing a rectangle. Thus knowing about edges helps much to identify objects in an image and consequently give the image meaning. This example shows the importance of features in semantic segmentation. An image holds much more information than only colours. Features can hold information about segmentation and semantic. Every approach to solve semantic segmentation is only as good as the chosen features.
There are many different approaches to solve the semantic segmentation problem. Every approach shows another way to handle and compute the information of the features thus an image can be separated into differed known objects. A basic separation of semantic segmentation approaches can be done, by grouping them into traditional methods (2.1.1), methods using neural networks (2.1.2) and approaches based on RFs (2.1.3). In the following sections, these approaches will be discussed.

2.1.1 Traditional Methods

Before NNs or RFs came up, feature and classification methods for semantic image separation were the most important ones [2]. In the following sections, three heavily used traditional methods are discussed, and their faults and benefits will be highlighted.

K-means Clustering

K-means clustering is an unsupervised method for semantic segmentation. In the first step, a fixed defined set of k cluster centroids are randomly placed in data space. In a simple form, the image space can be spanned by the RGB colour space. The algorithm assigns each data point to one of the k clusters by calculating the distance between each data point and each centroid point of a cluster. Every data point is assigned to the nearest centroid. After the assignment, the k centroids points are repositioned by calculating the average position of all data points which are assigned to the related cluster. These steps are repeated until the centroid point does not change any more or a threshold is reached [3]. The Figure 2.2 shows the basic k-means algorithm.

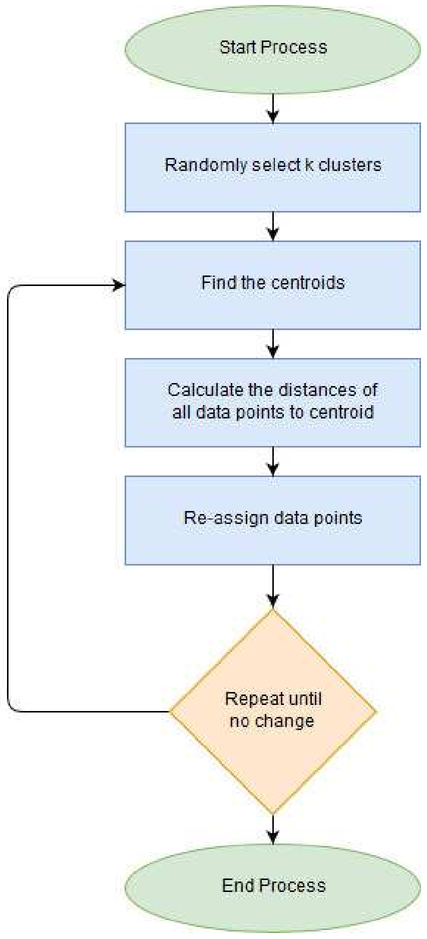


Figure 2.2: K-means algorithm [3]

The main disadvantage of the k-means algorithm is that the number of clusters has to be defined before ahead of the computation. This limitation is very significant because the algorithm cannot define the number of clusters by its own, and thus it is not very robust against different images. Figure 2.3a shows a simple image with four objects. With this information, it is no problem to segment the image into five clusters what is shown in 2.3c. But if the algorithm gets the wrong information about cluster number, the result is not the one we expect, like it is shown in 2.3b. In 2.3b the algorithm was initialized with two clusters. Three objects were separated together in one cluster, and the pentagon was assigned to the background cluster [4].

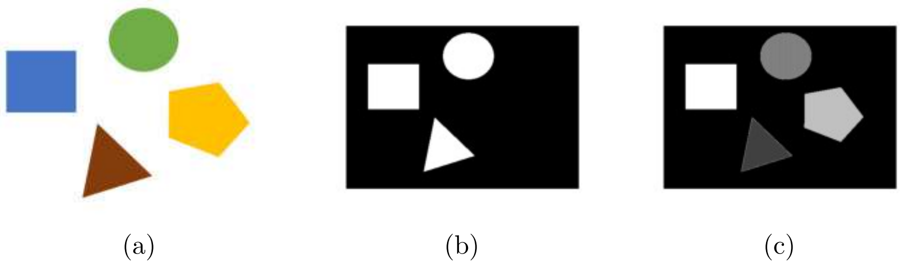


Figure 2.3: K-means clustering with different k-values

Support Vector Machine (SVM)

SVMs are supervised and widely used classifiers. They solve the classification problem by separating the data points with hyperplanes. In the set of training data each data point is assigned to one of two categories. A data point can be viewed as an p-dimensional vector and the task is to find a $p-1$-dimensional hyperplane which separate all data point related to their category.
A training data set can be represented as tuple (x_i, y_i) where x_i is the feature vector and $y_i \in \{-1,1\}$ is the binary label vector. A hyperplane can be written as set of points which satisfy the equation $w \bullet x + b = 0$. Wherein w is the normal vector to the hyperplane and b is the bias vector like it is shown in figure 2.4. In 2D space the axis can be annotated with feature 1 and feature 2 of one data point x_i [4].

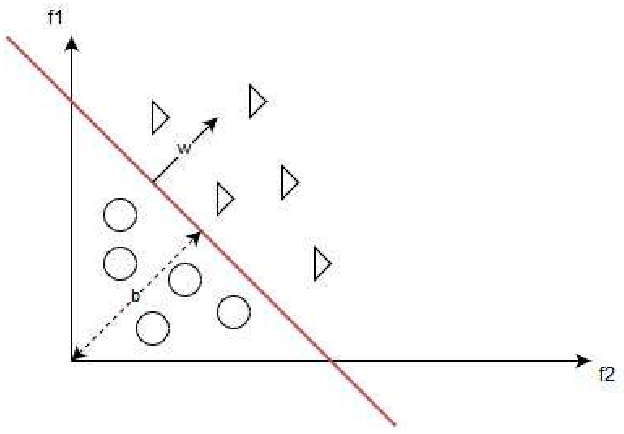


Figure 2.4: Data points in 2D space desperate by a hyperplane

For all data points above the hyperplane, the expression $w \bullet x + b$ has a positive sign what correspondents to label 1 and for all points, below the hyperplane, the label is -1 because the expression has a negative sign. The challenge is to find a suitable hyperplane with the help of the training data and the corresponding ground truth. This can be done by solving the objective function

$$min \frac{1}{2} * \|w\|^2 \tag{2.1}$$

subject to

$$y_i(w \bullet x_i + b) \geq 1, \forall x_i \tag{2.2}$$

A solution can be found with the help of Lagrange multipliers like it is explained in the paper of Ton Wen and Alan Edelman [5].
Probably many of these hyperplanes which separate the data points exist. One choice is to choose the hyperplane, which represents the maximum separation. To get this hyperplane, the distance from the nearest data points on both sides should be maximized. Such a hyperplane is known as maximum-margin-hyperplane. For this approach to works, data points must by linear separable. If they are not, we would need to choose a non-linear separation function, or we could project the data points to a higher dimensional space where they are linearly separable. The second attempt is called kernel trick. Thus SVMs are not limited by linear separable datasets. Another limitation could be the number of categories. This limitation can be handled with multi class SVMs described in the paper [6]. This approach has the same problem with the semantic relationship to segmentation as described in the previous Section 2.1.1 because the semantic meaning of each segmentation class is not given. In a simple form, this can be done by adoption that every segmentation section stands for a specific object.

Markov Random Field (MRF)

Image segmentation according to MRFs is based on probabilities. Herein a probability measure of all possible segmentations of the set Ω is done and the one with the highest probability is taken. A Image can be described as set $f = \{f_s : s \in S\}$ where f_s is the feature vector of a pixel s. Additionally to that a label set $\omega = \{\omega_s : s \in S\}$, which describes a single labelling possibility for image, is defined. ω_s can accept values from a label value set $l = \{1,2,3,...,n\}$. The first thing we have to do is evaluate how any occurrence of ω_s fits to f. This is expressed by the image model $P(f|\omega)$ where P is the conditional probability that f occurs under the condition that ω is true. Secondary a possibility distribution $P(\omega)$ is defined where ω is constrained by properties regardless to the image [7]. $P(\omega)$ describes the possibility that ω satisfy the supposed properties. The post posterior probability is given according to the Bayes-Therom by

$$P(\omega|f) = \frac{P(f|\omega) * P(w)}{P(f)} \tag{2.3}$$

Based on the fact that $P(f)$ is constant the posterior probability can be simplified to

$$P(\omega|f) \propto P(f|\omega) * P(w) \tag{2.4}$$

The goal is to find the maximum posterior probability

$$\widehat{w} = \arg\max_{w \in \Omega} P(f|\omega) * P(w) \tag{2.5}$$

Thus $P(\omega)$ and $P(f|\omega)$ need to be described to find the maximum. Under the assumption that ω follows a Gibbs distribution $P(\omega)$ can be formalized with help of the energy function $U(\omega)$ [7], as follows

$$P(\omega) = \frac{1}{Z}\exp(-U(\omega)) = \frac{1}{Z}\exp(-\sum_{c \in C} V_c(\omega_c)) \tag{2.6}$$

where Z is an normalization constant over all possible labellings related to the cliques. Cliques are associated pixel groups like it is shown in Figure 2.5.

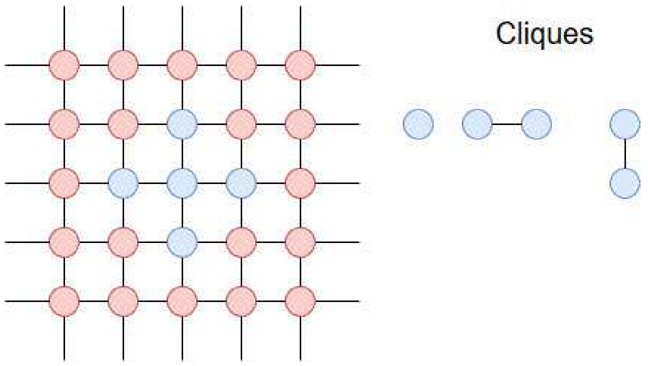


Figure 2.5: First order neighbourhood system [7]

The set of all possible cliques c is named C.With the function

$$V_c = \delta(\omega_s,\omega_r) = \begin{cases} +1, \omega_s \neq \omega_r \\ -1, otherwise \end{cases} \tag{2.7}$$

the potential of a clique can be measured. The cliques, shown in Figure 2.5 are just defined over a first order neighbourhood. They could also be more complex. $P(f|\omega)$ can be found under the assumption that $P(\omega_s|f_s)$ follows a normal distribution, the likelihood is formalized as

$$P(f_s|\omega_s) = \frac{1}{\sqrt{(2\pi)^n}} \tag{2.8}$$

and $P(f|\omega)$ is the product of all possibilities where $s \in S$.

$$P(f|\omega) = \prod_{s\in S} P(\omega_s|f_s) \tag{2.9}$$

Maximizing the full equation can be done by simplifying and solving the function like it is described in Section 3.3 and 4 in the paper of Zoltan Kato and Ting-Chuen Pong [7].
The method after MRFs is not restricted by semantic like the approaches discussed above because the set of labels can be defined freely, and the algorithm can classify a subset of these labels to the images.

2.1.2 Methods using Neural Networks (NNs)

NNs are heavily used in semantic data interpretation. Thus they are not restricted to semantic image segmentation and can be used to solve common semantic interpretation problems. Before we go deeper into the image specific approaches, a basic understanding of NNs is important which is given in the following section.
NNs can be viewed as a net of computation nodes where each node has several inputs and one output. Connecting these nodes forms a complex computation network which brings the effort to emulate what we know about biological neuronal networks. NNs are separated into minimum three layers. The input layer, which represents the input features, the multiple hidden layers are representing the core calculation network and the output layer representing the predicted semantic interpretation [8]. In Figure 2.6 the basic construction of a NN and the layer segmentation is shown. A single node of any layer is called neuron and is designed similar to a neuron in the human brain. A neuron takes several weighted inputs which are analogue to biological synapses. The inputs are committed to a transfer function where the net input is calculated. This is done by summing all the weighted inputs. After the net input calculation, an activation function like a sigmoid or tanh calculates the output which is used for the next hidden layer or if it is the last hidden layer directly for the output. To get a fully connected network, each node output is connected to each node of the next layer. Important is that not all layers must have the same number of nodes. There are also other types of NNs where intermediate

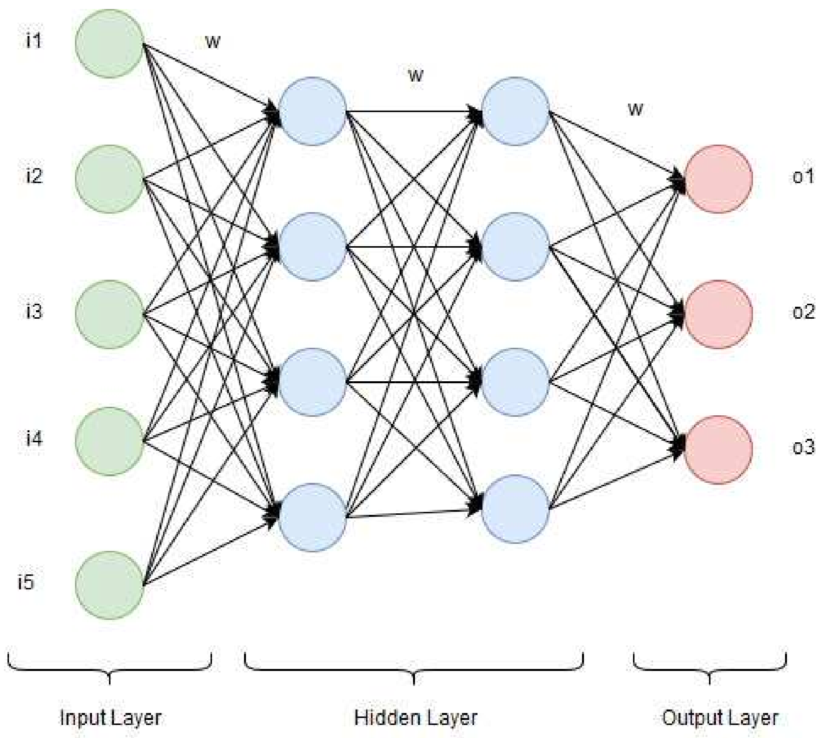


Figure 2.6: Basic neuronal network

node connection is handled differently.
Equation 2.10 shows the principal output calculation of a neural network with one hidden layer[8].

$$o_k = g[\sum_{j=0}^{N_j} w_{kj} g(\sum_{n=0}^{N_n} w_{jn} i_n)] \tag{2.10}$$

Where g is for example the sigmoid activation function

$$g(x) = \frac{1}{1 + e^{-x}} \tag{2.11}$$

or the tanh activation function

$$g(x) = tanh(x) \tag{2.12}$$

Determining the output is straight forward when all the parameters are known. The determination of the parameter is the real challenge. This is done by training the NN with ground truths. In the first round of training, the parameters, especially the weights, are chosen randomly. Then the input of the first ground truth is applied to the network, and the outputs are calculated. Comparing the outputs to the anticipated result gives an error which has to be minimized by backward correction of the parameters. During the training, the loop of forward estimation, error determination and backward correction is executed several times until the error is in an acceptable range. All that is not merely done for one input data but also for several input data sets to get a robust and commonly usable network in terms of semantic interpretation. [8]

Convolutional Neural Network (CNN)

CNNs are special forms of neural networks with some restrictions and simplifications that they can be used for semantic image segmentation in an acceptable way in terms of calculation time and complexity. In Figure 2.7 the typical process chain of a CNN is shown. The first step is to convolute the input layer with several different kernels to get the image features. In the next step, the activation function is executed on the convolution layers. Through the polling layers downsizing the convolution layers reduce their computational complexity. The steps from convolution to polling can be repeated multiple times. At the end of the process chain, the last layer is connected to a fully connected layer where the resulting labelling is done. CNNs are realized under 3 main adjustments, which are described in the following subsections [9].

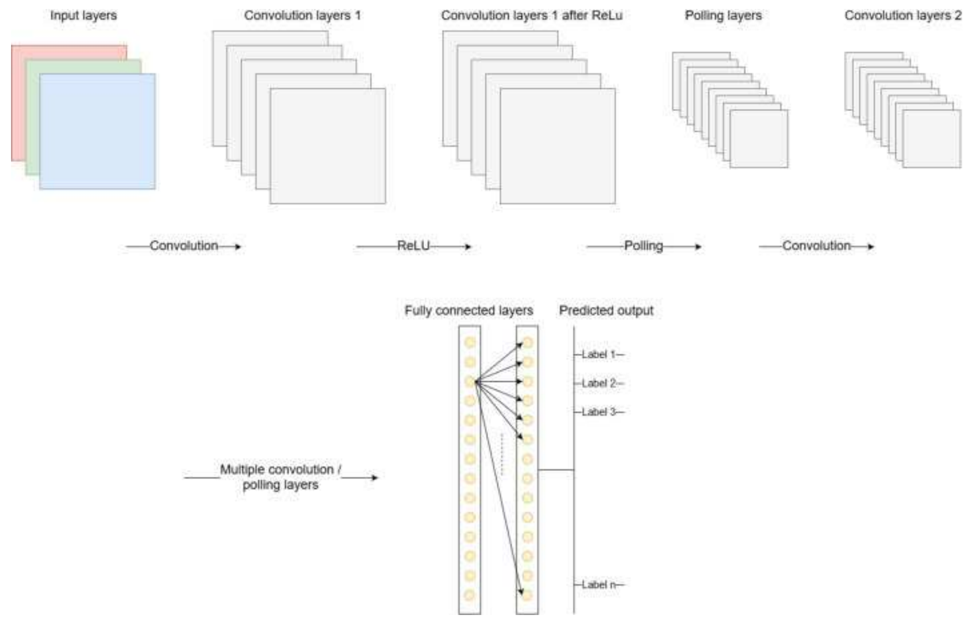


Figure 2.7: CNN

Local receptive fields

In classical NNs every input node is connected to every node of the next layer. That would mean, under the assumption of a single layer grayscale input image with a size of 800x600 pixels, that we have to connect 480.000 input nodes to every node of the next layer. As we can imagine, this would get complex, really fast and computationally intensive. To handle this problem CNNs are focused on restricted image areas with the size of the filter kernels [9].

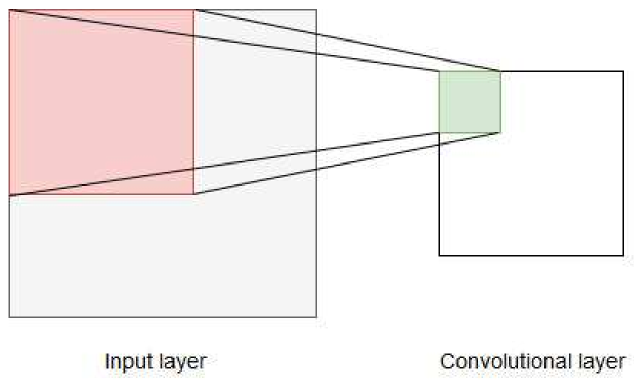


Figure 2.8: Local receptive fields

As it is shown in Figure 2.8, the result of the convolution with the filter kernel of a restricted area is only connected to one node or pixel of the convolution layer. Depending on the size of the kernel, the convolution layer accordingly gets smaller. This can be obviated by increasing the input layer dimension by half of the filter kernel with interpolated pixels. The restriction of focusing only on parts of the image can be done under the assumption that the features of an image are spatial bounded [9].

Shared weights

The second simplification is to reduce the number of parameters by keeping the local connection weights fixed for all the neurons of the next layer. Thus in comparison to a basic NN where we would have for example 76800 parameters for 3 input layer with a size of 32x32 pixels and a kernel size of 5x5 we decrease the number of parameters to 75. As we can see, this simplification decreases the complexity of processing and training of the network. As a consequence of that, the computational intensity also decreases [9].

Pooling

Pooling layers are responsible for reducing the spatial size of the convolution layers. Two frequently used types of polling are shown in Figure 2.9. Max polling returns the maximum value of the kernel focused area. It also performs a noise suppressant action by ignoring the other values of the kernel window which might manipulate the result. Average polling is another approach where the values covered by the filter kernel are averaged. Based on the noise suppressant action, we can say that the max polling approach performs a lot better [9].

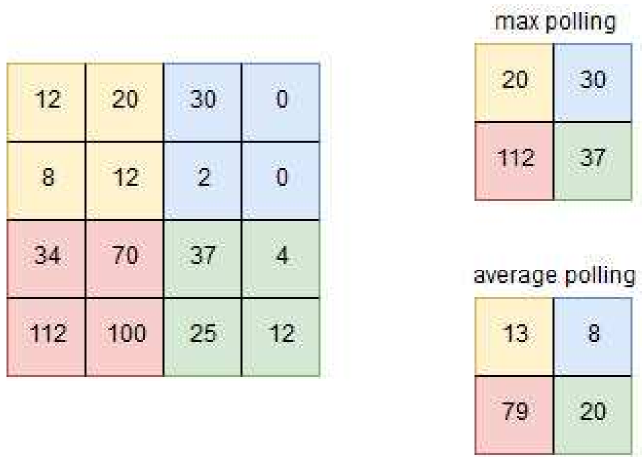


Figure 2.9: Polling types

This simplification decreases the computational intensity through the dimensionality reduction. Additional it is useful the get rid of dominant features which would influence the labelling result negative.

Basic CNNs are still computationally intensive, especially for large input images sizes, and they are also restricted to a fixed input image size. Thus they have to be reimplemented whenever the input image size is changed. Further CNNs are just detecting one object per image. If we want to detect more than one object, we have to split the input image into smaller sub-images and search in all of them. This approach would take a lot of computation time, especially when the sub-images are overlapping. In the next two sections, methods are described which try to get rid of these limitations.

Regional Convolutional Neural Network (RCNN)

The main goal of a RCNN is to detect more than one object in an image but without splitting the image into every possible sub-image segmentation. Thus this method tries to identify a smaller number of interesting regions before the CNN processing, and semantic segmentation of a single object is started. First, RCNN uses selective search to generate about 2000 region proposals, like it is shown in Figure 2.10. The selective search algorithm is described in the paper [10]. Region proposals are visualized as red rectangle bounding boxes.

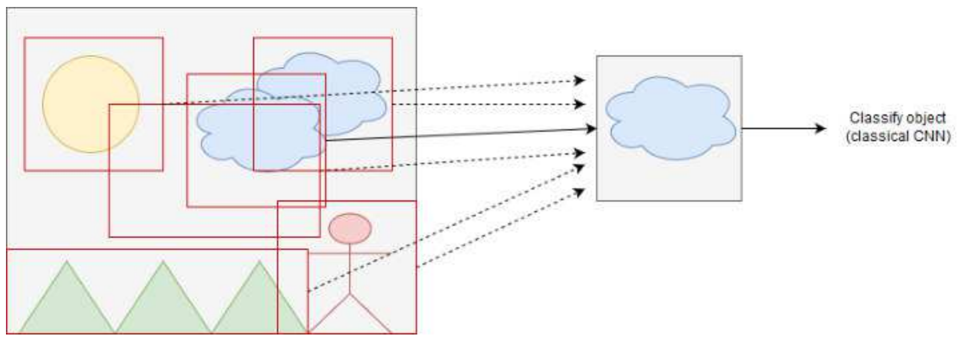


Figure 2.10: RCNN

After establishing the bounding boxes, image classification is done for each bounding box through CNNs trained for different input image sizes. With the help of regression, the detected object can be associated with the position in the input image.
This approach solves only one of the limitations we have with classical CNNs. A method for solving the struggles with changing input images sizes is described in the next section [11].

Fully Convolutional Network (FCN)

FCNs extend classical CNN by making them robust against changing input images sizes and the limitation of single object detection by per-pixel dense prediction. This is done by using only convolution and polling layers. The fully connected NN at the end of the process chain is omitted respectively replaced by additional convolution layers derived from the NN [12]. At the end of the process chain, an upsampling layer is added to reconstruct the output layer size to the corresponding input image size. The resulting process chain is shown in Figure 2.11.

Upsampling from a low resolution after the convolution layers may false the result because of the high interpolation degree. Thus using the results of preceding convolution layers improves the result. This technique is known as output fusing and is described in the paper of Jonathan Long, Evan Shelhamer and Trevor Darrell [12].
Thus this method does not decrease the computational complexity as the attempt described above, but it solves the pixel-wise label prediction for each pixel of the input image, and additionally, it also solves the sizing problem.

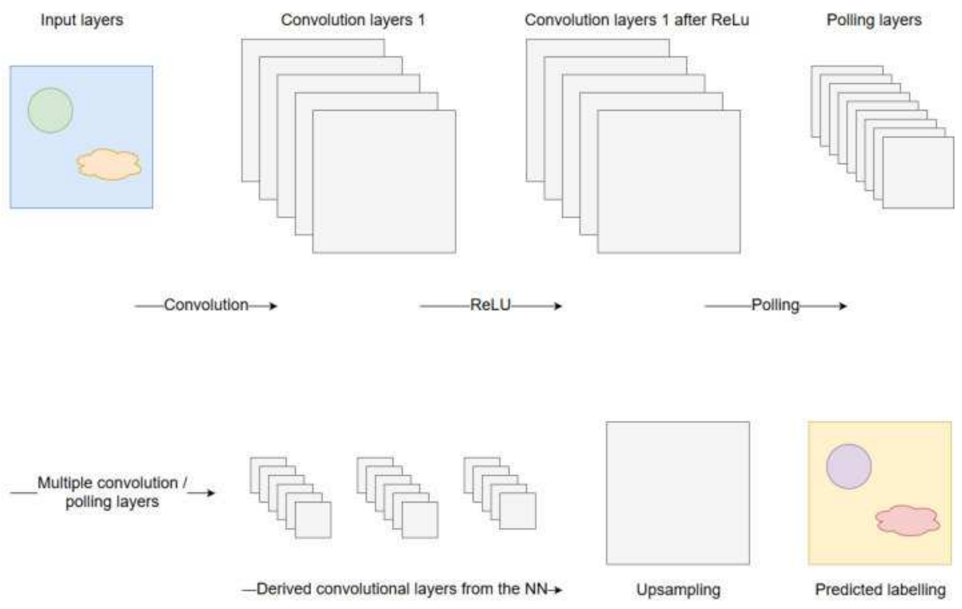

Input layers
Convolution layers 1
Convolution layers 1 after ReLu
Polling layers
Convolution
ReLU
Polling
Multiple convolution / polling layers
Derived convolutional layers from the NN
Upsampling
Predicted labelling

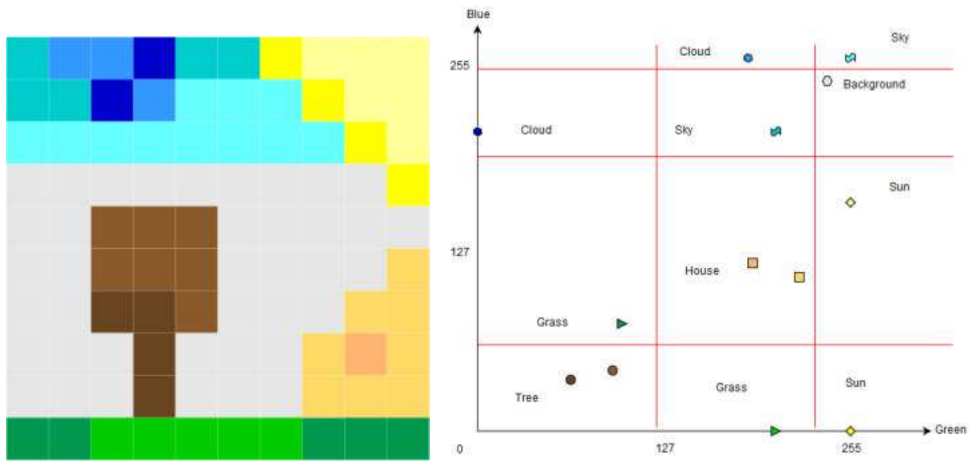
Figure 2.11: FCN

2.1.3 Random Forest (RF)

RF is a supervised classification algorithm and consists out of multiple independent random trees. A random tree can be imagined as a Decision Tree (DT) with additional adjustments. Before discussing the basic algorithm, DTs will be discussed for a good basic knowledge [13].

Decision Tree (DT)

DTs are widely used machine learning algorithms and have a typical tree structure. In this description, we will focus on image data as input data but they can also be used for more generic data sets, for example in the field of weather forecast or the domain of finances.
For simplification we assume that we have an image of the size of $10x10$ pixels as it is shown in Figure 2.12a, represented by RGB values. Further, we will use a binary trees because they are easier to train than trees with more than two child nodes. As we can see in subsection 2.1.3 we will also use binary trees for the random trees. A tree with multiple child nodes can be transformed into a binary tree. Practically, we would use more complex features than just RGB values, like edges, special filtered or channel modulated image representation to get better results [14].
To get a two-dimensional feature space, we are just using the green and blue values. The feature space is shown in Figure 2.12b.



(a) (b)

Figure 2.12: RGB nature image with the corresponding green/blue feature space

The main task of an DT is to separate the feature space into several clusters, thus that the data points can be assigned to their corresponding label. Thresholding the features does fulfil this requirement. Each node of the DT can either hold a decision where a threshold is evaluated against a feature or a resulting label as a leaf node. Thus each stage of the tree makes the semantic segmentation more granular. This evaluation is done for each pixel of the image to get an entirely labelling for every pixel. An example tree for the feature space 2.12b is shown in Figure 2.13 [14].

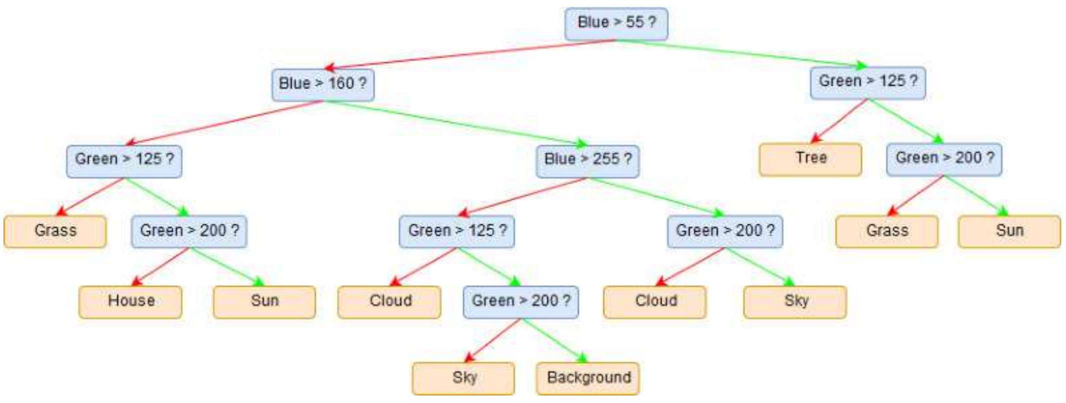


Figure 2.13: Decision Tree (DT)

The challenge is to find the best decision for each node and to decide when the error is small enough to stop branching. This can be done by train the tree with the help of ground truth, the concept of information gain and entropy. In equation 2.13 a entropy definition of a node is given.

$$E(S) = -\sum_{i}^{L} p_i \log_2(p_i) \tag{2.13}$$

Where S is the corresponding subset of training data points to the actual tree node and $p_i|i \in L$ is the fraction of data points within the label $i \in L$. L represents the set of labels. Each traning node is provided with a label i. The entropy can also be calculated in restriction to a decision or test 2.14.

$$E(T,X) = \sum_{x \in X} P(x)E(x) \tag{2.14}$$

The set X includes all the subsets $x \in X$ of the separated data points throw the decision. $P(x)$ stands for the possibility of label in subset x. Now we have two types of entropies the one just for a subset of data points in a node and the constrained one for a chosen decision. With this information and the essence that a completely homogeneous dataset has zero entropy, while a perfectly random dataset converges to maximum entropy. We can define the equation 2.15 for the information gain [14].

$$G(T,X) = E(T) - E(T{,}X) \tag{2.15}$$

Now we can calculate the information gain for every possible feature threshold combination in a specific node. The one with the highest is chosen. If the information gain of all possibilities is under a certain threshold, branching will be stopped, and the node becomes a leaf.
Two of the main problems are overfitting and the high amount of computation time, which is necessary to get the information gain for all possible feature threshold combinations. One way to prevent overfitting is to give the tree a maximum depth limitation, and a minimum number of required data points for a split. The feature threshold degeneration can be stopped, for example by a weighting function which reduces the complete set of all possibilities to a manageable subset [14].

Random Tree (RT)

A RT is special form of a DT. During training, the feature threshold possibilities are chosen randomly. Thus for every node, only a subset of features is chosen. Further, not all possible thresholds are tried. Only a fixed number of randomly chosen values is selected. With this restriction, the computational complexity is reduced dramatically, and the tree also gets robust against overfitting because

not always the most likely decision based on the training set is taken [13].
In our case, each node only has two child or leaf nodes. At the end of the training when every data point is passed through the tree, the leaf nodes are holding an empirical distribution over classes learned during the training which is matching each data point to the most likely label as it is shown in Figure 2.13. Node decisions are also known as tests.

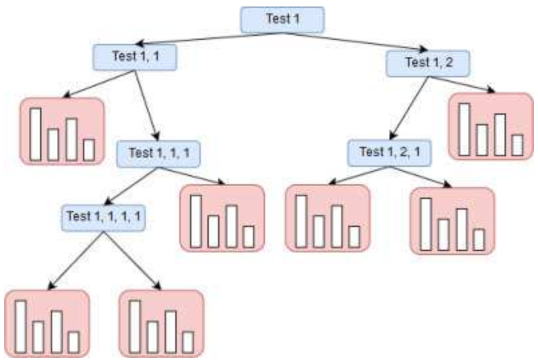


Figure 2.14: Random Tree (RT)

From RTs to a RF

Basically, a RF consist out of n RTs. An example of an RF is shown in Figure 2.15. For each tree, the features and thresholds, as also the depth restriction and the minimum amount of data points for splitting is chosen randomly. Thus it is nearly impossible that two equal trees are generated after training.
After a data point is passed through each of the n trees the data point has n labelling distributions. All these distributions are summed up to get an average labelling distribution. This way of classification is much more robust than just a single tree because every tree is unique.
If one label of the dataset is dominating the other, it could be possible that the result quality decreases. Through weighting this behaviour can be prevented. The weighting factor is proportional to inverse label occurrences.
Using just one tree makes the prediction highly sensitive to noise in the training set while the average of many trees is not if the trees are completely diverse from each other. With the randomness and a high amount of trees, this algorithm gets less sensitive against noise [14].

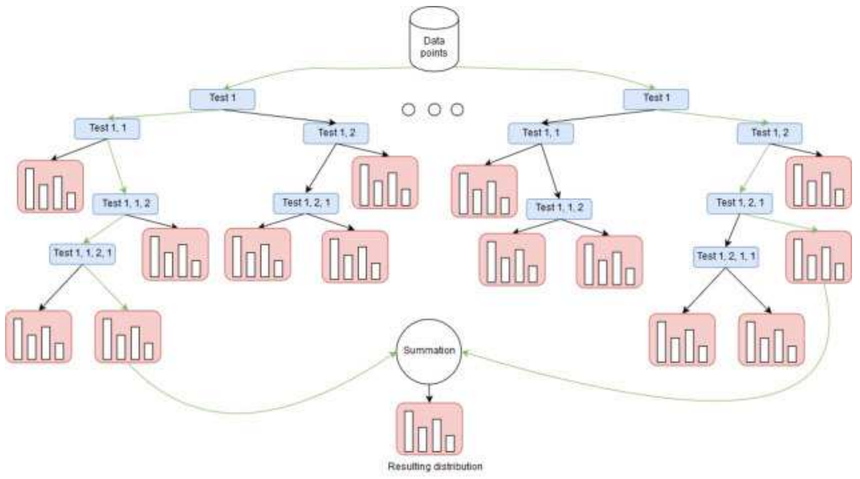


Figure 2.15: Random Forest (RF)

Hyperparameter tuning

Hyperparameters are free settable parameters for the training. Tuning these parameters increase the quality of the result. For RFs these parameters include the number of trees and features considered by each node for splitting as also the maximum tree depth and a minimum amount of data points for splitting. The parameters do not include the ones of the model itself like thresholds and the chosen features. Tuning the hyperparameters is more based on experiments than theory. One way to evaluate them is to using cross-validation. This evaluation method splits the training data set into k subsets. Now $k-1$ subsets are used for training. After training the algorithm is validated against the validation subset, which was not used for training. To get a more stable test result, this is done k times but every run with a different validation set. The whole process is executed for every set of available parameters. In the end, the parameter set with the best evaluating result is chosen to train a RF with the whole training data set. As we can imagine, this process takes much time. In practice tools automate this process. Cross-validation is just one method for hyperparameter tuning and the only one described in this paper because it is widely used [15].

2.1.4 Recapitulation

This section shortly summarizes the advantages and disadvantages of the different approaches for semantic image segmentation. All have their benefits and faults. Thus it is not easy to decide which one is the best one to use for further experiments or implementations.

K-means Clustering

This method is easy to implement and has a fast execution time. It also has not to be trained like some other approaches. The main disadvantage is that the number of classes has to be defined before ahead or a model assumption had to be made what is, in general, a difficult task because not in every image all classes are present. Initial seeding of the centroid points has a substantial impact on the performance of the algorithm also like the order of the dataset. The performance is quite good in the face of the fact this method is not trained and the dataset is good separable [4].

Support Vector Machine (SVM)

SVMs are performing very good on unknown datasets. They can separate even complex-nonlinear problems with the help of the kernel trick, but it could be tough to find a well-preforming kernel function. This method also has a good generalization and performs well against overfitting. It can scale high dimensional data also as lower dimensional data. Additionally SVMs are solved for global optima, not like NNs or RFs. The performance might be not as good as the one of NNs or RFs, but in some fields they are indispensable. The resulting model after training might be very complex and not easy to understand [4].

Markov Random Field (MRF)

Using MRF for semantic image segmentation can yield to excellent results, especially considering that the method needs no training. The method is also able to solve complex nonlinear problems. Semantic segmentation is done per-pixel. Finding the maximum can lead to an enormous challenge because it is likely to be stuck at a local optimum.[7].

Methods using Neural Network (NN)

In general, they have the best performance, but it always depends on the classification problem. They are also able to handle complex nonlinear problems very well under the usage of large training datasets. For smaller training datasets the likely convergent to local minima. Another disadvantage is that NNs use a broad set of parameters thus, they are not intuitively understandable. Further, they are used in many applications of the global internet players, and thus these methods are well proven. With the improvements served by RCNNs and FCNs, CNNs are independent against changing input image size and the execution time can be decreased. Disadvantages of NNs are that they need a very high development time and large training datasets to perform good [9] [10] [12].

Random Forest (RF)

It is a great method to use early in the development process because it is easy to build, and it is easy to see how the algorithm works. Further, the performance of the algorithm is outstanding compared to traditional methods, and it provides the possibility to handle different types of features, like numerical or nominal features. Additionally, this method can classify linear also as nonlinear datasets, which is a significant advantage compared to the traditional SVM algorithm. If the trees are correlated, this method gets sensitive to noise in the images. Thus it is essential to take care of proper hyperparameter tuning. The performance might be not as good as the performance of NNs, but in case of an early development stage, it is manageable and a good compromise [14].

2.2 Image Processing in Agriculture

Estimating soil cover of arable land brings many benefits for efficiently, modern farming. The information which is got by semantic segmentation of images of soil can be used for instance to reduce the number of agrochemicals by destroying only the weed in the field with herbicides. In traditionally farming, herbicides are distributed with the same dose over the whole field [16]. Another advantage is to detect the percentage of soil covered by material. With this information, it is possible to protect the soil from erosion [17]. There are a lot of other use cases for how soil cover information can improve modern and sustained farming. The next section discusses some approaches to get an overview of the current researching and how they extract the soil cover information.

2.2.1 Researched Fields

Intelligent Weed Control

Reducing the amount of herbicides in agriculture-based semantic knowledge of the soil cover is the primary goal of the work of Andres Milioto, Philipp Lottes and Cyrill Stachniss in their paper "Real-time Semantic Segmentation of Crop and Weed for Precision" [16]. They use a CNN for pixel-wise semantic segmentation of crop field images. Their system is able to generalize well and operates around 20 images per second. Thus it is suitable for real-time estimation in the field [16].

Protect Soil from Erosion

Based on the knowledge of soil covered by material, soil erosion protection can be done. To get the percentage of covered soil, Peter Riegler-Nurscher, Johann Prankl, Thomas Bauer, Peter Strauss and Heinrich Prankl extended a basic RF algorithm with entangled features to get better segmentation results. The main improvement of this approach is the usage of probable labelling information of the direct neighbourhood pixels as an additional feature during training. More details are explained in their paper "An Integrated Image Analysis System for the Estimation of Soil Cover" [17].

Counting Fruits

Counting or detecting fruits is used to yield optimization and harvest scheduling. One way to achieve this goal is described in the paper "Monocular Camera Based Fruit Counting and Mapping With Semantic Data Association" [18]. The team designed a economical, lightweight, and fast CNN based fruit counting method with just one monocular camera and achieved very good performance compared to more complex system setups. Keeping the setup as simple as possible was one of the main challenges they mastered. Thus it is possible to run their method in regions where expensive hardware is not affordable.

Estimating Plant Health

Under the use of a CNN, Halimatu Sadiyah Abdullahi, Ray E. Sheriff and Fatima Mahieddine described a maize health classification method in their paper "Convolution Neural Network in Precision Agriculture for Plant Image Recognition and Classification" [19]. Given the constantly growing human population and the parse amount of food, the information about the plant health becomes more and more important. With their approach, they achieved an average prediction accuracy of of 99.58 percent.

2.3 Near Infrared (NIR)

Light interacts differently according to the surfaces of objects. This fact is used to get additional information from plant surfaces. The EMR from plants differs according to the light frequency [20]. Thus by filtering the frequency spectrum of the reflected light is from particular interest.
In the spectrum of EMR there are three regions of special interest for plant light interactions behaviour. The VIS region from about $400nm - 700nm$ for green leaves, the NIR region from about $700nm - 1100nm$ for absorption by

dry matter and the SWIR region from about $1100nm - 2500nm$ for absorption by water takes places [21]. Figure 2.16 show the reflectance spectra signature of green vegetation. The peak of the curve in the NIR region is clearly identifiable. This spectral behaviour only takes place of living green vegetation because of the EMR manipulation while the plant can do photosynthetic activities.

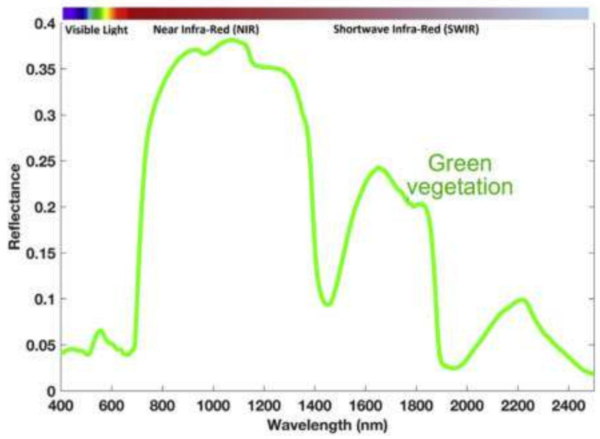


Figure 2.16: Reflectance spectra signature of green vegetation [21].

Filtering the reflected light before the camera sensor between the range of $700m - 1100nm$ will provide a Hyperspectral Image (HSI) from the NIR region.

3 Soil Cover Estimation

The main goal of this thesis is to perform the soil cover estimation in the field of agriculture by adding additional NIR and depth information to our semantic segmentation process. As it is described in Section 2.3, the NIR information gives us, especially in the field of agriculture, strong features to segment green and thus living organic elements of the image from the others. Based on the related work and the benefits of RFs we decided to use this algorithm for image segmentation because we want to develop the system in a fast time inference. Further, the RF algorithm gives us the possibility to estimate how frequently the additional NIR and depth information is chosen for splitting, what may be an indicator for the advantages of this extra information based on the information gain.
In the Subsection 3.1 the used camera setup is described. The practical construction and mechanical implementation is specified in Section 3.7.1 in detail. Camera calibration (3.2) and the stereo matching (3.4) are pre-steps before projecting of the NIR image to the RGB is possible. After projecting, the information sources (3.5) to the same spatial coordinate system the implementation of the algorithm and the particularities (3.7) also as the used features (3.6.2) are discussed.

3.1 Camera Setup

Our setup consists of three cameras. Two RGB cameras for stereo depth information and one camera between them which provides NIR information. A schematic representation is shown in Figure 3.1.

With this setup, it is possible to merge the 3D information of the current scene with the NIR information and RGB information. All three cameras are mechanically fixed on the same metal plate to guarantee that the distances d between the cameras are not changing. Modifying the distance of the cameras would require a recalibration of the whole setup. Further, the focus of the cameras is fixed thus that the sharp field of view is in the distance around one meter. One disadvantage of this setup is that the cameras are assembled side by side and thus the shared field of view decreases. This restriction is minimized by keeping the distance small, but large enough to get accurate

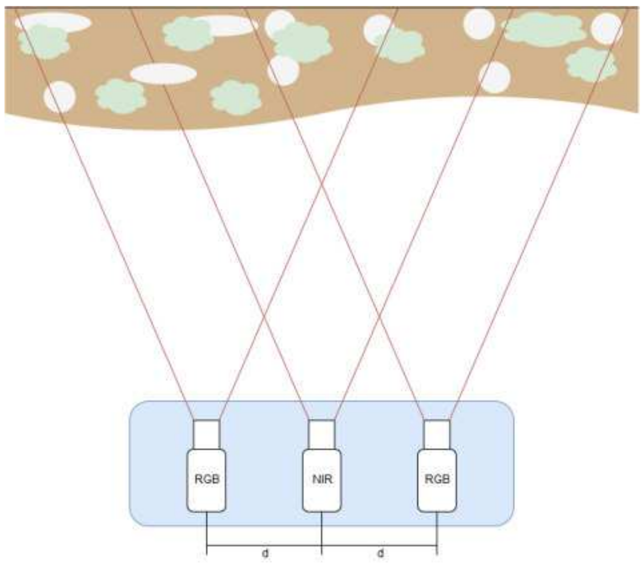


Figure 3.1: Camera setup

depth estimation.

3.2 Camera Calibration

Bring the world coordinate system and the camera coordinate system in relation depends on a set of lens and image specific parameters. Through calibration of the cameras, these parameters are determined and thus, is possible to figure out the relative position of the cameras. Further, we can bring the three cameras in relation to each other what is necessary for reprojecting the NIR image to the RGB image and depth information. The parameters are categorized into three groups, the linear intrinsic, the extrinsic and the nonlinear intrinsic parameters, also known as distortion coefficients. Figure 3.2 shows the transformation from the world coordinate system to the image plane coordinate system. Between these two spaces there is the camera coordinate system. The extrinsic $4x3$ parameter matrix E is used for a perspective transformation from the world to camera coordinate system. For the camera system to the image $2D$ image coordinate system the intrinsic $3x3$ parameter matrix K and the camera pinhole model is used [22]. The combination of these transformations is shown in Equation 3.1.

$$p_c = \begin{bmatrix} & E & \\ 0^T & & 1 \end{bmatrix} p_w \tag{3.1a}$$

$$p_i = \frac{1}{z_c} \begin{bmatrix} K & 0 \end{bmatrix} p_c \tag{3.1b}$$

$$p_i = \frac{1}{z_c} \begin{bmatrix} K & 0 \end{bmatrix} \begin{bmatrix} & E \\ 0^T & 1 \end{bmatrix} p_w \tag{3.1c}$$

Where p_w is representing a homogeneous point $\left[X_w, Y_w, Z_w, 1\right]^T$ in the world coordinate system, p_c a homogeneous point $\left[x_c, y_c, z_c, 1\right]^T$ in the camera coordinate system and p_i a homogeneous point$\left[u_i, v_i, 1\right]^T$ in the image coordinate system [22].

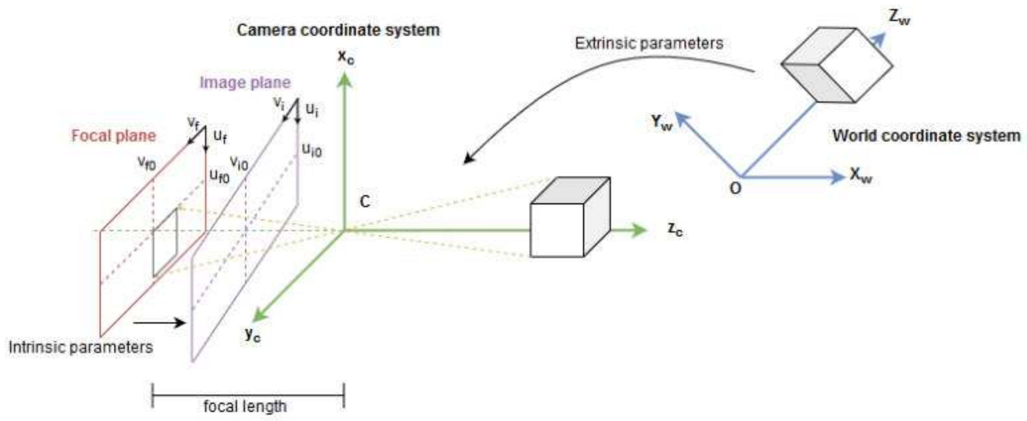


Figure 3.2: Pinhole camera model and relation between the camera coordinate system and the world coordinate system.

The fraction $\frac{1}{z_c}$ in Equation 3.1 comes from the pinhole camera model. The camera pinhole model describes the relation of coordinates of a point in $3D$ camera space and its projection onto the image plane as it is illustrated in Figure 3.3.

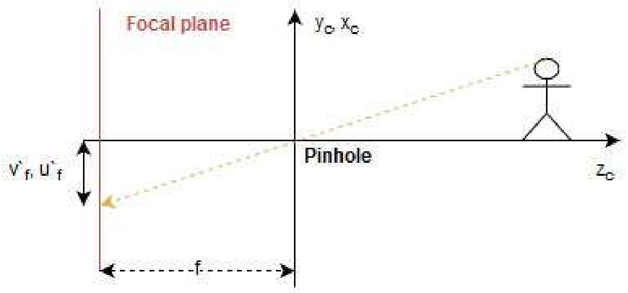


Figure 3.3: The geometry of a pinhole camera.

Equation 3.2 delineates this projection.

$$\begin{bmatrix} u'_f \\ v'_f \end{bmatrix} = \frac{1}{z_c} \begin{bmatrix} f & 0 \\ 0 & f \end{bmatrix} \begin{bmatrix} x_c \\ y_c \end{bmatrix} \tag{3.2}$$

Where $[\,u'_f,\, v'_f\,]^T$ are the projected points on the image plane related to the center. f is the focal length of the model. Comparing the Equations 3.1a and 3.2 leads to the conclusion that $\left[\begin{smallmatrix} f & 0 \\ 0 & f \end{smallmatrix}\right]$ is the intrinsic parameter matrix of the pinhole camera model. Based on the fact that this model does not include a lens, f is the only intrinsic parameter. In our approach we are using cameras with lenses and photo sensors, thus we get additional intrinsic parameters which are discussed in the following section [22].

3.2.1 Intrinsic Parameters

Intrinsic parameters are responsible to describe the camera internal composition, encompass the focal length f, scale factor relating distances to pixel, a skew coefficient for u,v relation and a principal point $[\,u_{i0},\, v_{i0}\,]^T$ [22].
Based on the Equation 3.3, which is the homogeneous representation of the Equation 3.2, the composition is described.

$$\begin{bmatrix} u'_f \\ v'_f \\ 1 \end{bmatrix} = \frac{1}{z_c} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} \tag{3.3}$$

To get a relation between the focal length in terms of pixels and the actual focal length, they get scaled by the scaling factors m_x and m_y [22]. By substituting $fm_x = \alpha_x$ and $fm_y = \alpha_y$ the equation gets modified as it described below(3.4).

$$\begin{bmatrix} u'_i \\ v'_i \\ 1 \end{bmatrix} = \frac{1}{z_c} \begin{bmatrix} \alpha_x & 0 & 0 & 0 \\ 0 & \alpha_y & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} \tag{3.4}$$

Further, the skew coefficients are added to the intrinsic matrix delineate by Equation 3.5.

$$\begin{bmatrix} u''_i \\ v''_i \\ 1 \end{bmatrix} = \frac{1}{z_c} \begin{bmatrix} \alpha_x & -\alpha_y cot(\gamma) & 0 & 0 \\ 0 & \alpha_y \frac{1}{sin(\gamma)} & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} \tag{3.5}$$

The actual skew of the image plane in relation to the focal plane is caused during the manufacturing of the cameras. The angle γ describes the extent of

the skew [22].
Because the center of the image and the origin of the image coordinate system are not equal, the centroid offsets u_{i0} and v_{i0} are added to the intrinsic matrix, shown in Equation 3.6. The point described through $\left[\begin{smallmatrix} u_{i0}, & v_{i0} \end{smallmatrix}\right]^T$ is also known as optical center.

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \frac{1}{z_c} \begin{bmatrix} \alpha_x & -\alpha_y cot(\gamma) & u_{i0} & 0 \\ 0 & \alpha_y \frac{1}{sin(\gamma)} & v_{i0} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} \tag{3.6}$$

The determination of these parameters is discussed in Section 3.2.4.

3.2.2 Distortion Coefficients

The pinhole camera model only deals with geometrical distortions caused by the object projection to the image plane and the inaccuracies during manufacturing. Distortions in a practical camera system are also caused by lens properties and there geometric arrangement related to the image plane. There are existing many of different distortions types but in this thesis we will focus on the radial (3.2.2) and tangential (3.2.2) distortions. The distortions are described trough coefficients which are determinated during the calibration process.

Radial Distortions

Radial distortions are the most obvious ones. They are especially noticed when we expect straight lines but see curved lines in an image. Figure 3.4 illustrates the three types of radial distortion, the barrel, moustache and pincushion distortion.

For simplification the radial distortion is approximate as low degree polynomial 3.7 [23]. k_1,...,k_n are the distortion coefficients.

$$\lambda = 1 + \sum_{i=1}^{n} k_i r^{2i}, n \ll \tag{3.7a}$$

$$r^2 = u_i^2 + v_i^2 \tag{3.7b}$$

To get the distorted image points $\left[\begin{smallmatrix} u_d, & v_d \end{smallmatrix}\right]^T$ we have to apply λ to the focal plane points $\left[\begin{smallmatrix} u'_f, & v'_f \end{smallmatrix}\right]^T$. This is done in the Equation 3.8 for $n = 3$.

$$\begin{bmatrix} u_d \\ v_d \end{bmatrix} = (1 + k_1 r^2 + +k_2 r^+ k_3 r^6) \begin{bmatrix} u'_f \\ v'_f \end{bmatrix} \tag{3.8}$$

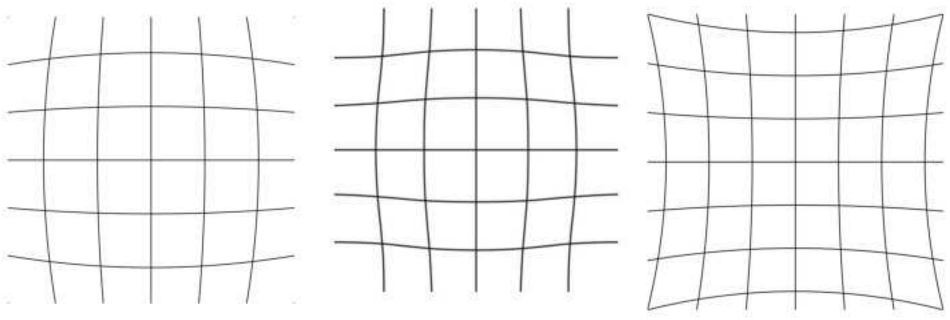


Figure 3.4: Types of radial distortions. (a) Barrel Distortion, (b) Mustache Distortion and (c) Pincushion Distortion

Tangential Distortions

The tangential distortions occur when the lens is not completely parallel to a photosensor. Figure 3.5 shows the effect of the distortion to an image. These distortions can be approximated through the Equation 3.9 [23].

$$\begin{bmatrix} u_d \\ v_d \end{bmatrix} = \begin{bmatrix} u'_f \\ v'_f \end{bmatrix} + \begin{bmatrix} 2p_1 u'_f v'_f + p_2(r^2 + u'_f) \\ p_1(r^2 + v'_f) + 2p_2 v'_f u'_f \end{bmatrix} \tag{3.9}$$

Wherein p_1 and p_2 are the distortion coefficients.

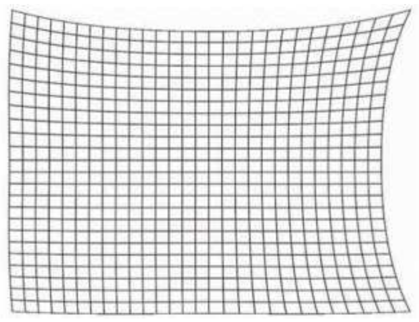


Figure 3.5: The geometry of a pinhole camera.

Merging the Distortions

Finally, we can combine the distortions and the intrinsic parameter matrix to get a good approximation of projection from the camera coordinate system to the image plane, including the most decisive camera system influences.

$$\begin{bmatrix} u \\ v \end{bmatrix} = K \begin{bmatrix} u_d \\ v_d \end{bmatrix} \tag{3.10}$$

Wherein $[\, u_d,\ v_d \,]^T$ is a combination of the radial and tangential distortions:

$$\begin{bmatrix} u_d \\ v_d \end{bmatrix} = \lambda \begin{bmatrix} u'_f \\ v'_f \end{bmatrix} + d \tag{3.11}$$

and

$$d = \begin{bmatrix} 2p_1 u'_f v'_f + p_2(r^2 + u'_f) \\ p_1(r^2 + v'_f) + 2p_2 v'_f u'_f \end{bmatrix} \tag{3.12}$$

3.2.3 Extrinsic Parameters

The extrinsic parameter matrix P projects a point from the world to the camera coordinate system as it is shown in Figure 3.2. The matrix P consist out of two parts:

$$P = \begin{bmatrix} R & t \end{bmatrix} \tag{3.13}$$

Wherein R is the rotation matrix and t the translation vector [22]. The rotation matrix consists out three independent rotations around each axis. They are defined as follows:

$$R_x(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\psi) & -sin(\psi) \\ 0 & sin(\psi) & cos(\psi) \end{bmatrix} \tag{3.14a}$$

$$R_y(\theta) = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix} \tag{3.14b}$$

$$R_z(\phi) = \begin{bmatrix} cos(\phi) & -sin(\phi) & 0 \\ sin(\phi) & 0 & cos(\phi) \\ 0 & 0 & 1 \end{bmatrix} \tag{3.14c}$$

Thus the rotation matrix can be thought of a sequence of these three rotations as its described through Equation 3.15.

$$R = R_z(\phi) R_y(\psi) R_x(\theta) \tag{3.15a}$$

$$R = \begin{bmatrix} cos(\theta)cos(\phi) & sin(\psi)sin(\theta)sin(\phi){-}cos(\psi)sin(phi) & cos(\psi)sin(\theta)cos(\phi){+}sin(\psi)sin(phi) \\ cos(\theta)sin(\phi) & sin(\psi)sin(\theta)sin(\phi){+}cos(\psi)sin(phi) & cos(\psi)sin(\theta)cos(\phi){-}sin(\psi)sin(phi) \\ -sin(\theta) & sin(\phi)cos(\theta) & cos(\phi)cos(\theta) \end{bmatrix} \tag{3.15b}$$

The translation matrix t holds the origin position of the world coordinate system, expressed in coordinates related to the camera coordinate system [22]. Thus, the system origins are related by 6 parameters, the three angels and the three positioning values.

3.2.4 Camera Parameter Estimation

One approach to calibrate the camera is to use a checkerboard. The board is positioned in many different poses in front of the camera, as it is illustrated in Figure 3.6. With the knowledge about the (scaled) positions of the checkerboard corners in world coordinates and the corresponding detected corners in the image plane, a relation between the respective points can be done. Through these relations, the intrinsic and extrinsic parameters can be estimated.
In our approach, we used the OpenCV (Intel open source computer vision library) [24] for calibration, which internally based on the Zhang's calibration method [25].

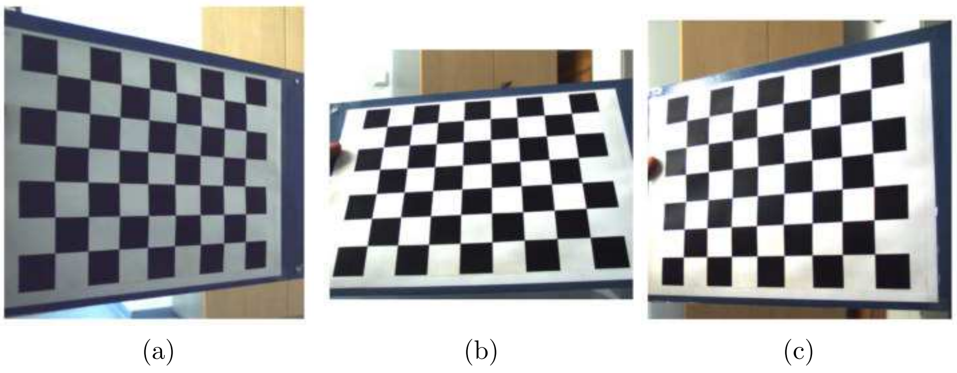

(a) (b) (c)

Figure 3.6: Different orientations of the checkerboard.

Zhang's Calibration Method

The calibration process after Zhang's method, roughly based on seven steps [26] [27] and includes the intrinsic parameter, extrinsic parameter and radial distortion coefficient estimation.

Step 1

M images from different perspectives of the checkerborad are captured [26]. The camera or the checkerborad should be moved thus the model is in many different poses and at least in every image area present once.

Step 2
The second step is to detect all corners of the pattern in each of the M images. Assuming N corners per image will result in MxN image coordinate points $[\, ^{u_k,}\ ^{v_k}\,]^T$ [26].

Step 3
Now the homographies H_1,...,H_M are estimated for each independent model image. A homography describes the mapping of the observed image coordinates to the real $3D$ world coordinates. The homography matrix can be estimated through Direct Linear Transformation (DLT) based on the linear equation system, which can be defined with the world coordinate system related image points from step 2. This step is described in detail in the article [26].

Step 4
By ignoring the distortion coefficients the intrinsic parameter can be determinated out of the homography matrices H_1,...,H_M. For $M > 2$ and $N > 1$ assuming a skew of 1, the solution is unique [26].

Step 5
With the knowledge of the intrinsic parameters, the extrinsic parameter matrix P can also be estimated for each captured image. R and t are unique for each view because there is no depth related information known during the calibration process. Thus the extrinsic parameters are given with a unique scale factor for each view [26].

Step 6
The next step is to determine the radial distortions coefficients by least-squares minimization [26].

Step 7
In the end, the estimated parameters are used as an initial guess for the parameter refinement. The refinement is done by nonlinear optimizations over all captured views [26].

The implementation of the OpenCV improves this method by additionally estimating the tangential distortion coefficients and refine the minimization process [25].
Finally, a rectifying of the images can be done by using the determined camera matrices as it is illustrated in Figure 3.7. Where the left figure 3.7a shows the

distorted image and the right figure 3.7b the undistorted one. On the lower and right side area of the undistorted image, the corrections are recognizable.

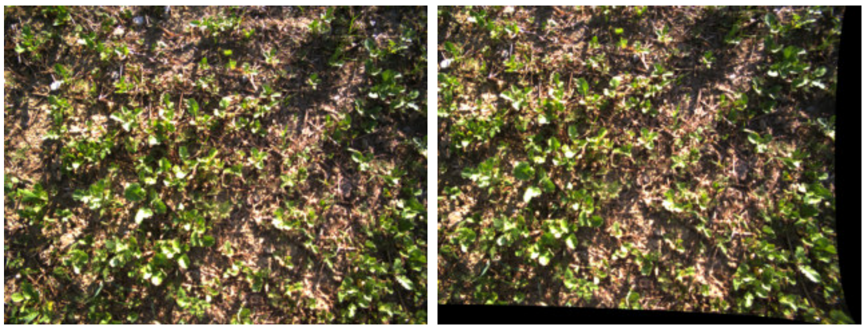

(a) (b)

Figure 3.7: (a) Distorted image (b) Undistorted image

3.3 Stereo Calibration

The goal of stereo calibration is to bring all the camera coordinate systems in relation through defining the rotation and translation matrices between them, as is shown in Figure 3.8.

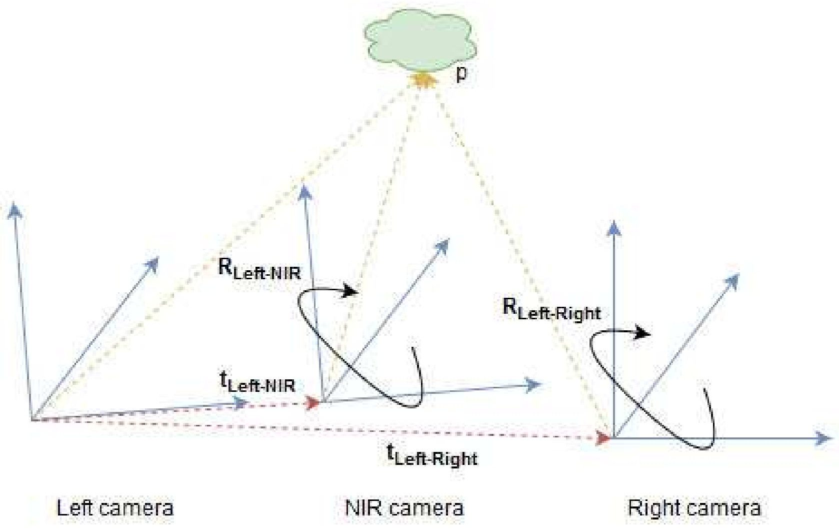
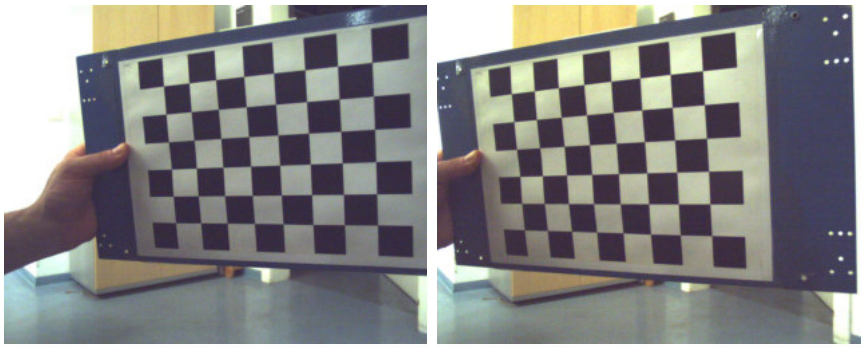


Figure 3.8: Stereo - NIR camera setup.

In our approach we have to do two stereo calibrations, the one between the left and the right camera and the one between the left and the NIR camera to

estimate the rotation matrix $R_{Left-NIR}$ and the translation vector $t_{Left-NIR}$ also as the rotation matrix $R_{Left-Right}$ and the translation vector $t_{Left-Right}$. We decided to choose the left camera coordinate system as origin. The origin camera system can chosen freely.
The calibration is based on pattern recognition in stereo images which were captured simultaneously. An example is illustrated in Figure 3.9. A pattern like one of the checkerboard can be used for this method. If the same corner is identified in both images, we can calculate the projection matrix between these two cameras. The calculation and the needed requirements are described generally in the section below.



(a) (b)

Figure 3.9: Simultaneously captured stereo calibration images.

Before calibration, the cameras must be physically fixed thus that the captured images are always in the same relation to each other. Further, the intrinsic parameters must be determined before because the stereo calibration is executed in undistorted image plane coordinates [28]. Additionally the extrinsic parameters R_1, t_1 and R_2, t_2 of the left and right camera are needed. They can be determined uniquely for each stereo image pair, as it is described in step 5 of the Zhang's calibration method in Section 3.2.4.
For an arbitrary point p in world coordinates the Equations 3.16a and 3.16b describing the corresponding points p_1, p_2 for the left and right camera [28].

$$p_1 = R_1 * P + t_1 \tag{3.16a}$$

$$p_2 = R_2 * P + t_2 \tag{3.16b}$$

The points p_1 and p_2 are related through the perspective matrix [28]:

$$p_1 = R^T(p_2 - t) \tag{3.17}$$

Through these three equations, the rotation matrix R and translation vector t of the perspective projection matrix can be determined:

$$R = R_2(R_1)t \tag{3.18a}$$

$$t = t_2 - R * t_1 \tag{3.18b}$$

This process is performed for multiple corner points of each stereo image pair and several different views thus the error of the resulting perspective matrix decreases. We used the OpenCV library [24] for stereo calibration which internally works with the same method.
In the upper half of Figure 3.10 an unrectified stereo image pair is shown. In the bottom half of Figure 3.10 the same stereo image pair is illustrated after intrinsic and stereo rectification. Before the rectification, the stones surrounded with a red circle do not lie on the same horizontal line in the left and right image. After rectification, the stones lie on the same line with a horizontal shift. This behaviour is needed for stereo matching what is discussed in the next Section 3.4.

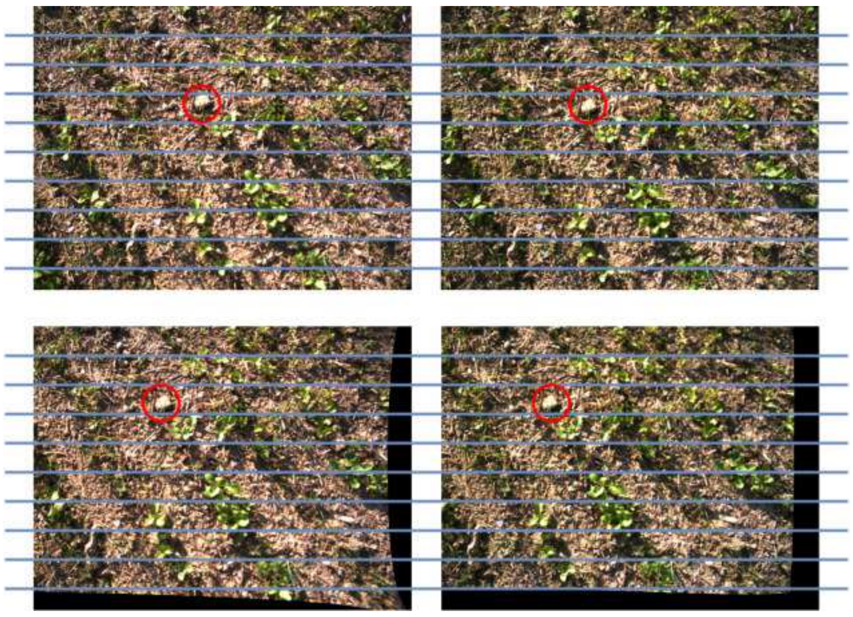


Figure 3.10: Unrectified stereo images (upper half) compared to the rectified stereo images (bottom half).

3.4 Stereo Matching and Disparity Map

Matching a point from the world coordinate system in two different camera views is known as stereo matching. The matching can only be done for the overlapping image areas. In our approach, we use the fast and straightforward block matching algorithm to find corresponding points. The method is described in the following section. The disparity map(3.4.2) is generated based on the results of stereo matching an can be interpreted as unscaled and unitless depth information [28].

3.4.1 Block Matching

The principle of block matching is to find the same pixel area of one image in another. This is done by sliding a pixel window of the first image L over the whole or a restricted area of the second image R. At each location of the pixel window on the second image, an evaluation metric is calculated. The evaluation metric is an indicator of the similarity of the two regions [28]. Two primary evaluation metrics are described below:

$$MAD = \frac{1}{N^2} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} |L_{i,j} - R_{i,j}| \tag{3.19a}$$

$$MSE = \frac{1}{N^2} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} (L_{i,j} - R_{i,j})^2 \tag{3.19b}$$

Wherein MAD is the Mean Absolute Difference and MSE is the Mean Squared Error. $N = n^2$ is the window size. The region with the lowest deviation error is taken as matching result.
Working with rectified stereo images brings the benefits of epipolar geometry. That means it is not necessary to search the whole image to find a corresponding region. Only the regions along the related epipolar line in the second image have to be scanned for matches [28]. This decreases the computational time complexity of $O(N^2)$ to $O(N)$.
In our implementation, we used the OpenCV [24] stereo block matching algorithm.

3.4.2 Disparity Map

After matching a $3D$ point in both images, the disparity can be calculated as it is described by Equation 3.20.

$$d = x - x' \tag{3.20}$$

Wherein x is the distance from the point p_1 to the camera center C_1 and x' is the distance from the point p_2 to the camera center C_2, as it is illustrated in Figure 3.11

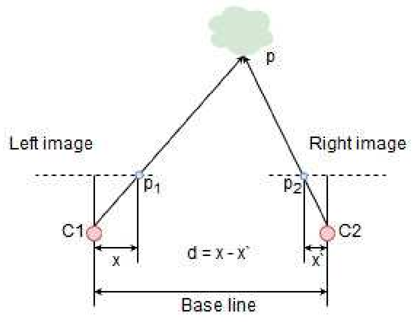


Figure 3.11: Disparity calculation for one match.

Doing this for each match is resulting in a disparity map. Because some scenes are not seen from both cameras and regions which could not be matched by block matching, not every pixel gets a corresponding disparity. This leads to holes in the map as it is shown in Figure 3.13b. For projection of the NIR image, a comprehensive disparity is necessary. To achieve this, we are using a disparity filter, described in the next section.

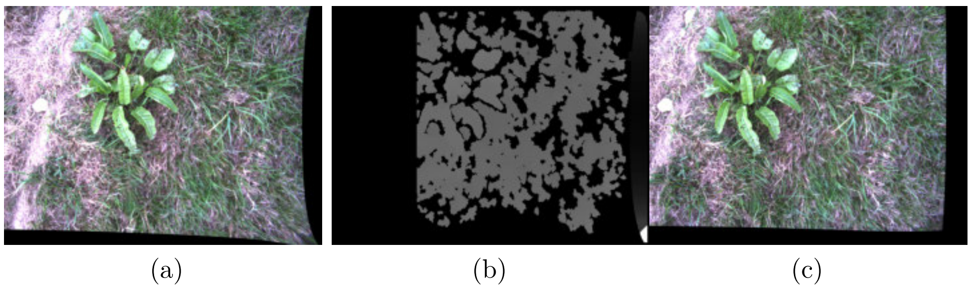


Figure 3.12: Disparity map (b), calculated from the rectified stereo image pair (a)(c).

3.4.3 Disparity Filter

To approximate the missing disparity values, a filter can be used. In our approach, we are using the Weighted Least Squares (WLS) disparity filter provided by OpenCV [24]. The WLS filter is a guided filter what means that the features of the original input images are used for an density approximation.

In general a guided filter can be described through Equation 3.21 [29].

$$q_i = \sum_j W_{ij}(I)p_j \tag{3.21}$$

Where q_i is the output image (disparity), p_j the input image (disparity) and I the origin reference image. i and j are the pixel indexes and W is the filter kernel.
The WLS filter kernel provided by OpenCV [24] uses two input disparities maps and one reference input image for processing. One disparity map is generated conventionally and the other one is generated by interchanging the left and right input images for block matching and disparity determination. Figure 3.13a shows the filtered disparity compared to the perforated disparity 3.13b. As we can see, the filtered disparity map has much fewer holes than the unfiltered. With these results, we are able to do a sufficiently accurate NIR image projection 3.5.

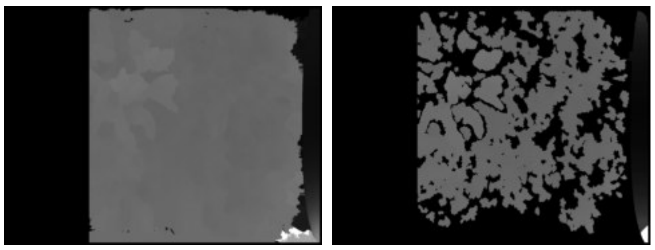

(a) (b)

Figure 3.13: Filter disparity (a) compared to non filtered (b).

3.5 NIR Image Projection

Finally, the NIR image can be projected to the left camera coordinate system thus that it is possible the overlay the NIR and the RGB information pixel-wise. The projection can be made in four steps.

Step 1

With the disparity map(3.4.2), the perspective projection matrix (3.3) between the left and right camera and the respective camera parameters (3.2.4) the image points can be projected to the world coordinate system. This is done via the triangulation principal.

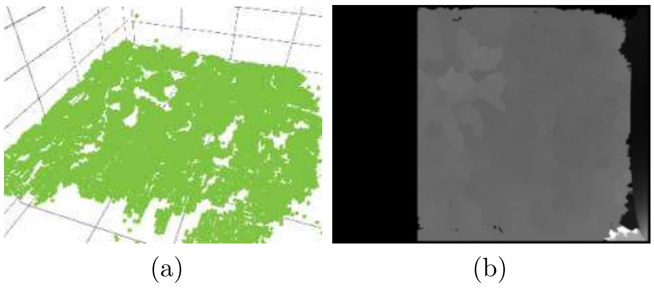


Figure 3.14: Reconstructed 3D point in the world coordinate system (a) and the corresponding disparity map (b).

Step 2
Second, the $3D$ points have to be transformed to the left camera coordinate system by using the extrinsic parameters (3.2.4).

Step 3
The third step is do project the points from the left camera to the NIR camera coordinate system by using the rotation matrix $R_{Left-NIR}$ and the translation vector $t_{Left-NIR}$.(3.3)

Step 4
The last step is to project the points from the NIR camera coordinate system to the image plane.

After these four steps we are able to relate a RGB pixel value to a 3D point also as the corresponding NIR pixel value. Figure 3.15 illustrates the resulting NIR image projection overlaying a RGB image partly transparent visible in the left and upper side of the figure.

Now we can feed our algorithm with the necessary information to do semantic image segmentation based on RGB, NIR and depth information.

3.6 Features

A feature holds a piece of information from a specific area of the image. What information features are holding is defined through their construction and how they are calculated. Thus it is possible to calculate the derivation of a specific image area to get the information about the degree of change within this area. Further, features can be calculated on different image channels. Thus it is possible to get derivation feature information from a simple grayscale image

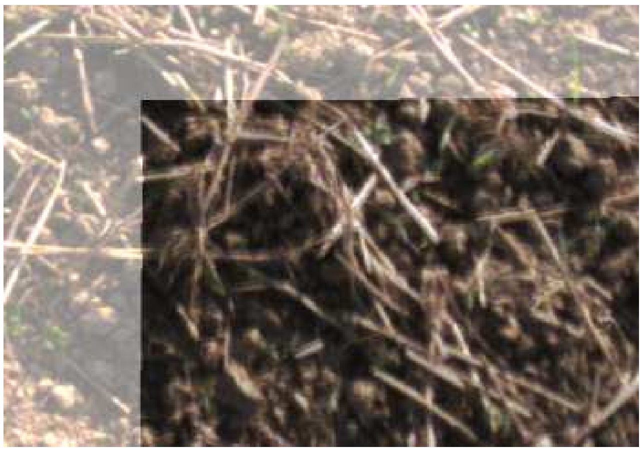


Figure 3.15: NIR image partly overlaying RGB image.

also as from a normalized green channel image. Before the used features are discussed, the different image channels are described in the following section.

3.6.1 Channels

Channels are different image interpretations. Thus it is possible to make them more robust against external influences, but also extract more information out of the raw RGB, NIR and depth channel by transforming or normalize them. In our approach, we transform the RGB channels into the L*a*b* colour space to get more robust against lightness changes. Further, we normalize the basic NIR image to the Normalized Difference Vegetation Index (NDVI) channel to get specific knowledge about the vegetation environment. Additionally, we are using a normalized green channel because thus it is possible to split the green vegetation better from the rest. The depth information is added by using the disparity map. In the following subsection, the channels are described briefly.

LAB Color Space

The RGB colour space is not invariant against illumination, and the channels are not linearly separable. Additionally, the RGB colour space does not cover the complete perceptible range of colours and also holds no direct useable lightness information. To solve these restrictions, the LAB colour space was introduced. Where the L stands for the lightness, A for the red/green value difference and B for the blue/yellow value difference as it is shown in Figure 3.16. Thus the image can be separated into illumination channel and linear separable a and b channels. The LAB values can be calculated from the RGB colour space over the XYZ colour space as it is described by Equation 3.22a. After the calculation of the XYZ colour space values it is possible to calculate the LAB values by using the Equations 3.22b, 3.22c and 3.22d [30].

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4125 & 0.3576 & 0.1804 \\ 0.2127 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9502 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{3.22a}$$

$$L = 116\sqrt[3]{\frac{Y}{Y_n}} - 16 \tag{3.22b}$$

$$A = 500(\sqrt[3]{\frac{X}{X_n}} - \sqrt[3]{\frac{Y}{Y_n}}) \tag{3.22c}$$

$$B = 200(\sqrt[3]{\frac{Y}{Y_n}} - \sqrt[3]{\frac{Z}{Z_n}}) \tag{3.22d}$$

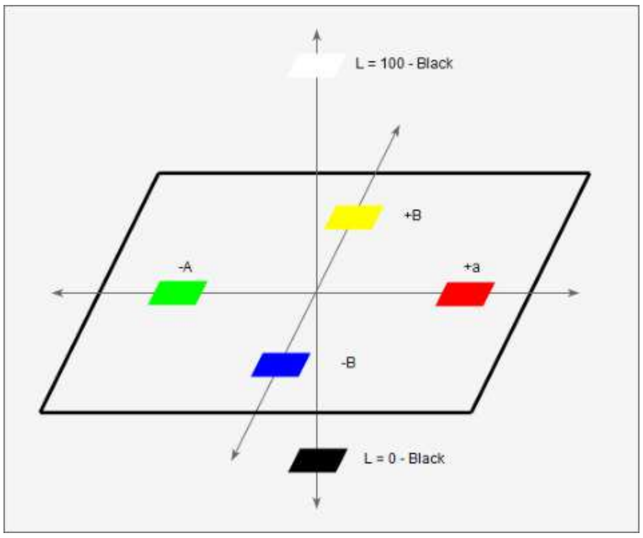


Figure 3.16: LAB colour space.

Wherein X_n, Y_n, Z_n are constants related to the kind of illumination. Two very common illumination types are $D65$ and $D50$.
We are using the LAB colour space to get rid of the lighting difference between the images. Thus the algorithm gets more robust against lighting because it is not trained for specific lightness conditions rather than for independent lightness conditions.

Normalized Green

The normalized green channel represents the difference between the red and green channel and especially holds information about the green vegetation. Thus the difference between vegetation and the other objects gets larger. In general, green regions are good indicators for living vegetation, but if other objects are also green, this assumption is false. This channel is calculated as it is defined by Equation 3.24.

$$NG = \frac{G - R}{G + R} \tag{3.23}$$

Wherein NG is the normalized green value calculated by using the green channel G and red channel R. Figure 3.17 shows the basic RGB image 3.17b compared to the normalized green image 3.17b. It is clearly recognizable that the green areas can be extracted from the rest because of the large distance between the values. Based on the larger value distances the feature thresholding gets more sufficient.

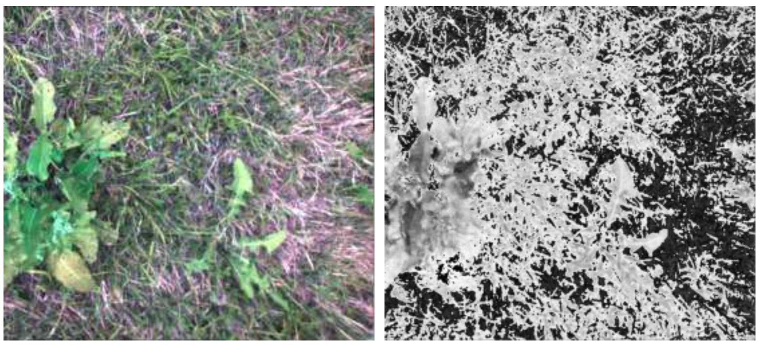

(a) (b)

Figure 3.17: Normalized green channel (b) compared to the captured scene (a).

Near Infrared (NIR)

The projected NIR channel is used directly and in two normalized forms. Especially, the NIR channel should provide us with reliable information about the green vegetation. Thus it is possible to separate the data with a high information gain related to living and non-living organic material. The two normalized NIR channels are described in the following Sections 3.6.1and 3.6.1.

Normalized Difference Vegetation Index (NDVI)

The NDVI channel quantifies the vegetation by measuring the difference between the NIR channel and the red channel of the RGB image. The NDVI values are always in a range from -1 to $+1$. A value near to -1 indicates that it is likely water. Where values near to $+1$ are indicators for living green vegetation. The values can be calculated as it is described by Equation 3.24.

$$NDVI = \frac{NIR - R}{NIR + R} \tag{3.24}$$

Wherein, R stands for the red channel values and NIR for grayscaled NIR values. This calculation is done for every pixel to get a full range NDVI channel. Figure 3.18 clearly shows the extraction of vegetation. This vegetation extraction approach is more general than with the normalized green channel (3.6.1) because the extraction is not only based on colour information, rather than real vegetation information from the NIR channel.

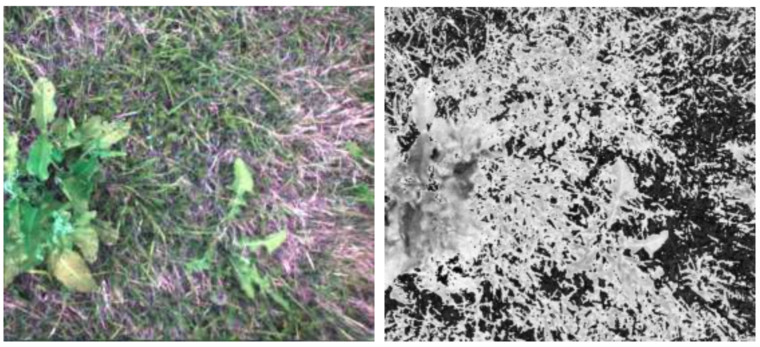


Figure 3.18: NDVI channel (b) compared to the captured scene (a).

Optimized Soil-Adjusted Vegetation Index (OSAVI)

The OSAVI channel is another normalization of the NIR channel in combination with the red image channel optimized for agricultural monitoring [31]. The normalization is defined through the Equation 3.25.

$$OSAVI = \frac{NIR - R}{NIR + R + 0.16} \tag{3.25}$$

Where the coefficient 0.16 describes the optimal value to minimize the variation with soil background [31]. The calculation is similar to the one of the NDVI channel besides the introduced normalization coefficient.

Depth information

In our approach, we use the depth information in the form of the disparity map (3.4.2) because we only need a relative relation between the values and no absolute depth information. The depth information channel is different from the others in the sense of feature thresholding. Thus if we split the image based on depth information, related regions are separated. However, in an early stage of the algorithm execution, this information can nevertheless be helpful in terms of splitting the flat soil from objects on it. As it is shown in Figure 3.19, the different grayscales are representing different depths which could be split into soil and objects on the soil.

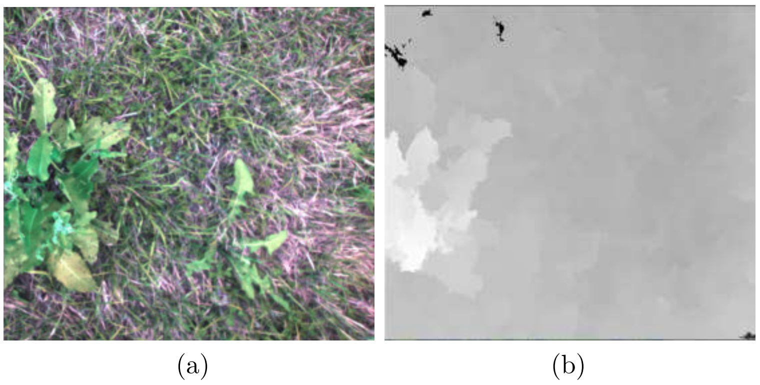

(a) (b)

Figure 3.19: Disparity map (b) compared to the captured scene (a).

3.6.2 Features

As it is described above, features hold specific information of an image area. In this thesis, we focus on five different types of features. Each feature extracts different information from the image. Some of the features are more reliable at early algorithm execution stages and others more on later stages. The used features are described in the following sections.

Absolute

The absolute feature is calculated by the mean value of the region R with an relative pixel offset Δ to the pixel coordinate x as it is defined by Equation 3.26.

$$f_{abs,C}(x,\Delta,R) = I_C(R(x+\Delta)) \tag{3.26a}$$

$$C \in \{L, A, B, NG, NDVI, OSAVI\} \tag{3.26b}$$

This feature is calculated and implemented on each image channel described above. With this information, the specific channel is separated by a threshold according to the absolute mean value of a region. Thus it is the purest form of grouping the pixel values of a channel.

Unary

With this feature, the absolute mean difference of two regions of the size of R at two-pixel coordinates is calculated. The first region is placed at the current evaluation position, and the offset Δ shifts the second to this position. This is formalized by Equation 3.27.

$$f_{unary,C}(x,\Delta,R_1,R_2) = I_C(R_1(x)) - I_C(R_2(x+\Delta)) \tag{3.27a}$$

$$C \in \{L, A, B, NG, NDVI, OSAVI\} \tag{3.27b}$$

Like the absolute feature, this is also applied on each channel. It is an indicator for the similarity of the current evaluation region compared to a nearby region.

Pairwise

Similar to the unary feature the pairwise feature is also an indicator for similarities of neighbouring regions with the variation that the difference is not calculated against the current evaluation region rather to another neighbouring region shifted by the offset Δ_1. The second neighbouring region is shifted by Δ_2. This feature can also be seen as an image gradient, thus it is robust against illumination differences and can be applied to each image channel [17]. It is described through Equation 3.28.

$$f_{pairwise,C}(x,\Delta_1,R_1,\Delta_2,R_2) = I_C(R_1(x+\Delta_1)) - I_C(R_2(x+\Delta_2)) \tag{3.28a}$$

$$C \in \{L, A, B, NG, NDVI, OSAVI\} \tag{3.28b}$$

Variance

The variance feature describes the variance of a specific region R around the current evaluation position shifted by an offset Δ as it expressed through Equation 3.29.

$$f_{var,C}(x,\Delta,R) = VAR(I_C(R(x+\Delta))) \tag{3.29a}$$

$$C \in \{L, A, B, NG, NDVI, OSAVI\} \tag{3.29b}$$

This feature is an indicator for the level of scattering of a region. That means, for example, a channel representation can be split into flat and less flat regions by evaluating this feature on it.

Linearness-/Pointness-Feature

This feature consists of two parts. The first part describes the variance of region R at the point x from the derived image, as is shown by Equation 3.30. Where the resulting value indicates the reliability of the value.

$$f_{lt,C}(x,R) = VAR(arctan(\frac{\partial_x I_C(R(x))}{\partial_y I_C(R(x))})) \tag{3.30a}$$

$$C \in \{L, NDVI\} \tag{3.30b}$$

The second part of this feature indicates the line structure in the region R. Which is used to calculate the mean derivations at the point x as it is described by Equation 3.31.

$$f_{g,C}(x,R) = MEAN(||\partial_x I_C(R(x))|| + ||\partial_y I_C(R(x))||) \tag{3.31a}$$

$$C \in \{L, NDVI\} \tag{3.31b}$$

The feature is only implemented for the L and the $NDVI$ image channel because it has massive computational complexity and is consuming much memory.

3.7 Implementation

The following section describes how image segmentation based on the the RF algorithm is implemented. Our implementation approach is described in Section 3.7.3. Before the algorithm can train or classify images, preparatory work has to be done. The preparatory work can be split into the image capturing (3.7.1) and data preparation (3.7.2). Wherein the first section deals with a special build camera-box for autonomous image capturing and related streaming software. In the second section, the principal data preparation, according to the data container and labelling software, is explained.

3.7.1 Image Capturing

For image capturing, we designed a camera-box which can stream images autonomously to a computer on which a streaming software is running. The box has to be autonomous because it has to be possible to use it without

significant circumstances to an agriculture field or fix it on a vehicle without any wiring.
Additional to the box implementation, an image streaming service has to be realized on the used ODROID-XU4 micro controller [32] board as well as a client streaming software to receive the images. The realization of this related work is described in the following sections.

Camera-Box Components

The camera-box consists of five main components. They are all necessary to realize the required functionality. Figure 3.20 shows the box and its components, which are described in the following subsections.

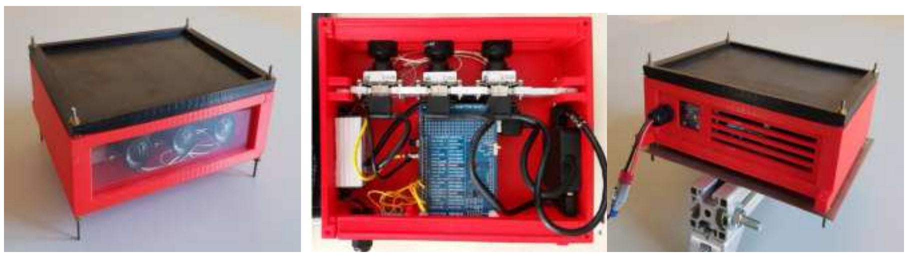

(a) (b) (c)

Figure 3.20: Different views on the camera box.

Power Management

For the autonomous power supply, we are using a conventional power bank and a twelve to five voltage DC-DC voltage converter, which is shown in Figure 3.20b on the lower left side inside the box.

Cameras

The camera construction which is shown in the upper part of the box in Figure 3.20b consist out of two UI-32521LE RGB cameras with a resolution of 1600x1200 pixels and one UI-3241LE-NIR-GL NIR camera with a resolution of 1280x1024 pixels in the middle of them [33].

Wireless

For the wireless communication of the box software with the PC software, a conventional wireless stick is used. Thus it is possible to open a hotspot on the box side of the application and set up a connection to it.

Micro Controller

The host the box software we are using a ODROID-XU4 [32] single board computer with an SAMSUNG Exynos 5422 processor and 2 GB RAM. This microcontroller can handle the camera image capturing and sending the images over wireless LAN at the same time by relatively low power consumption.

Casing

The casing is 3D printed and joins all the components together, which are described in the upper sections. It consists of four main parts. The fixing construction and the cover is shown in Figure 3.21a and 3.21b. The fixing construction is designed to hold the three cameras in the front as also the microcontroller board and voltage converter in the back part. The front consists of a plexiglass. Thus it is guaranteed that the cameras are protected from environmental influences. The back of the box is a 3D printed element with ventilation slots as Figure 3.21a illustrates. The ventilation slots are necessary because of the heat caused by the cameras and the microcontroller. Finally, the cover holds all the 3D printed components together and protects them.

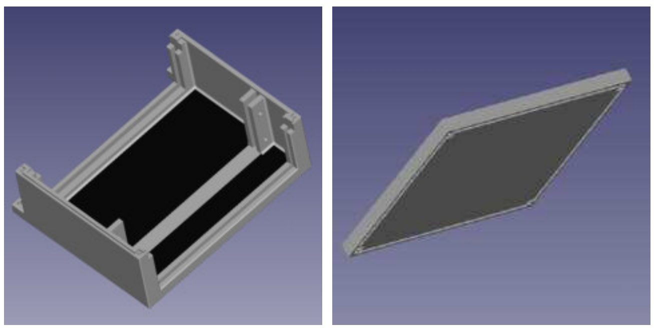

(a) (b)

Figure 3.21: Model of the 3D printed camera casing.
(a) Fixing construction and (b) Cover

Capturing Software

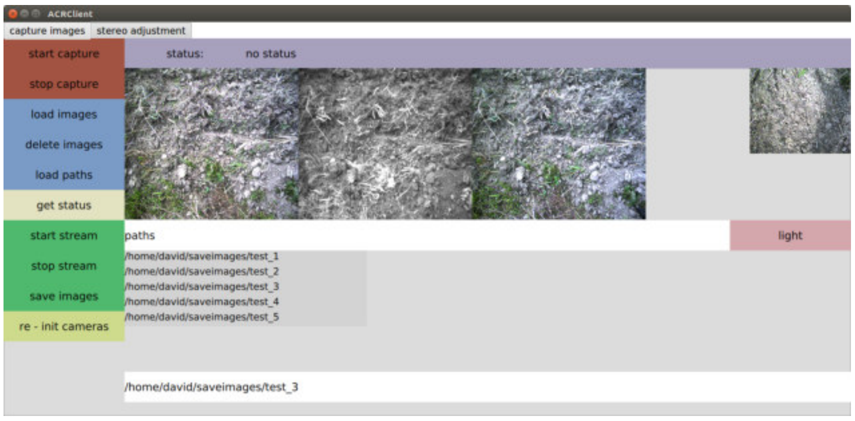
The capturing software consists of two central parts. The one which is running on the microcontroller inside the camera box and the one which runs on a conventional computer. These two applications communicate over a wireless LAN connection which is provided by the operating system of the microcontroller. In the following two sections, these two software parts are explained in more detail.

Box Software

On the microcontroller, software consisting of two main parts is running. The first part is responsible for the parallel image capturing of all three cameras. This is realized with an external trigger signal which is connected to all cameras. When the trigger signal rises, the cameras take an image which is saved to the internal memory, and the camera driver is notified that new images are available. The notification is passed to the software, and so it is possible to load the three synchronous captured images to the internal box software memory. Second, the software also handles image streaming. This is done by starting a TCP server which the PC software can connect. For a live update, a small-sized image is streamed to the PC. If the PC software requests a recording the latest three images are sent in full quality. Further, image capturing is also possible when no PC connection is possible, for example, during driving. For this case a temporary on device capturing mode is implemented. The images are stored locally on the device. When it is connected to PC again, all captured images can be transferred to the PC.

PC Software

In Figure 3.22 a screenshot of the PC capturing software is shown. On the left side, the buttons for starting a live stream, image capturing and initiated a temporary on-device image capturing are placed. The three images in the middle show the latest images saved in full quality. Further right, the small live stream image is shown. Additionally, the cameras can be reinitialized via the re-init cameras button on the lower left side.



status: no status

/home/david/saveimages/test_1
/home/david/saveimages/test_2
/home/david/saveimages/test_3
/home/david/saveimages/test_4
/home/david/saveimages/test_5

/home/david/saveimages/test_3

Figure 3.22: Image capturing PC software.

3.7.2 Data Preparation

After image capturing the image processing also as the NIR image projection (3.5) and the labelling has to be done. To handle and provide a good overview of this process, an individual image viewer as also a labelling software was implemented and is explained in the following two subsections. Additionally, a train, classify, and evaluation program was implemented, which is described in more detail in Section 3.7.3.

Image Viewer

The image viewer provides an overview over the currently loaded image set, the related rectified images, the disparity map, and overlay view from the NIR image and the RGB image and a 3D point cloud view of the disparity values (3.23).

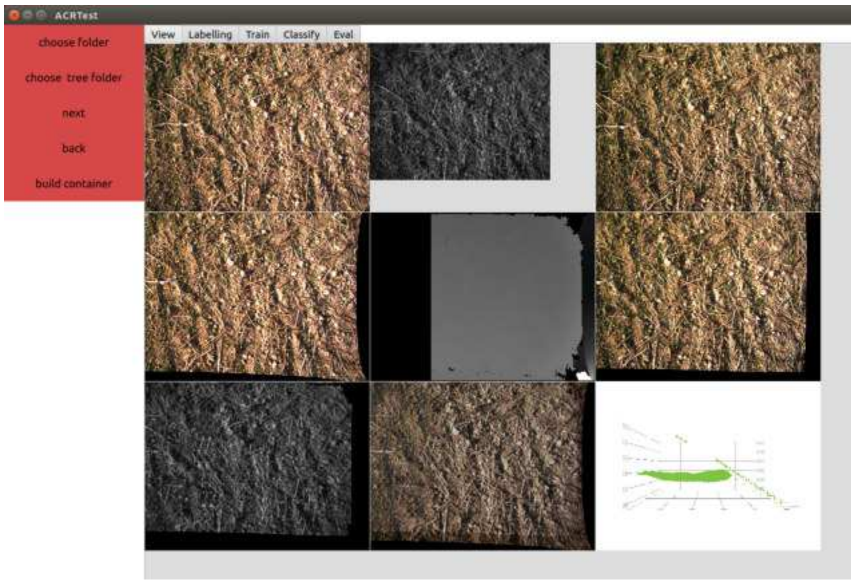


Figure 3.23: Image Viewer

With the help of the image viewer, it is possible to adjust the stereo parameters and the projection parameters. Each tuning effect of these parameters can be verified directly in this view. Because the complete process is susceptible to small changes, it is constructive to get an overview of the data preparation effects.

Labelling

The labelling is implemented as a semi-automatic pre-segmentation method as it is shown in Figure 3.24. With this implementation, it is possible to paint the specific parts of the image with the corresponding label with a mouse click on the current pixel. In the background, the pre-segmentation algorithm automatically detects corresponding pixel clusters and fills them with the selected label into the label image, which overlays the original image. Considering this method, a fast and efficient semi-automatic image labelling can be done.

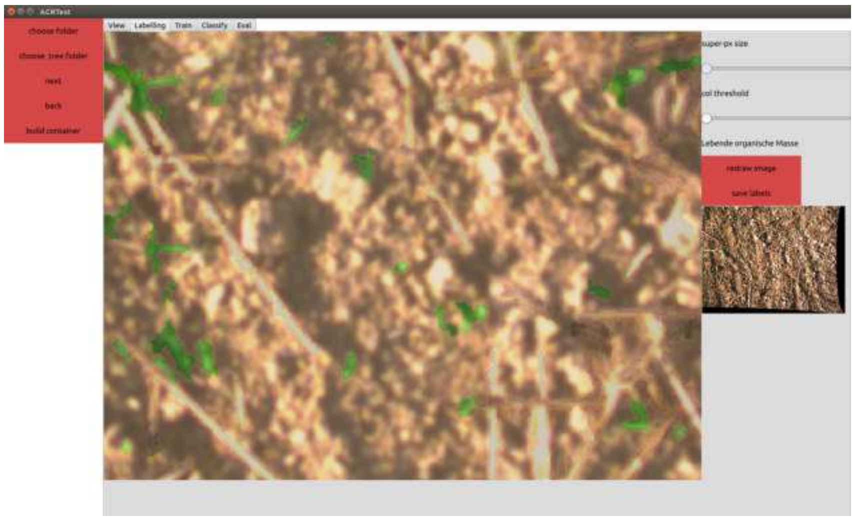


Figure 3.24: Image Labelling Software

3.7.3 Algorithm Implementation

The algorithm consists of three main parts, namely the training, classification and evaluation part. These three parts are working with the same base implementation, which are described in Section 3.7.3. The special properties in terms of implementation and efficient computation are discussed in the following section.

Training, Classification and Evaluation

The implementation of the training and classification is very similar and follows the basic algorithm described in Section 2.1.3. To decrease the computational time, the algorithm is implemented as a multi-threading system. Thus for each node, the complete data container which holds all the image channels of the complete dataset is evaluated against the actual feature. After splitting the

dataset, the next level of the trees is computed in separated threads for the left and right child node until the cancellation criteria reached. Additionally, loop parallelism is implemented thus it is possible to calculate several feature threshold combinations parallel during training. Based on this efficient parallel implementation approach, the tree is growing unbalanced. The classification uses the same underlying implementation, but in fact, only the trained feature is evaluated.
That the unbalanced occurrences of the labels in the training dataset can be regularized a level weighting is introduced which provides the system from disregarding infrequent labels.
The evaluation algorithm takes the classified and the ground truth label images as input. Under the used performance measure metric, which is explained in Section 4.2, the result is calculated pixel-wise.

Hyperparameters

The RF algorithm offers much flexibility in term of adjustable parameters. They can be grouped into two main categories, the algorithm specific and feature specific parameters. To have adjustable parameters can be very helpful because then it is possible to tune the algorithm especially for the needed use case but on the other hand, it is hard to find the right ones the achieve the best results. The parameter evaluation of our approach is discussed in Section 4.3. The properties of both parameter groups are described in more detail in the following sections.

Feature Specific Parameters

As it is described in Section 3.6.2 all the features are implemented over a region. The size of these regions is known as the window size. Additionally, these regions can be shifted by an Δ offset from the currently viewed pixel.

Window Size

The window size defines the size of the region which is used for calculation of the average or the variance. This parameter is sensitive to the features and the resolution included in the image. For images with a high resolution, the window size can be chosen smaller than for lower resolution images because the image with a higher resolution holds more information within a smaller region compared to lower resolution images.

Pixel Offset

Every image has based on the resolution and the feature density another distance between the different objects. The offset which is used for differential information extraction should not be so large that the viewed region is in another object than the directly neighbouring.

Algorithm Specific Parameters

The algorithm-specific parameters can be split into termination criteria parameters and robustness parameters. They directly influence the behaviour of the algorithm. The eight main algorithm parameters are described in the subsections below.

Minimum Information Gain

With the minimum information gain, the propagation of the tree can be stopped because if the information content is low for a further split, it has no positive effect on the performance of the algorithm.

Minimum Number of Points for Splitting

The minimum number of points for a split is also a propagation termination criteria. From a certain number of nodes in a tree node, it would gain no improvement to continue with splitting.

Number of Features

The number of features tried in each node influences the performance of the algorithm. A high number of features results in more accurate results, but it also increases the computational complexity. Thus there has to be found a compromise between accuracy and computational intensity.

Number of Thresholds

The number of tried thresholds has a similar behaviour as the number of tried features. There also has to be found a compromise between accuracy and computational intensity. The thresholds are tried after the feature is fixed for the current tree node. Thus it is an improvement of the feature with the temporary best information gain.

Image Subset Size

For robustness, not the complete image set is chosen for training. Every tree does only take a subset of the available training images. Thus it is guaranteed that each tree does not have the same information for training, and this makes the forest more robust against changes in the classification dataset.

Feature Subset Size

The idea behind the feature subset size is similar than the one behind the image subset length. Using not all features for every tree makes the resulting forest more robust.

Maximum Tree Depth

The tree depth is also a propagation cancellation criteria and is limiting the trees to a maximal depth. It prevents the trees from becoming too fine-granular.

Number of Trees

For robustness, the number of trees is essential because as it is described above the trees are trained with a different image and subset ratio. Thus every tree has another speciality and only with a set of trees it is possible to cover all the scenes and soil compositions with their peculiarities. To find a good number of trees, we tried to estimate with a lightweight estimation described in Section 4.3.2.

4 Experiments

Using additional NIR and depth information is the core of this thesis. Our approach is based on the hypothesis that the additional information source which is sensitive to plants increase the performance of the semantic image segmentation process. Thus it is possible to give a more accurate prediction about soil coverage. To measure the performance improvement, we use the F1 score, which is described in Section 4.2. The training dataset and the evaluation dataset is explained in Section 4.1. In Section 4.3 the evaluation of the hyperparameters is described. The segmentation results of both experiments are described and discussed in Section 4.4.

4.1 Dataset

The training dataset consists of more than 70 images which have been labelled with the method described in Section 3.7.2. To cover a wide range of different scenes, we decided to focus on four main soil coverage compositions which are shown in Figure 4.1. The first composition is shown in sub Figure 4.1a and mainly consists of soil and residues. Figure 4.1b represents scenes, which have a high amount of soil and plants. The third scene (4.1c) covers a mix of soil, plants and residues. The last scene has a high incidence of stones and residues. Additional to the training dataset an evaluation dataset consisting out of 20 labelled images is provided. It also includes the four scenes to make a reliable performance measurement possible.

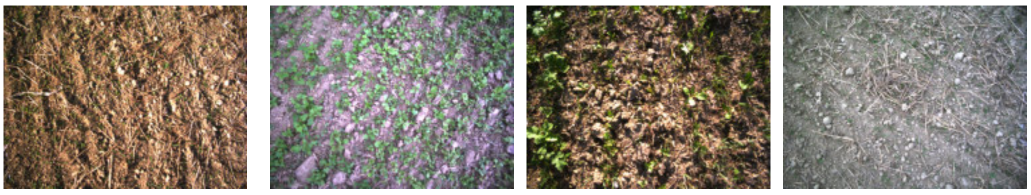

(a) (b) (c) (d)

Figure 4.1: Different soil coverage scenes

The images were captured under conventional environmental conditions with-

out any artificial light. Thus, the results are nearly comparable with the ones in practical usage excepted to air impurities and the impacts of movements. All the images and their labels are listed in the appendix (6).
For the preparation of the datasets, we use image subregions of the size $320x240$ pixel. These subregions are selected in a way that each holds different information about the scenes in terms of surface texture, illumination and quality. A wide field of different subregions is necessary to make the classification more robust, especially when the training dataset is limited. The low resolution of the images decreases the computational complexity of the training as also of the classification but make it harder for the algorithm to perform well. In our approach, we decided to choose the resolution that we get a good compromise between computational complexity and performance.

4.2 Performance Measure

To measure the performance of multi-class classification algorithms, many different methods exist. One frequently used is the F1 score, which also takes the unbalanced occurs of labels into account compared to the accuracy, which gives no reliable results on unbalanced label distributions. The F1 score performance measurement is explained in Section 4.2.1.

4.2.1 F1 Score

The basis of the F1 performance metric is the confusion matrix which describes the true conditions against the actual predicated conditions for each label. Thus it is possible to determine each classified pixel against the ground truth. With the help of the information held by this matrix (4.2.1) the precision and the recall value of each label can be calculated. These two factors are necessary to calculate the F1 score metric, which is explained above [34].

	Label A	Label B	Label C	
Predicted Label A	30	20	10	Total predicted label A
Predicted Label B	50	60	10	Total predicted label B
Predicted Label C	20	20	80	Total predicted label C
	Total label A	Total label B	Total label C	

Table 4.1: Confusion matrix with sample values

The diagonal of the confusion matrix contains the true positives for each label. Summing up the columns results in the number of pixels which should have the specific label and the sum of rows is the number of pixels which were predicted for the specific label. Thus the precision, as an indicator for positive predicated pixels, can be calculated as it is defined through equation 4.1.

$$precision(X) = \frac{true\ positives\ label\ X}{total\ predicted\ label\ X} \tag{4.1}$$

The recall value, which holds the sensitivity information of each label can be calculated as it is shown by equation 4.2.

$$recall(X) = \frac{true\ positives\ label\ X}{total\ label\ X} \tag{4.2}$$

Finally, the F1 score performance metric is defined through twice of the multiple of precision and recall divided by the sum of these values 4.3.

$$F1\ score(X) = 2 * \frac{precision(X) * recall(X)}{precision(X) + recall(X)} \tag{4.3}$$

4.3 Hyperparameter Estimation

For our approach, we did a lightweight parameter estimation based on fast and efficient evaluation progress. As initial parameter settings, we took the ones from the paper [17] and adapted them for our particular case. It turned out that some parameters have more influence on the results than others. Thus it was possible the assume some of them in our lightweight hyperparameter tuning process (4.3.1). A good number of trees is evaluated based on the performance result described in Section 4.3.2. Additional a sufficient analysis of window size, pixel offset and minimum information gain was done as it is explained in Section 4.3.3.

4.3.1 Adopted Parameters

The parameters explained in the following section are adapted from the paper [17] and were not explicitly evaluated based on our approach of developing the system in a fast interference.

Minimum Number of Points for Splitting

The minimum number of points for a further split is set to five. This assumption is based on the experience we made during training and the observation of the most common termination criteria. The number of minimum points for splitting is scarce in terms of termination in comparison to the information gain.

Image Subset Size

The image subset size is set to an image ratio of 0.75. With this ratio, the trees are individual but also sufficient enough intersection, thus that they can be used for a meaningful voting process.

Feature Subset Size

The feature subset size is set to a ratio of 0.9. This feature restriction also results in higher robustness because of the individuality of each tree.

Number of Features

In our approach 32 features are implemented. After the limitation of the feature subset length, every tree gets a set of 29 features to evaluate. Expecting a feature threshold scaling from 100 thresholds per feature results in 2900 possible feature threshold combinations. Assuming that the window size and the pixel offset is also chosen randomly in a specific range, we decided to evaluate 4000 feature threshold combinations for each tree node.

Number of Thresholds

To optimize the node-specific feature, 4000 randomly chosen thresholds are tried because a good value for the threshold has a massive impact on the segmentation performance.

Maximum Tree Depth

Also, the minimum number of points for splitting, the maximum tree depth is a termination criterion which is compared to the information gain very rare efficient in our approach. Thus we are chose 25 for the maximum tree depth to guarantee that the tree depth has not much impact on the results and a too huge tree growing is avoided.

4.3.2 Number of Trees

The number of trees is determined through a performance measurement of several different tree numbers as it is shown in the chart 4.2. The quality of classification increases with the chosen number of trees. Between the usage of 2 trees and 6 trees, the most noticeable improvement is recognizable, but from 6 to 10 trees also improvement is visible. Thus we chose 10 trees for the forest because the results are good enough, and the increase of the performance for a remarkable tree number does not justify to the increasing computational complexity.

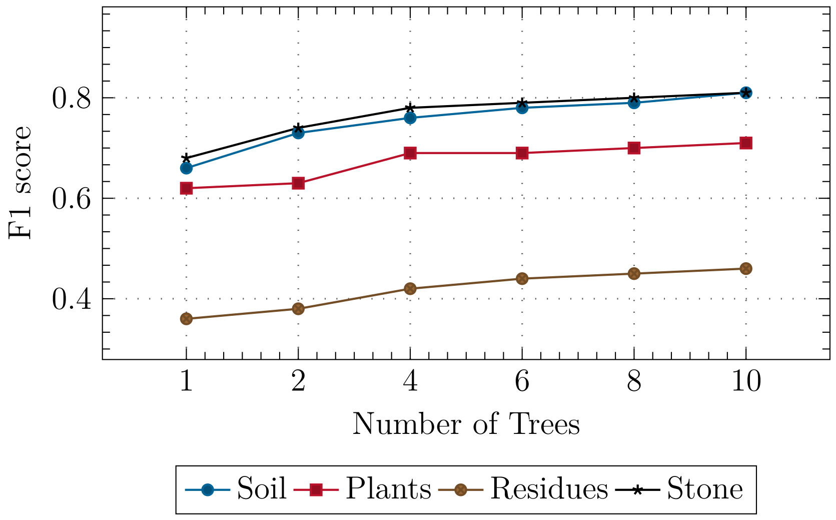


Figure 4.2: Number of trees estimation

4.3.3 Window Size, Pixel Offset and Minimum Information Gain

To evaluate the minimum acceptable information gain, several different values in the range from 0.5 to 0.01 were evaluated. Based on these experiments it has shown that for our training dataset which includes many different soil compositions in a high density the information gain has to be very low because it is not possible to split the dataset with a relative high information gain. Thus we decided to use 0.01 for the minimum information gain need for a further split. A larger dataset could avoid this limitation.
Based on our image quality and object density, window size and pixel tuning was done. Regions from 8 to 20 pixels in the square were also evaluated as pixel offsets from 7 to 15 pixels. The results showed that regions from the size of 20 pixels in square and an offset from 15-pixel yield to the best performance.

4.4 Classification Results

To evaluate the influence of the additional NIR information on the segmentation process, two experiments were done with the hyperparameters described above. First, a segmentation with additional NIR information against the evaluation dataset was done and second an evaluation without this additional information source was done (4.4.2). The results are described in detail in the following two Subsections 4.4.1 and 4.4.2. In Section 4.5 a comparison of the results takes place.

4.4.1 Classification with NIR Information

The following four figures are showing the segmentation results individually for the four trained classes measured with the F1 score metric against the evaluation dataset. Figure 4.3 shows a sample classification result from a scene, including all classes.

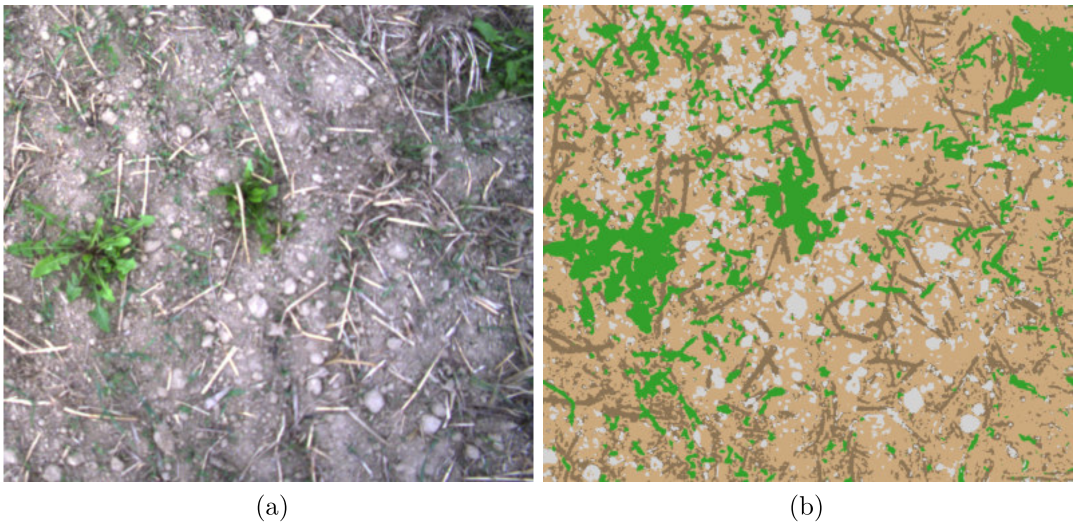


Figure 4.3: Classification result including additional NIR information.

Figure 4.4 show the result for the soil classification accuracy, which has an average F1 score of 0.81. Based on the hard differentiation between soil and residues and the variance of the labelling process, which is nearly in the same range, the result is sufficient.

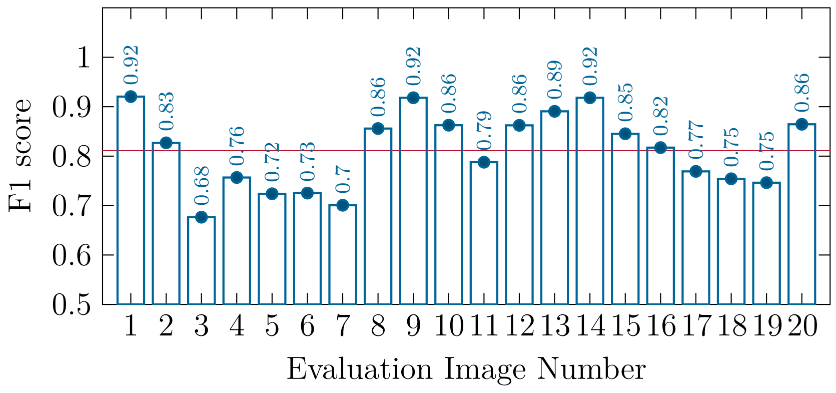


Figure 4.4: Soil detection accuracy with NIR information

With an average F1 score of 0.71 the plant classification (4.5) also representing meaningful results. Especially the edges of the plants mostly interpret larger then they are. This deviation from the ground truth labelling is responsible for a slightly lower value.

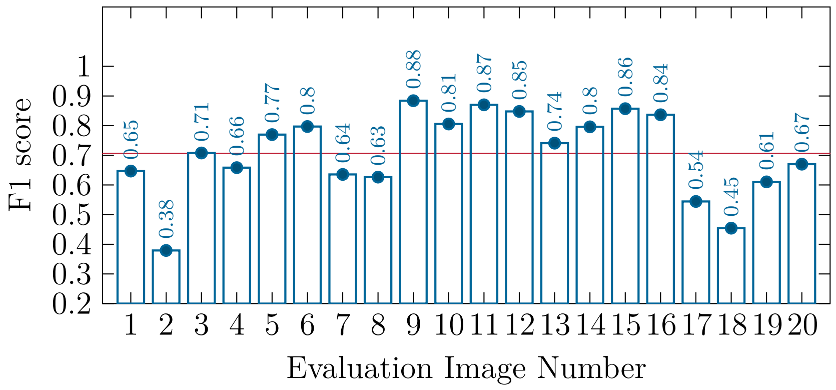


Figure 4.5: Plants detection accuracy with NIR information

The classification of the residues proportion turned out to be not as manifestly the other classes because of the difficult differentiations of residues in the images from the other classes, especially in dark and edge regions. Thus the classification reached an average F1 score from 0.46.

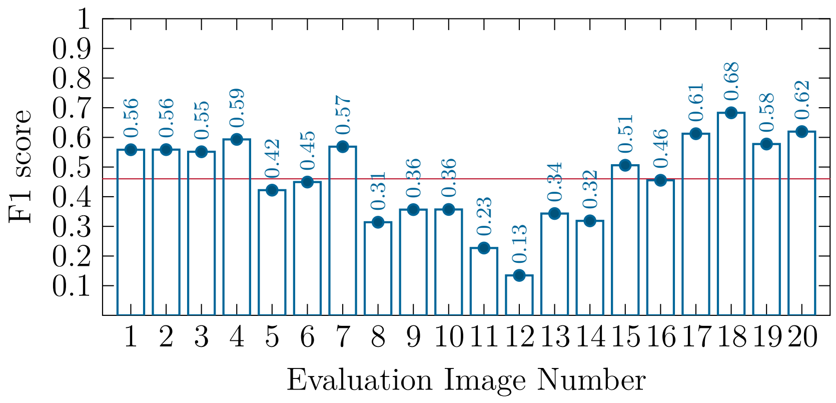


Figure 4.6: Residues detection accuracy with NIR information

The stone detection accuracy reached the second highest average F1 score from 0.8 compared to the other classes. This result can be explained by the fact that some of the evaluation images do not include stones, and thus these specific images reached an F1 score from 1.

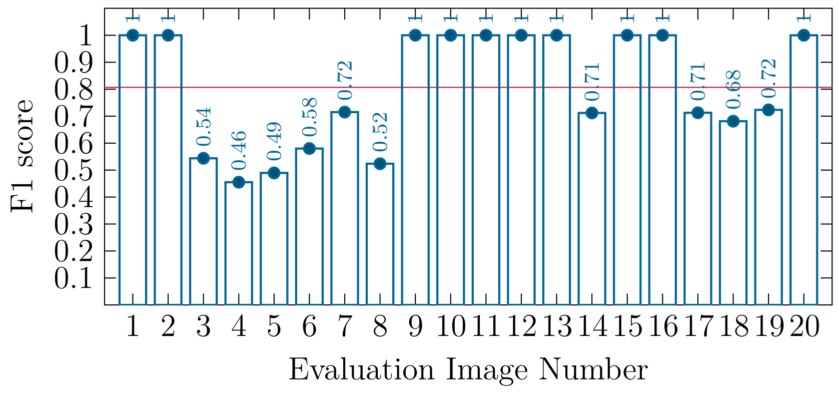


Figure 4.7: Stone detection accuracy with NIR information

4.4.2 Classification without NIR Information

The second experiment focused on the classification with out NIR information. An example classification result is shown in Figure 4.8. The result of the four different classes is discussed based on the following four charts.

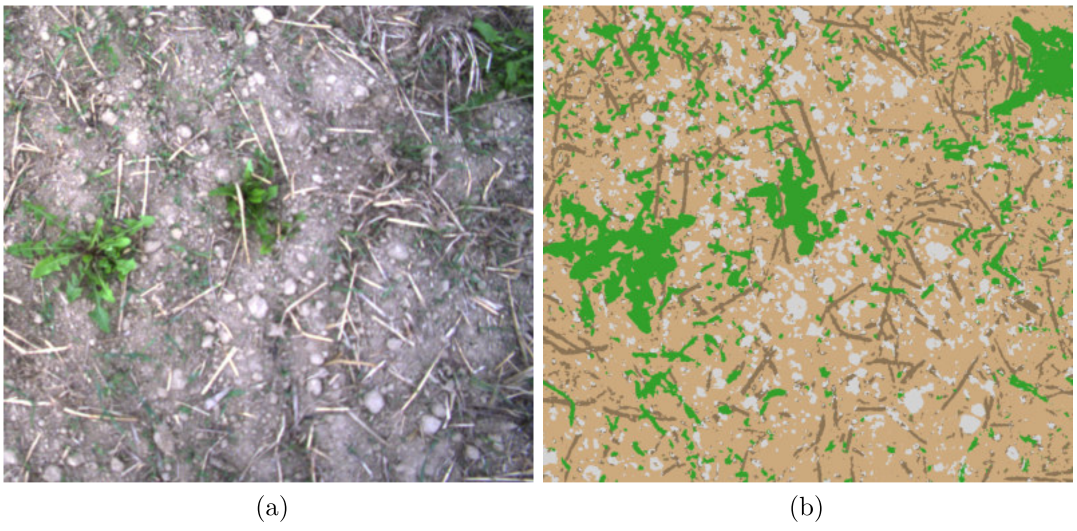


Figure 4.8: Classification result excluding additional NIR information.

An average F1 score from 0.82 rates the accuracy for the estimated soil coverage. As the results with NIR information, this value is sufficient for soil cover estimation based on the problematic datasets and the laborious labelling process which is also a challenge for a human eye based on the rare image quality and the low resolution. To train and evaluate an image segmentation algorithm with this hard environmental conditions can also be viewed as an advantage because it is an indicator for robustness.

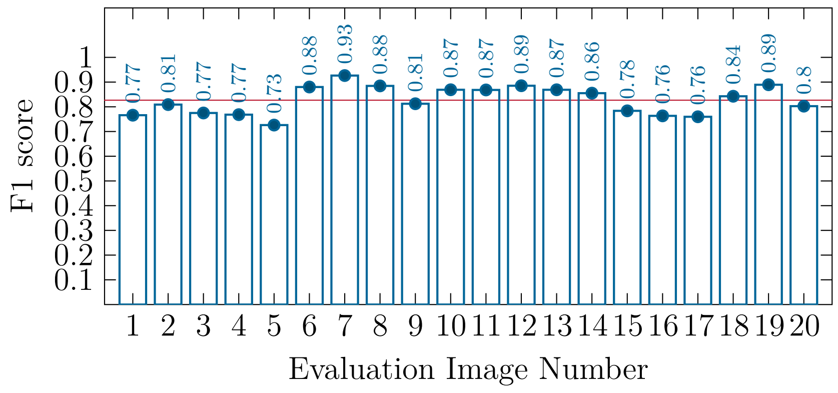


Figure 4.9: Soil detection accuracy without NIR information

For the living organic plants, the classification without NIR information reaches an average F1 score from 0.71 and is thus comparable with the results without NIR information. As the experiment with NIR, the lower score can be lead back to the same arguments in terms of edge inaccuracies.

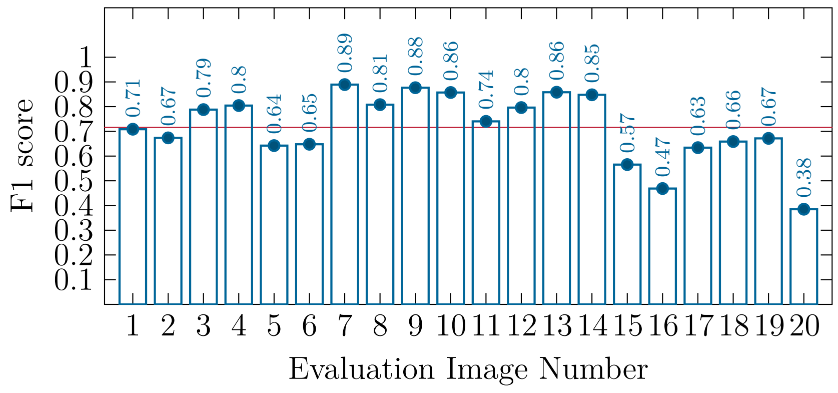


Figure 4.10: Plants detection accuracy without NIR information

The classification results for residues is not as good as the one with NIR information and reaching an average F1 score from 0.41. The generally lower values for residues are based on the fact that residues are often harsh to distinguish from the soil. Especially under these conditions, the labelling process of the evaluation and training dataset also has to be afflicted by an error, which directly leads to significant errors during training and classification.

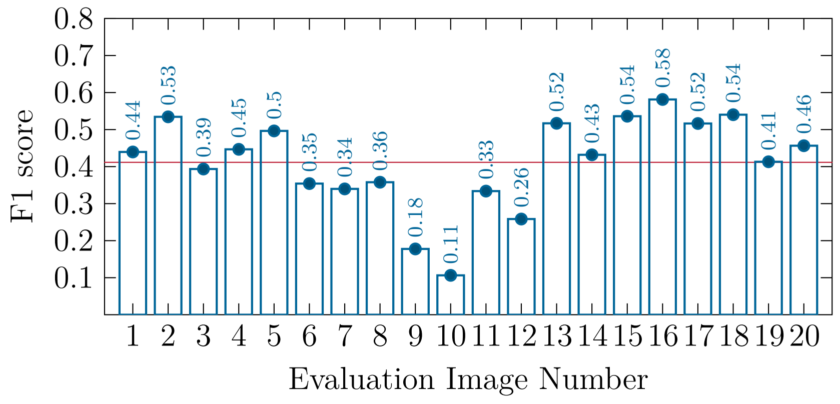


Figure 4.11: Residues detection accuracy without NIR information

For stones, the results are showing an average F1 score from 0.78. This high value also based on the evaluation images which are including no stones and thus they reached an accuracy from 1.

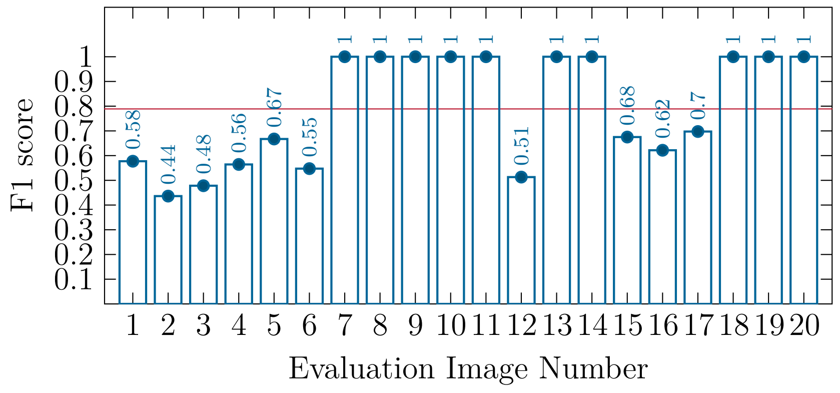


Figure 4.12: Stone detection accuracy without NIR information

In the following Section 4.5 the classification results from the two experiments are compared and discussed.

4.5 Comparison

In this section, the two experiments are compared and briefly discussed for each class in the following subsections to give a short summary of the results.

Soil

The F1 scores for soil are nearly the same. This similarity is the result which we expected because the additional NIR information has not much influence on soil classification. However, all in all, the results are suitable for both experiments under the constraint that there is also a variance on the label images which are used for training and evaluation.

Plants

As the soil classification, the segmentation results for plants are also very similar. This result was not expected because the NIR information should gain, especially in the segmentation of plants a considerable benefit. We suppose that the result is impaired by the short dataset and the significant variance during labelling. However, with an F1 score of 0.71, the results are precise enough for predictions in terms of soil coverage estimation and protecting soil from erosion.

Residues

Because the estimation of the residues proportion is very hard, based on the not clear demarcation to the soil, the F1 score is both experiments not high. We can see that the results with NIR information are reaching a higher F1 score by 0.5. This increase is an effect based on the additional NIR information, which not only extracts plants better from the soil but also the residues.

Stone

Stones are rated in both experiments very well. These results are good because some of the evaluation images do not include stones at all. Thus they are reaching an F1 score from 1 and increasing the average score. Without these images, the results would not be as good as they are. For both experiments, the results are in a range from 0.78 to 0.8. Also, as residues, stones are hard to extract from the underlying soil, especially when it is not clear if the stones are lying on soil or fine granular rock underground. That the results are nearly in the same range was expected because additional NIR information has not much impact on stone segmentation.

5 Conclusion and Future Work

This chapter summarizes the findings of soil cover estimation without NIR information and with additional NIR information. The main goal and the achieved results from this work also as the gained knowledge is summarized below. Improvements which could be done in future work are to optimize the results and bring the system to a state, that the camera box can be used in practical applications are given in Section 5.2.

5.1 Conclusion

Estimating the differences between soil cover estimation with NIR information and without was the primary goal of this work. A further aim was to build a camera box for image capturing. To achieve this goal, an autonomous camera box was build to capture stereo image pairs also as the corresponding NIR image. With these three images, a projection of the NIR image to the RGB colour image could be done. In a further step, these image pairs were labelled manually to get an training dataset as also an evaluation dataset. The training dataset was used to train a RF with NIR and another one without NIR information. Finally, both RFs were evaluated against the same evaluation training dataset set to get a meaningful comparison between the two approaches.
We showed that the results are significantly suitable for both methods under the constraint of the labelling variance and the problematic environmental conditions which lead to lower image quality and hard separable classes. For the residues class, we showed that the influences of additional NIR information performed significantly better by an F1 score difference from 0.5. This apparent difference yields from the hard separability of the residues class and the other classes in RGB images. In the NIR images residues are better separatable and thus easier to classify. The other classes performed very similarly. This effect can be traced back to small training dataset and labelling variance. The hyperparameter estimation is also a point which could be optimised to achieve a more precise differentiation between the approaches.
Additionally to the semantic image segmentation the 3D printed camera box was built and used for image capturing under nearly real conditions. Further, a whole software suite for image capturing, streaming, receiving, labelling,

training and classification was written to achieve the proposed goal.
A short overview of future work which could be done to improve the results and bring the whole system to a state where it can be used in real practical applications is given in Section 5.2.

5.2 Future Work

The classification results proposed in this work are sufficient for soil cover estimation and thus also for the application fields which could benefit from these results. Netherless there are some improvements which could be done in future work. First, the segmentation performance could be increased by several different approaches which are described lower. The second improvement is the calculation time and the hardware integration.
To increase the segmentation performance, the datasets for training and evaluation has to become larger and include more meaningful scenes. Thus the algorithm could be trained more robust and the NIR information would possibly get more effective on the results. Additionally, the image quality and capture process could be optimized. Thus, the labelling process could be done with a smaller variance what would possible lead to better-trained trees and more accurate classification results.
Another improvement which would be possible is to decrease the classification time by decreasing the trees depth, feature calculation time and data preparation calculation time. One attempt would be to implement the whole calculations directly in hardware. Another attempt would be to optimize the calculation parallelism. Thus, it would be possible to do the calculations on a microcontroller.
With all the improvements described above, it would be possible to get a tool to prevent soil from erosion, which could be directly added to farming machinery.

6 Appendix

In the following two sub sections the training dataset (6.1) and the evaluation dataset (6.2) is shown.

6.1 Training Data

The training dataset shown below includes all the colour image sub regions as also the ground truth labelled images which were used to train the algorithms.

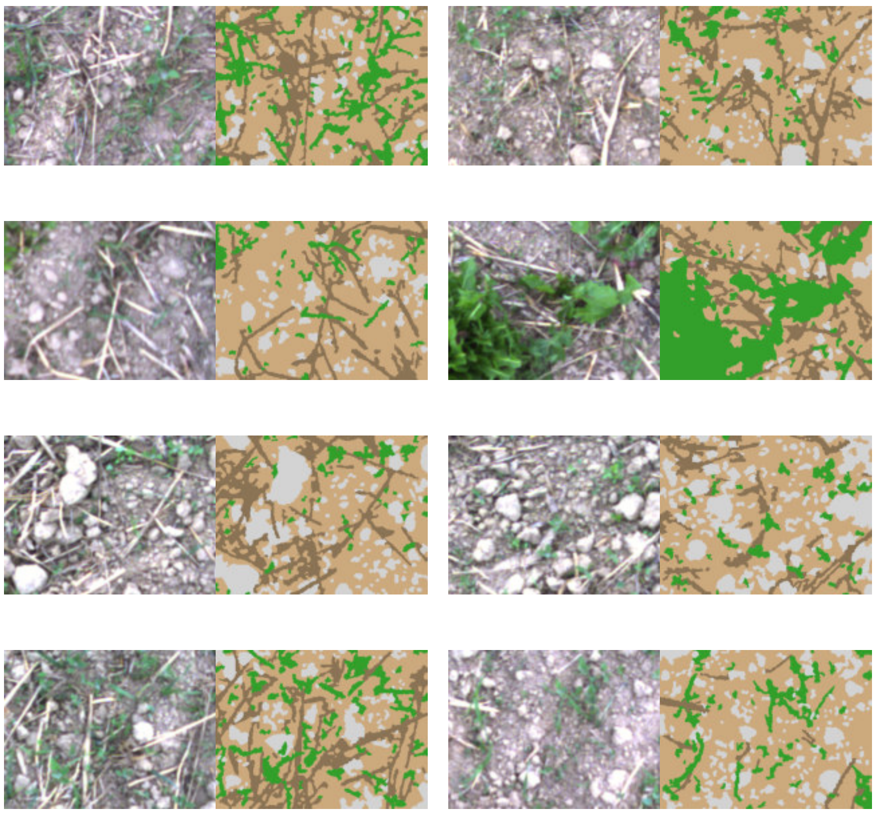


Figure 6.1: Training dataset with the color image and the label image.

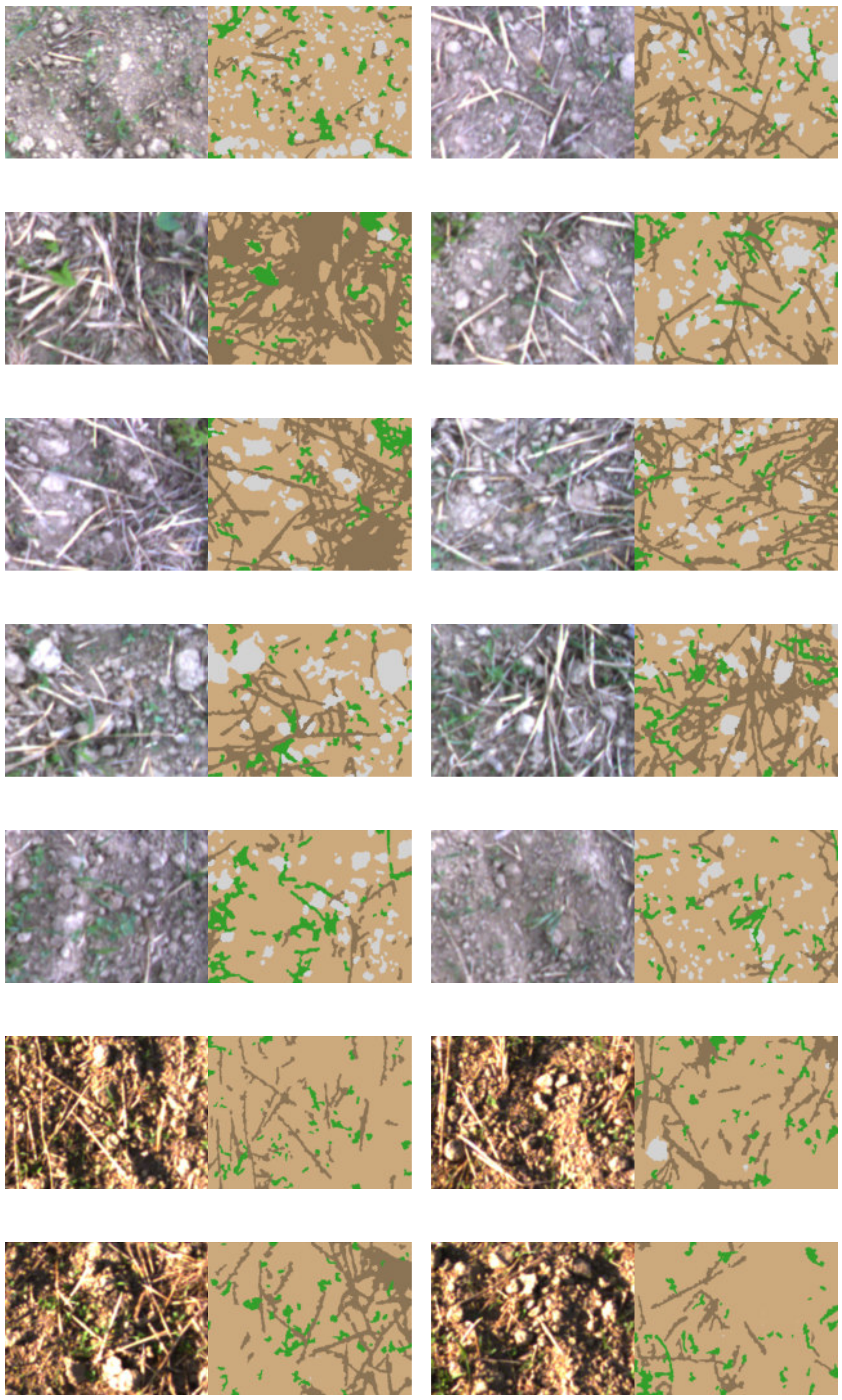


Figure 6.1: Training dataset with the color image and the label image.

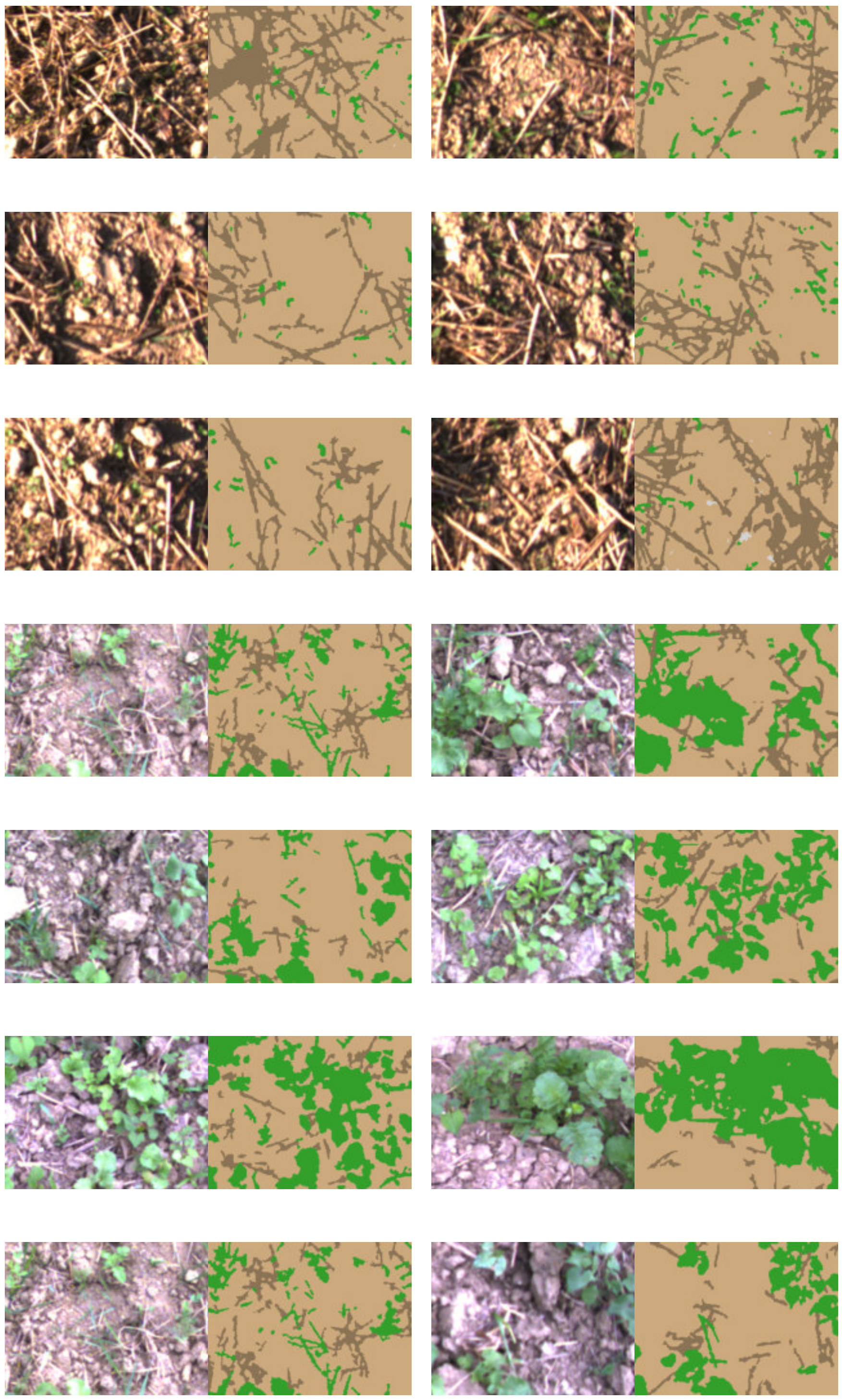


Figure 6.1: Training dataset with the color image and the label image.

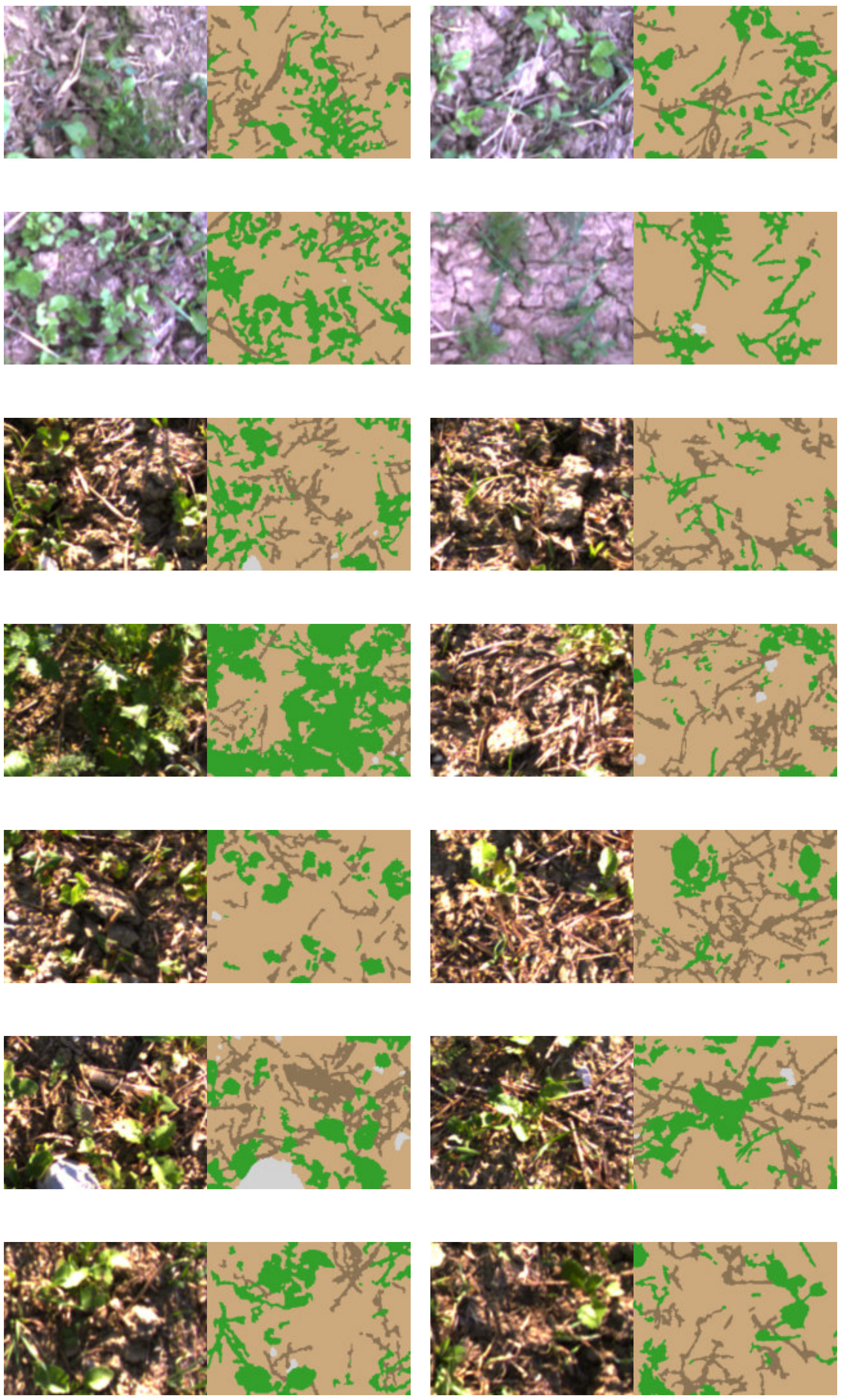


Figure 6.1: Training dataset with the color image and the label image.

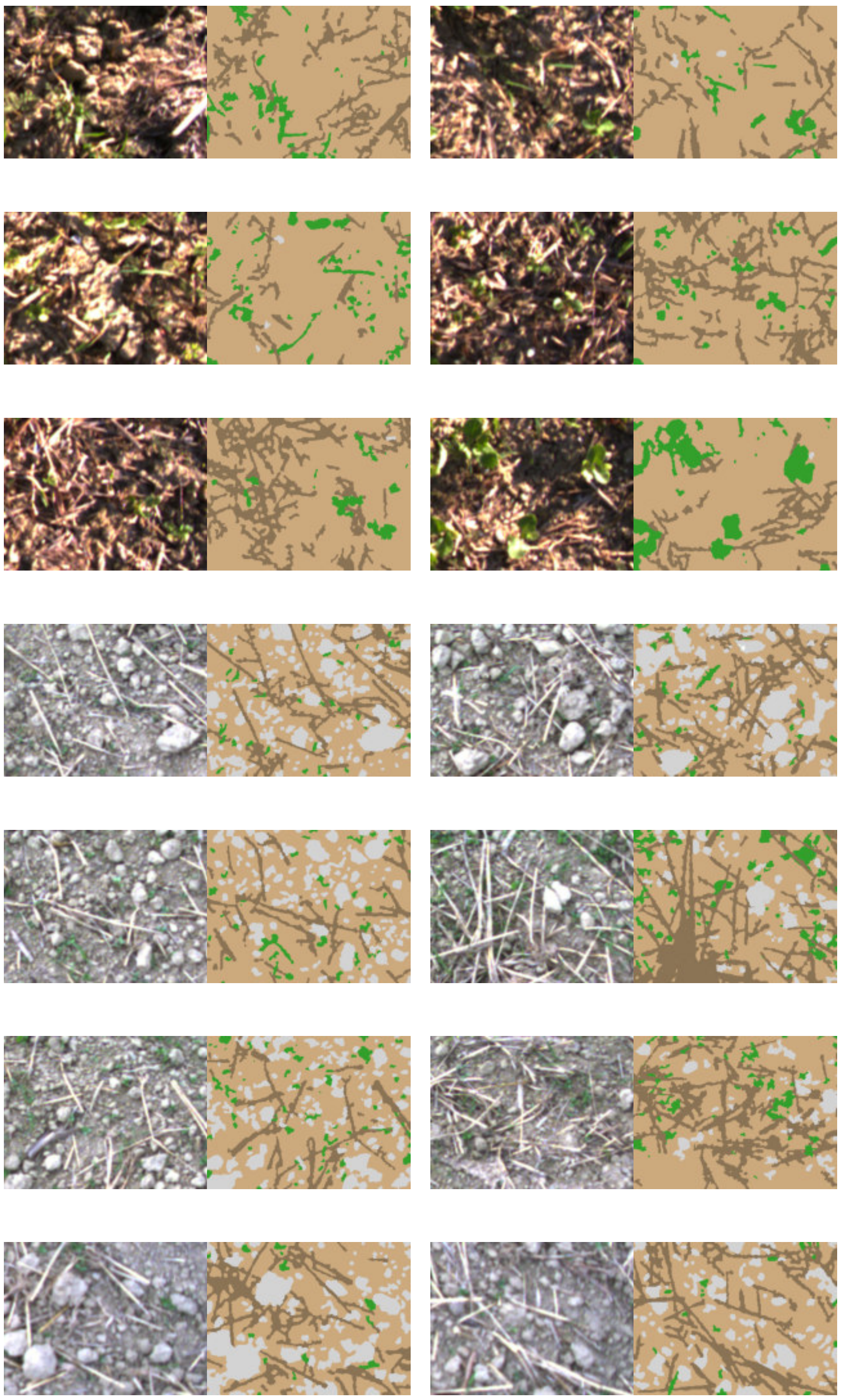


Figure 6.1: Training dataset with the color image and the label image.

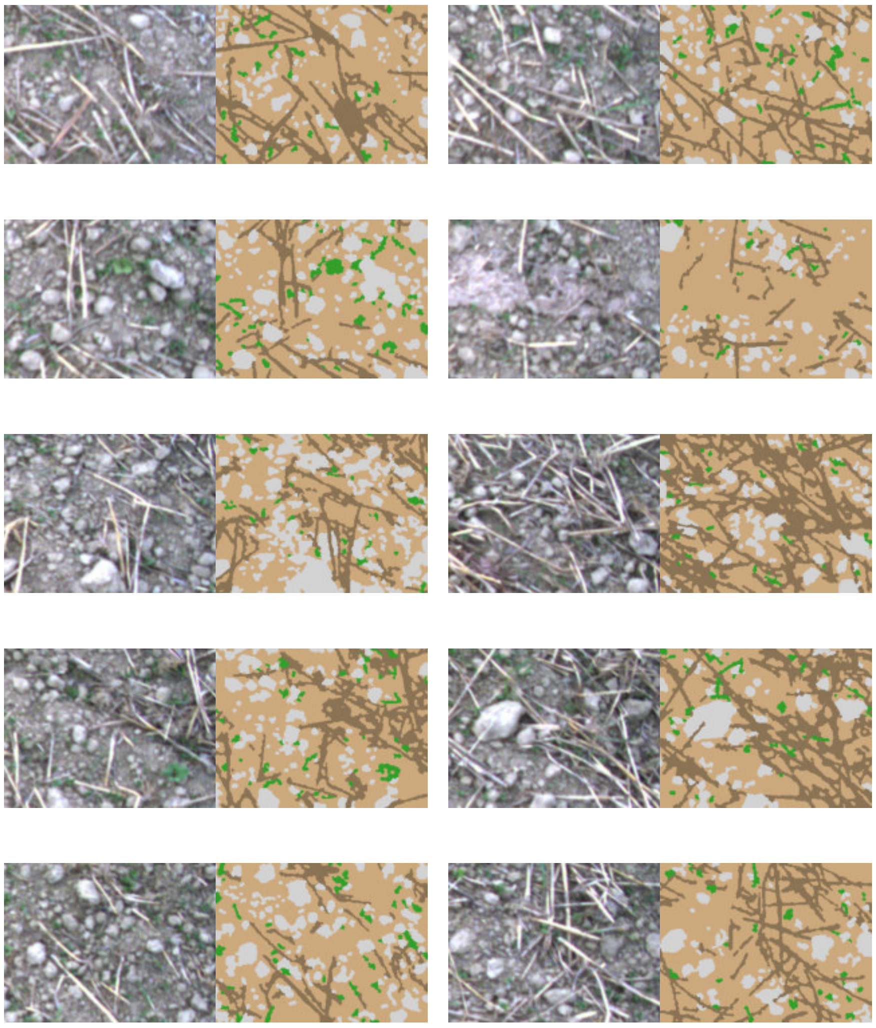


Figure 6.1: Training dataset with the color image and the label image.

6.2 Evaluation Data

The evaluation data set shown in figure 6.2 consists out of the color images, the ground truth labelled image and image classified by the algorithm.

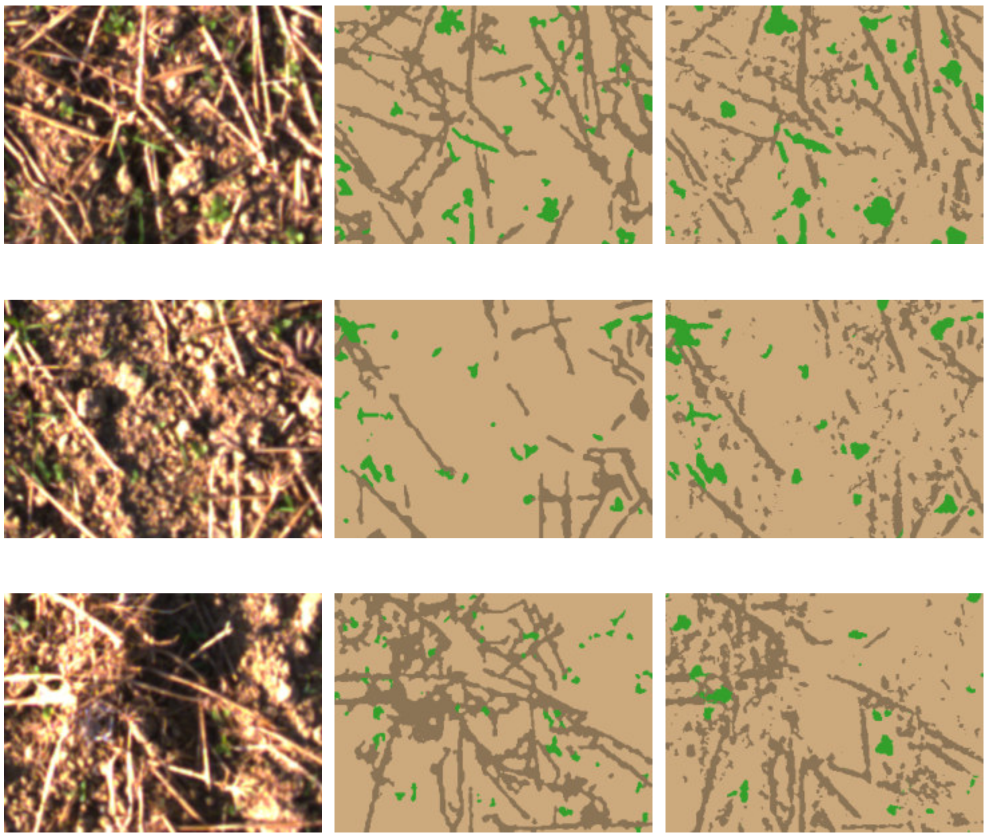


Figure 6.2: Evaluation dataset with the color image, labelled image and the classified image.

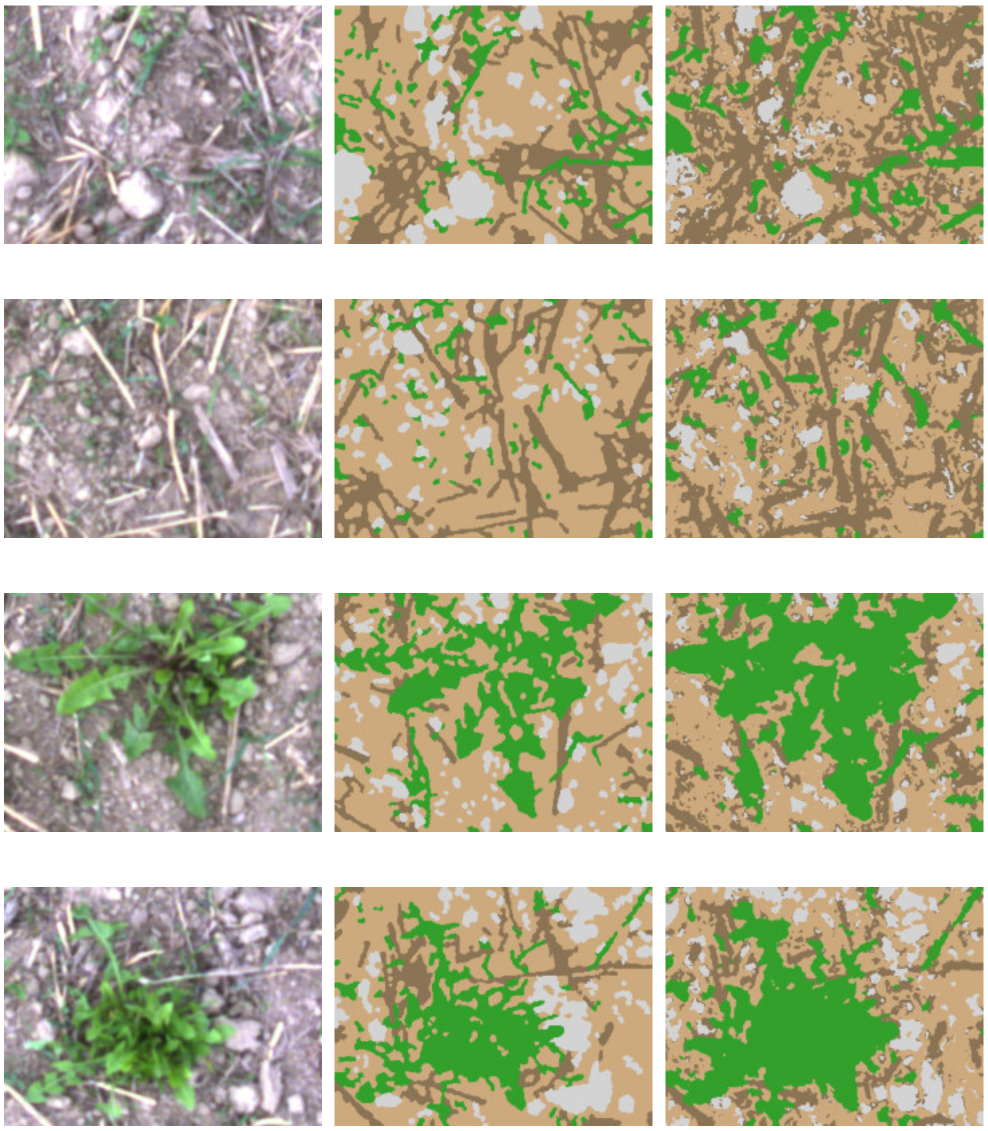


Figure 6.2: Evaluation dataset with the color image, labelled image and the classified image.

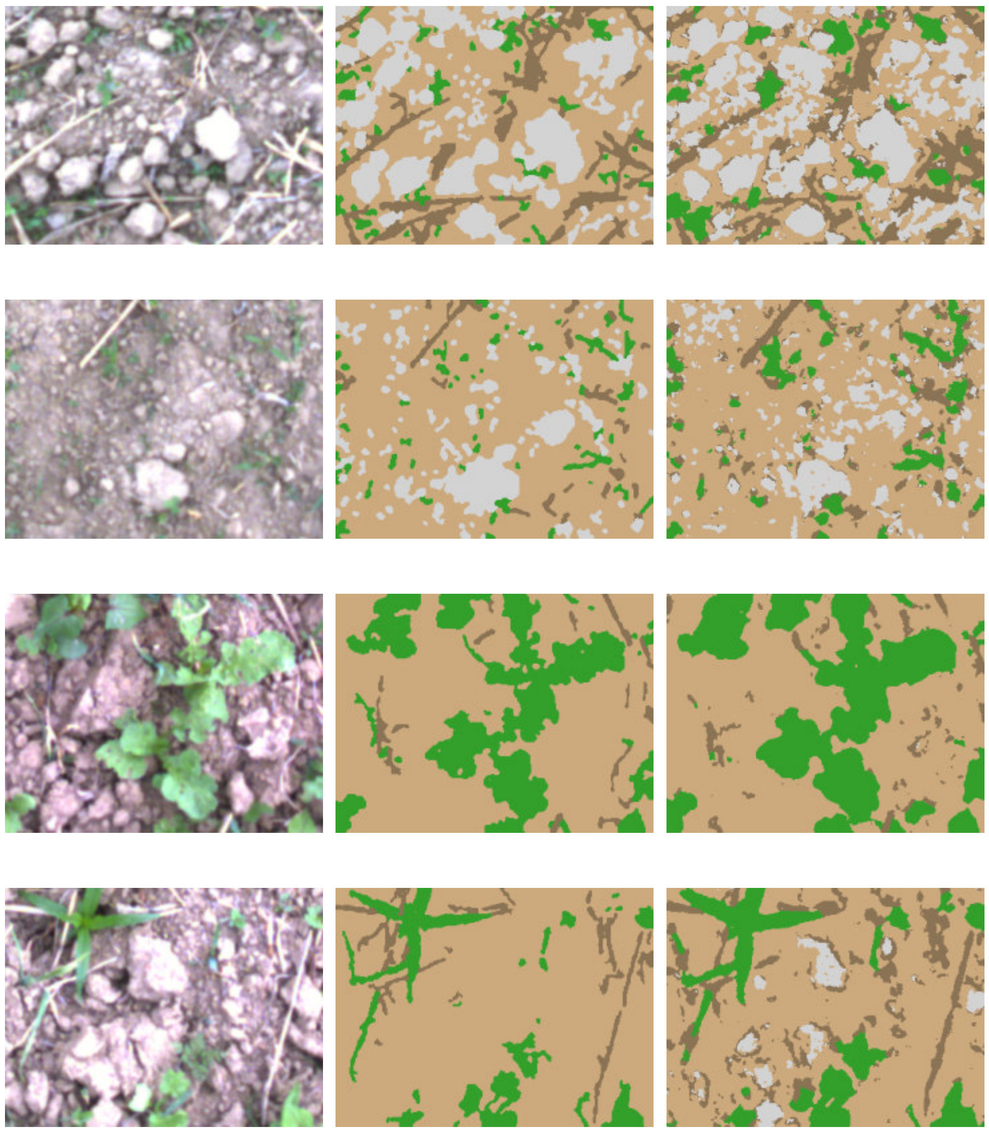


Figure 6.2: Evaluation dataset with the color image, labelled image and the classified image.

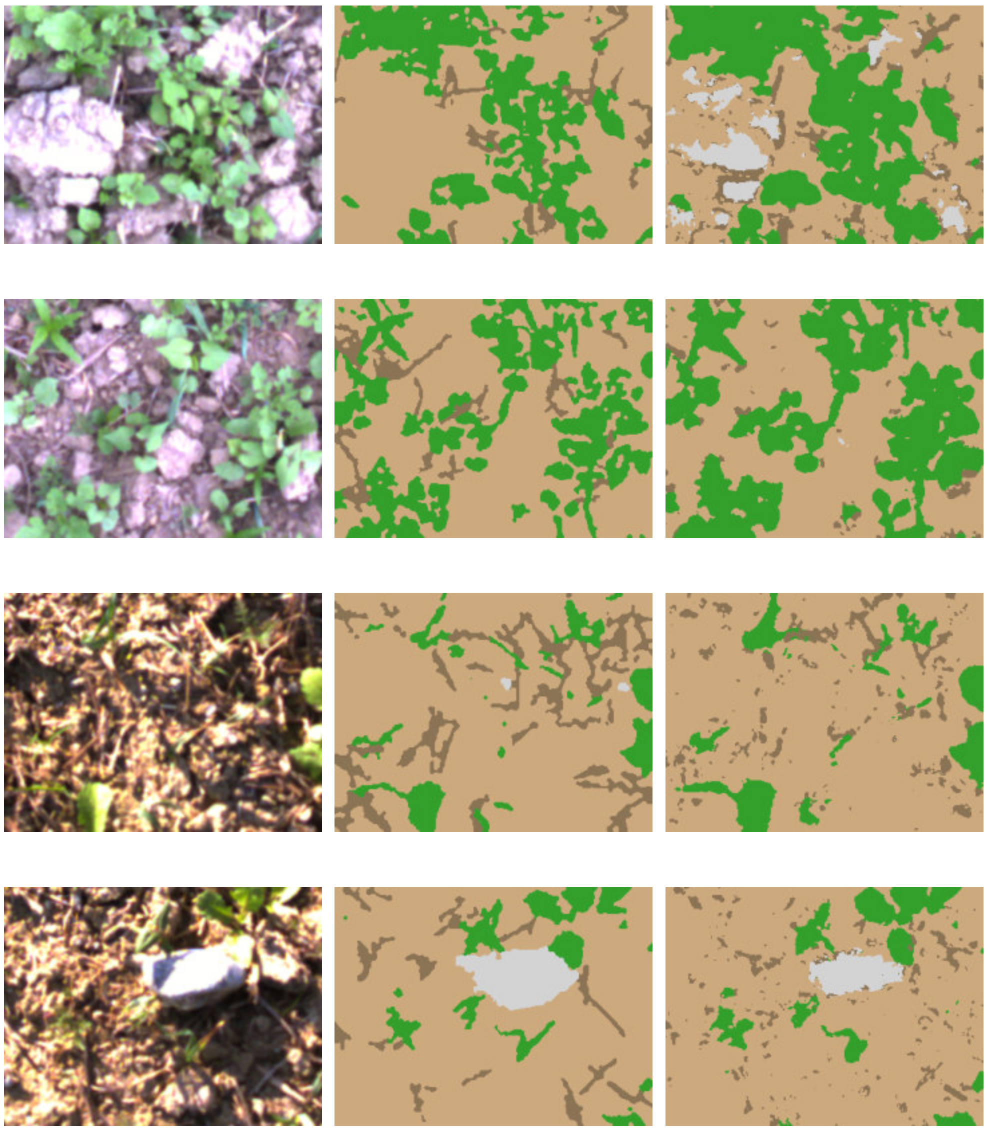


Figure 6.2: Evaluation dataset with the color image, labelled image and the classified image.

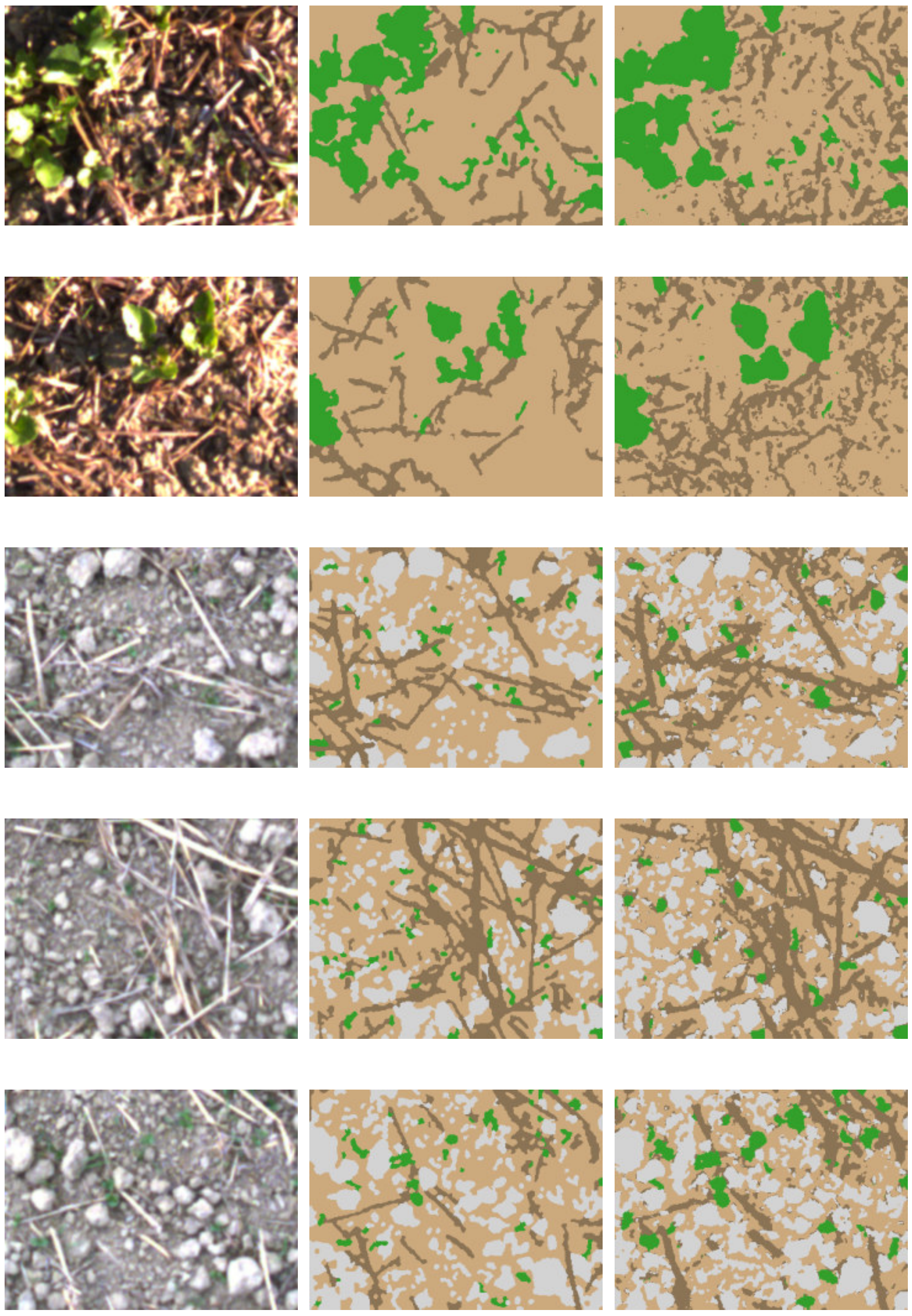


Figure 6.2: Evaluation dataset with the color image, labelled image and the classified image.

Bibliography

[1] J. Canny, „A computational approach to edge detection,“ *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 1986.

[2] X. Liu, Z. Deng, and Y. Yang, „Recent progress in semantic image segmentation,“ *Artificial Intelligence Review*, pp. 1–18, 2018.

[3] P. Panwar and G. Gopal, „Image Segmentation using K-means clustering and Thresholding,“ *Image*, vol. 3, Jan. 2016.

[4] M. A. Hearst, „Support Vector Machines,“ *IEEE Intelligent Systems*, vol. 13, no. 4, pp. 18–28, Jul. 1998, ISSN: 1541-1672. [Online]. Available: http://dx.doi.org/10.1109/5254.708428.

[5] A. E. Ton Wen, „Support Vector Machines Lagrange Multipliers and Simple Volume Decompositions,“ Aug. 2000.

[6] J. Weston and C. Watkins, *Multi-class Support Vector Machines*, 1998.

[7] Z. Kato and T.-C. Pong, „A Markov random field image segmentation model for color textured images,“ *Image and Vision Computing*, vol. 24, no. 10, pp. 1103 –1114, 2006, ISSN: 0262-8856. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0262885606001223.

[8] B. Kroese, B Krose, P. van der Smagt, and P. Smagt, „An introduction to neural networks,“ *J Comput Sci*, vol. 48, Jan. 1993.

[9] S. Albawi, T. A. Mohammed, and S. Al-Zawi, „Understanding of a convolutional neural network,“ in *2017 International Conference on Engineering and Technology (ICET)*, 2017, pp. 1–6.

[10] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders, „Selective Search for Object Recognition,“ *International Journal of Computer Vision*, 2013. [Online]. Available: http://www.huppelen.nl/publications/selectiveSearchDraft.pdf.

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik, „Region-Based Convolutional Networks for Accurate Object Detection and Segmentation,“ *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 1, pp. 142–158, 2016, ISSN: 0162-8828.

[12] E. Shelhamer, J. Long, and T. Darrell, „Fully Convolutional Networks for Semantic Segmentation,“ *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 640–651, 2017. [Online]. Available: `https://doi.org/10.1109/TPAMI.2016.2572683`.

[13] L. Breiman, „Random Forests,“ *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001, ISSN: 1573-0565. [Online]. Available: `"https://doi.org/10.1023/A:1010933404324"`.

[14] J. R. Quinlan, „Induction of Decision Trees,“ *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, Mar. 1986, ISSN: 0885-6125. [Online]. Available: `http://dx.doi.org/10.1023/A:1022643204877`.

[15] P. Refaeilzadeh, L. Tang, and H. Liu, „Cross-Validation.,“ L. Liu and M. T. Ã–zsu, Eds., pp. 532–538, 2009. [Online]. Available: `http://dblp.uni-trier.de/db/reference/db/c.html#RefaeilzadehTL09`.

[16] A. Milioto, P. Lottes, and C. Stachniss, „Real-time Semantic Segmentation of Crop and Weed for Precision Agriculture Robots Leveraging Background Knowledge in CNNs,“ *https://arxiv.org/abs/1709.06764*, Sep. 2017.

[17] R.-N. Peter, P. Johann, B. Thomas, S. Peter, and P. Heinrich, „An Integrated Image Analysis System for the Estimation of Soil Cover,“ Jun. 2016. [Online]. Available: `https://www.josephinum.at/fileadmin/content/BLT/Puplikationen/1568_00_E.pdf`.

[18] X. Liu, S. W. Chen, C. Liu, S. S. Shivakumar, J. Das, C. J. Taylor, J. Underwood, and V. Kumar, „Monocular Camera Based Fruit Counting and Mapping With Semantic Data Association,“ *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2296–2303, 2019, ISSN: 2377-3766.

[19] H. S. Abdullahi, R. E. Sheriff, and F. Mahieddine, „Convolution neural network in precision agriculture for plant image recognition and classification,“ in *2017 Seventh International Conference on Innovative Computing Technology (INTECH)*, 2017, pp. 1–3.

[20] S. Jacquemoud and S. L. Ustin, „Leaf optical properties: A state of the art,“ pp. 223–332, 2001.

[21] P. Mishra, M. S. M. Asaari, A. Herrero-Langreo, S. Lohumi, B. Diezma, and P. Scheunders, „Close range hyperspectral imaging of plants: A review,“ *Biosystems Engineering*, vol. 164, pp. 49 –67, 2017, ISSN: 1537-5110. [Online]. Available: `http://www.sciencedirect.com/science/article/pii/S1537511017302635`.

[22] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002, ISBN: 0130851981.

[23] L. Ma, Y. Chen, and K. L. Moore, „Rational Radial Distortion Models of Camera Lenses with Analytical Solution for Distortion Correction.," *I. J. Information Acquisition*, vol. 1, pp. 135–147, Jun. 2004.

[24] G. Bradski, „The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[25] Y. M. Wang, Y. Li, and J. B. Zheng, „A camera calibration technique based on OpenCV," in *The 3rd International Conference on Information Sciences and Interaction Sciences*, 2010, pp. 403–406.

[26] W. Burger, „Zhang's Camera Calibration Algorithm: In-Depth Tutorial and Implementation," May 2016. [Online]. Available: `https://www.researchgate.net/publication/303233579_Zhang's_Camera_Calibration_Algorithm_In-Depth_Tutorial_and_Implementation`.

[27] Z. Zhang, „A Flexible New Technique for Camera Calibration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 11, pp. 1330–1334, Nov. 2000, ISSN: 0162-8828. [Online]. Available: `http://dx.doi.org/10.1109/34.888718`.

[28] L. Zou and Y. Li, „A method of stereo vision matching based on OpenCV," in *2010 International Conference on Audio, Language and Image Processing*, 2010, pp. 185–190.

[29] K. He, J. Sun, and X. Tang, „Guided Image Filtering," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, pp. 1397–1409, Jun. 2013.

[30] A. Kaur and B. V Kranthi, *Comparison between YCbCr Color Space and CIELab Color Space for Skin Color Segmentation.*

[31] M. D. Steven, „The Sensitivity of the OSAVI Vegetation Index to Observational Parameters," *Remote Sensing of Environment*, vol. 63, no. 1, pp. 49 –60, 1998, ISSN: 0034-4257. [Online]. Available: `http://www.sciencedirect.com/science/article/pii/S0034425797001144`.

[32] *Odroid XU4 Micro Controller Board*, `https://www.hardkernel.com/shop/odroid-xu4-special-price/`, Accessed: 2019-05-08.

[33] *IDS cameras IDS Imaging*, `https://de.ids-imaging.com/home.html`, Accessed: 2019-05-08.

[34] Z. C. Lipton, C. Elkan, and B. Naryanaswamy, „Optimal Thresholding of Classifiers to Maximize F1 Measure," in *Machine Learning and Knowledge Discovery in Databases*, T. Calders, F. Esposito, E. Hüllermeier, and R. Meo, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 225–239, ISBN: 978-3-662-44851-9.

Erklärung

Hiermit erkläre ich, dass die vorliegende Arbeit gemäß dem Code of Conduct, insbesondere ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel, angefertigt wurde. Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.
Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

Vienna, 29th June, 2019

David Martin