USING MACHINE LEARNING TECHNIQUES TO FILTER VEGETATION IN COLORIZED SFM POINT CLOUDS OF SOIL SURFACES

O. Grothum[1*], A. Bienert[1], M. Bluemlein[1], A. Eltner[1]

[1] Institute of Photogrammetry and Remote Sensing, Dresden University of Technology, Germany
Oliver.Grothum1@tu-dresden.de

KEY WORDS: Soil erosion measurement, point cloud processing, vegetation filtering, deep learning, random forest, pointnet++.

ABSTRACT:

Various soil erosion processes can be quantified using digital elevation models (DEMs) of difference. In this study, cameras were used to capture images of bare soil during artificial rainfall simulations. The photos were then used to generate dense 3D point clouds with millimeter resolution using Structure-from-Motion and Multiview-Stereo (SfM+MVS) techniques. However, the point clouds also captured some vegetation, such as agricultural plants, grass, and weeds, present at the soil surface. It had to be removed to accurately measure soil erosion processes. The removal can be done manually and is hence time-consuming. In this study, several methods have been tested and compared to perform (semi-)automatic vegetation filtering from point clouds of soil surfaces. First, the point clouds were labelled into vegetation and ground data to establish a basis set for the subsequent experiments. Then, three branches of algorithms were tested. The first branch considers knowledge-based thresholding. Thereby, unique features were considered, e.g., point color, height, and roughness within a specified neighborhood. For instance, a threshold was set in the color space to separate green vegetation from brown soil considering the green band. The second branch used a machine learning (ML) algorithm to classify each point as vegetation or ground by automatically finding thresholds. Thereby, again features such as point color were used. In addition, multi-scale features were computed for each point to characterize it in the context of its neighborhood. The calculated features were used afterwards with the random forest (RF). The third branch considered an end-to-end learning approach and thus avoiding the necessity to define features. The deep learning-based architecture PointNet++ was used. For the classification, an adapted model for soil surfaces and vegetation was trained. The performance of the different methods was compared, and an assessment of each method was provided. Overall, the study aims to find a (semi-)automatic method to remove vegetation from time series of soil surface point clouds to achieve an accurate measurement of soil surface changes and thus eventually erosion processes while minimizing manual effort and time consumption.

1. INTRODUCTION

In the fields of geomorphic change detection, amongst other systems with multiple cameras are used. Those cameras capture images synchronized over different timespans of months, days or even seconds (Kromer et al., 2019, Blanch et al., 2021), which are processed to digital elevation models (DEMs). To generate those models, Structure-from-Motion and Multi-View Stereo (SfM+MVS) are widely used tools. In the field of soil erosion measurement during artificial rainfall simulations, also DEMs are used, e.g., to parameterize and validate soil erosion models (Hänsel et al., 2016). SfM+MVS is an option to capture with at least one camera the state of the observed soil before and after the rainfall from different perspectives (Eltner et al., 2016, Candido et al., 2020, Epple et al., 2022). To assess the state and development of the erosion during the experiment, time-lapse camera systems are used to generate a time series of DEMs (Eltner et al., 2017). Subsequently, these DEMs are subtracted from each other to map a signed change on each position of the observed plot corresponding to erosion and accumulation if other processes masking erosion processes can be neglected (Kaiser et al., 2018).
For some of those experiments, the influence of vegetation like crops and gras must be considered because plants might be visible in each DEM and thus would distort the results of algorithms to measure soil surface changes (Onnen et al., 2020). To filter vegetation, machine learning (ML) techniques can be reliable algorithms to segment the data into a preset of classes (Martins et al., 2023). This study presents a (semi-)automatic workflow, which trains a classifier on ground truth data, generated for various rainfall simulations, and which was labelled manually. The first classifier is a rule-based algorithm, which separates the green vegetation from the brown soil considering color and a manually extracted vegetation mask. The second classifier is a Random Forest (RF; Breiman, 2001) classifier, which has already been used successfully to classify erosion features (Malinowski et al., 2023). It takes handcrafted features at multiple scales, which are calculated for the training. The third classifier is the PointNet++ (Qi et al., 2017) classifier, which hierarchically samples and groups the point cloud in centroids with given neighborhood to calculate local features at each neighborhood with the PointNet (Qi et al., 2016) structured Multilayer Perceptron (MLP). Each method will be evaluated with a set of labelled test point clouds.
In the following, we present the data acquisition and introduce the rainfall experiments and camera set-up. Then, in section 3, we explain the methods used, which are divided into rule-based, RF and PointNet++ classifications. The obtained results of each classification method are presented and compared in Section 4. Furthermore, the derived results are discussed in Section 5. Finally, concluding remarks and suggestions for future work are made in Section 6.

2. DATA ACQUISTION

2.1 Rainfall experiments

The point clouds, which serve in this study as basis for the vegetation filtering, were calculated from images captured by time-lapse camera systems observing an artificial rainfall

[*] Corresponding author

simulation. The rainfall experiment consisted of two periods with a one-hour break in between to simulate a dry run and a wet run. The setup of the experiment was composed of a device, which irrigated the soil (3 m²) with a preset intensity (Figure 1). The sloped soil was bounded with metal sheets to capture the water running off the soil surface. The runoff was led through a pipe at the bottom of the plot to take water samples and to measure the amount of water. To scale and to reference (transform) each point cloud into the same coordinate system, about ten ground control points were installed around the bounded plot and their coordinates were measured with either total station or measuring tape before and after the rainfall simulation.

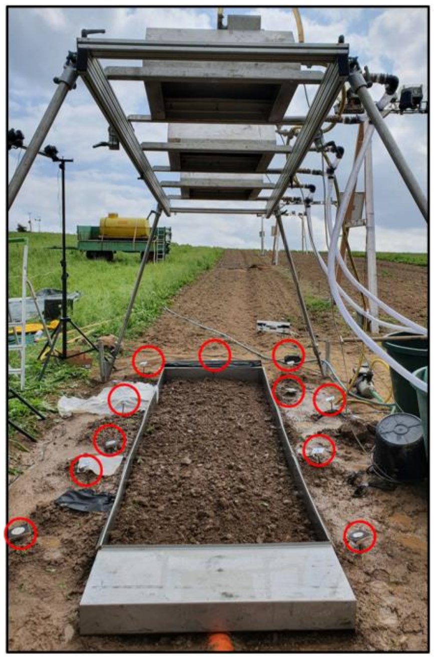
Figure 1. Rainfall simulator with ground control points (red circles) and cameras on a support structure (left side).

2.2 Camera set-up

For the point cloud reconstruction, seven to twelve cameras were installed beside the plot on a support structure, which held the cameras around 2.5 m above the ground. This ensured a near nadir perspective of the cameras, which then looked straight down onto the soil surface. All cameras were triggered externally to realize synchronous data capturing. The photos were then aligned into chunks to generate a time series of corresponding pictures. The software *Agisoft Metashape* (version 1.8.4) was used to absoultely orient the pictures and to calculate a dense point cloud for each time step.

3. METHODS

3.1 Point cloud preprocessing

Each point cloud generated from the SfM+MVS workflow was further processed in *CloudCompare* (version 2.12.4). The reconstructed experiment area was cropped from the surrounding and the bounding metal sheets, so that only the soil and the vegetation was kept. On average, the point clouds revealed an exceedingly high point density of 50-90 pts/cm², which made a spatial subsampling necessary. The point clouds were reduced to a point spacing of 2 mm.

3.2 Ground truth generation

To train and evaluate ML algorithms, the given datasets are split into training, validation and test fragments. The training set serves as data the models are trained from, whereas the validation set performs a first test of the model. In the context of validation, the models hyperparameters are tested to enable their optimization. The test set does not take part in the training routine and is used for testing the model on unseen data to provide unbiased accuracy metrics. For training and testing of the classifiers, a training and test set was prepared. The training set consisted of point clouds from five different intervals taken randomly from one experiment to ensure enough samples for training and validation. An interval comprised a time span of maximum two hours. For the test set, three point clouds from different rainfall simulation experiments with various vegetation structure were prepared and labeled (Figure 2) to test the classifiers on unseen and spatially uncorrelated data to give insights into the classifiers' ability to generalize. To check, if the training of the classifiers on samples from one time series favors the classification result on different point clouds of the same time series due to spatial correlation, a second test set is created using two point clouds from the same time series as the training dataset, just taken from different times during the experiment. Due to the positioning of the cameras only at one side of the plot, the SfM point clouds of the time series contain holes due to occlusions by higher vegetation or larger soil chunks (Figure 2, middle). The ground truth data were labeled by the same person to keep the subjective influence low. The labeling process was carried out in *CloudCompare.*

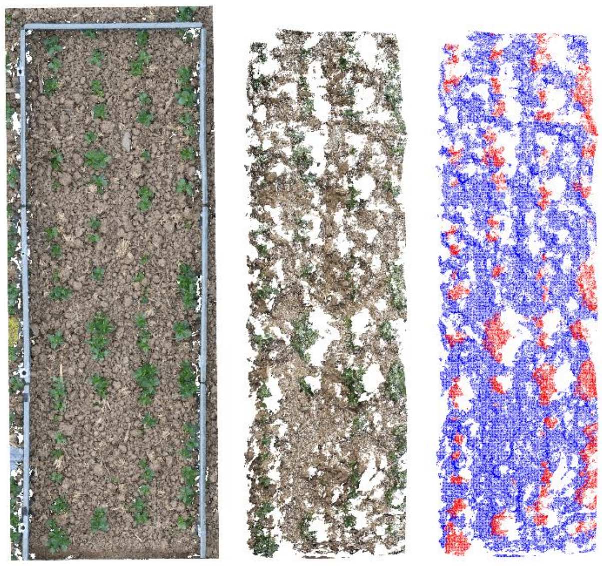
Figure 2. Ground truth: (left) SfM+MVS point cloud captured before rainfall experiment from arbitrary positions surrounding the plot; (middle) cut and reduced point cloud of the first interval of the wet run (2. part) captured from the cameras fixed at the support structure; (right) hand labelled point cloud with vegetation (red) and ground (blue) classes.

3.3 Rule-based classification

The rule-based approach is a simple workflow to classify vegetation and ground based on several different thresholds. To identify vegetation points in the SfM point cloud time series, first, a vegetation mask was generated. This mask was estimated using a point cloud of the rainfall simulation plot, which was calculated with SfM+MVS from images that were taken immediately before

the rainfall started. To ensure a high quality of this 3D model the images were captured from many perspectives by walking around the entire plot and taking the photos with only one camera, i.e., a full-format system camera. The outcome was a colorized high resolution point cloud. In that point cloud, vegetation was filtered first considering the point colors (RGB-values) because the vegetation points displayed a very distinct green color when compared to the bare soil surface. The red color scalar was divided by green (R/G) and only points below the threshold of 0.99 were kept as vegetation points. Remaining vegetation points were filtered manually. The filtered points were eventually used as the mask for vegetation points in the time-series models assuming that the time series represented a short time interval and vegetation growth could be excluded. However, vegetation movements due to water on the leaves led to a change in position, which influenced the quality of the mask filtering.
The point clouds of the time series were processed the following. First, the point colors were considered in the same way as for the high-quality 3D model, hence the threshold was considered again for the red-green color ratio. Then, the vegetation mask was applied to the point clouds to filter remaining vegetation points considering a kd-tree based nearest neighbor search with a maximum radius of 1 mm. If vegetation was within that radius, it was assigned to the vegetation class.
Remaining noisy points, which were assumed to be isolated vegetation points, were filtered based on the heights (i.e., Z value of the point coordinates) of points within a search radius of 10 mm and considering a maximum number of neighbors of 30. A threshold was defined as the sum of the averaged height values and the standard deviation of the heights, which was further multiplied by three. Points with values below that threshold were assigned to the ground class.

3.4 Random Forest classification

RF is a widely used ML algorithm that combines decision trees to perform classifications. It is an ensemble method that can handle both regression and classification tasks, and it excels at handling complex datasets with numerous features (Breiman, 2001).

Random Forest Classifier: The RF classifier constructs multiple decision trees, each trained on a randomly drawn subset of the training data and using a random selection of features. By splitting the data binary at nodes on one of the selected features with regards to the best way to separate the classes the sub-feature space is split into different regions. After the training process, the class of a sample can be predicted based on its position in the feature space. Each decision tree is aggregated into a voting forest. A sample's class is determined by collecting the prediction result of each decision tree and using the class with the most predictions, i.e., performing a majority voting. By doing so, RF avoids overfitting, reduces variance, and increases the overall stability and robustness of the model (Breiman, 2001).
The key advantages of RF are its ability to handle high-dimensional datasets and its capability to capture complex relationships between variables. Due to its logarithmic time complexity, even large datasets can be handled in a short time, which makes RF a viable classification tool for dense point clouds. The interpretability of RF is a further notable aspect. It can provide insights into feature importance, enabling the understanding of which variables contribute the most to the classification. This information can be valuable for feature selection and understanding the underlying patterns in the data.

Feature calculation: As single points of the point cloud with cartesian coordinates carry no information about its direct neighborhood and the object they belong to, different features need to be calculated before giving them to the ML algorithm. To do so, for each point in the point cloud, its neighborhood $\mathcal{N}$ is collected for a given search radius. The points' coordinates within the neighborhood are reduced to the centroid, which is the average of all point coordinates in the neighborhood. More specifically, the squared distances between each point $\mathbf{X}_i$ to the centroid $\bar{\mathbf{X}}$ is calculated, which will be used to form the 3D-covariance matrix **C** for the given neighborhood n on the query point:

$$\mathbf{C} = \frac{1}{n-1}\sum_{i\in\mathcal{N}}(\mathbf{X}_i - \bar{\mathbf{X}})(\mathbf{X}_i - \bar{\boldsymbol{X}}) \tag{1}$$

C is a 3 x 3-dimensional matrix. The matrix' main diagonal contains the variance of each coordinate dimension. Besides the main diagonal, the covariances are held. To gain a better understanding of the variance of the points in the neighborhood, i.e., the spatial extent and orientation, Eigenvectors and Eigenvalues are extracted. To analyze the neighborhood with regards to its largest variance directions, Eigenvalue decomposition is used.

$$\mathbf{A}\cdot x = \lambda\cdot x \tag{2}$$
$$\mathbf{A}\cdot x = \lambda\mathbf{I}\cdot x \tag{3}$$
$$(\mathbf{A} - \lambda\mathbf{I})\cdot x = 0 \tag{4}$$

For **C**, the three Eigenvectors e_1, e_2, e_3 and corresponding Eigenvalues $\lambda_1, \lambda_2, \lambda_3$ (where $\lambda_1 > \lambda_2 > \lambda_3$), are extracted using the identity matrix **I**. Geometrically interpreted, the first Eigenvalue points in the direction with the largest variance in the neighborhood with the corresponding Eigenvalue as a measure of the magnitude of the variance. The second Eigenvector points, perpendicular to the first Eigenvector, in the direction of the second largest variance in that neighborhood. The third Eigenvector completes the orthogonal Eigenvector frame. These Eigenvectors form the basis of the feature computation to analyze and encode the underlying surface structure.
A set of nine unique features (Table 1) were extracted for each point from the point cloud for a specific neighborhood. For this purpose, the program *CloudCompare* was used. For each set of nine features ten different radii between 1 and 10 cm were considered for the computation. The selection of features was based on studies conducted by (Weinmann et. al., 2017). The main components for the feature calculation were the Eigenvalues λ_i:

Feature	Sign	Formula
Eigenentropy	E_λ	$\sum_{i=1}^{3}\lambda_i\cdot\ln\lambda_i$
Omnivariance	O_λ	$\sqrt[3]{\lambda_1\cdot\lambda_2\cdot\lambda_3}$
Anisotropy	A_λ	$(\lambda_1 - \lambda_3)/\lambda_1$
Planarity	P_λ	$(\lambda_2 - \lambda_3)/\lambda_1$
Linearity	L_λ	$(\lambda_1 - \lambda_2)/\lambda_1$
Surface variation	C_λ	$\lambda_3/(\lambda_1 + \lambda_2 + \lambda_3)$
Sphericity	S_λ	λ_3/λ_1
Verticality	V	$1 - \lvert\langle[0\ 0\ 1], e_3\rangle\rvert$
Sum of Eigenvalues	Sum_λ	$\lambda_1 + \lambda_2 + \lambda_3$

Table 1. Extracted features based on Weinmann et al, 2015.

Training of RF classifiers: The training was performed using the programming language python and the freely available ML library scikit-learn (Pedregosa et al., 2011). For the calculation of the RF, following hyperparameters were set:

- number of decision trees to build.
- maximal depth each decision tree is allowed to be split into,
- number of features, which are randomly drawn as a subset from the whole feature set.

To find the optimal hyperparameter set, a grid search cross validation was performed. On a given set for above mentioned hyperparameters, each permutation was selected and used for the training of a model. Each training was performed with k-fold cross validation. The training set was split into k sets, where k-1 sets were used as training samples. The model was then validated with the left-over set. The role of the validation and training set was switched, so that each set once had been a validation set. In the end, for one permutation of hyperparameters k models were trained. On the one hand, this process of cross validation and grid search returned an optimal choice of hyperparameters for the classification problem. On the other hand, the different models for each hyperparameter permutation highlighted imbalances in the training data, which occurred during the validation.

3.5 PointNet++ Classification

PointNet++ is an extension of the PointNet deep learning architecture that is designed for point cloud processing tasks for 3D object classification and segmentation (Qi et al., 2017). The main idea behind PointNet++ is the improvement of the hierarchical feature learning process in PointNet (Qi et al., 2016) by capturing local and global context. While PointNet processes each point independently, PointNet++ introduces a neural network architecture that hierarchically aggregates local features and thereby gradually expands the receptive field. PointNet++ operates in a multi-scale grouping and sampling manner. It first performs farthest point sampling to select a subset of representative points. Then, it forms local regions by grouping neighboring points together. Within each local region, it uses shared MLPs to learn local features. These local features are then aggregated at different scales using a hierarchical set of grouping and sampling operations. By progressively increasing the receptive field and incorporating multi-scale features, it can better understand the spatial relationships and interactions between points in a point cloud.

Training of PointNet++ models: For the training of PointNet++ models the code from Yan (2019) was used. The repository contains code for training the models and importing datasets from public available datasets like ScanNet (Dai et. al., 2017) and ShapeNet (Chang et. al., 2015). To import the point clouds presented in this study, the code had to be adapted. The changes were made to the import function for the ScanNet semantic segmentation dataset, which is given in the original implementation. In the adapted version, our labelled datasets were used for training and validation and the number of points K was set. K is an upper threshold for the number of points a neighborhood can have. With respect to K, the input cloud was subsampled with the farthest point algorithm in such a way, that all core points and its neighborhood covered the whole point cloud.

3.6 Metrics for performance assessment

To evaluate and compare the trained models and implemented algorithms for the classification tasks, sampled ground truth and predicted classification labels need to be compared. Kohavi & Provost (1998) provide a brief description of terms and evaluation techniques for ML algorithms.

Confusion Matrix: To calculate the quality of a classifier, a confusion matrix was used (Figure 3). It has a quadratic shape of L x L, where L is the count of classes. For each predicted sample with (ground-truth-label, predicted-label) as label pair, a field in this matrix counts each occurrence of such pair. Each entry on the main diagonal corresponds to a correct classified sample whereas each entry beside those fields is a misclassification.

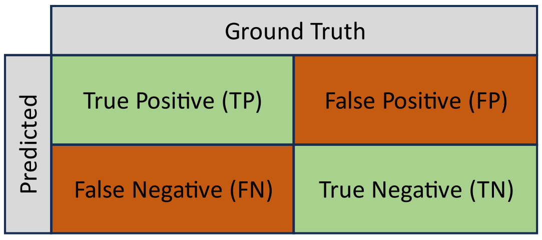

Figure 3. Schematic representation of the confusion matrix with correctly classified (green cells) and misclassified (red cells) samples.

Important metrics: To quantify the classification results, several metrics, that can be derived from the entries of the confusion matrix, were used. The most important are the true positive rates (TPR) also known as recall, the false positive rates (FPR), the precision and the F1 score:

$$\mathrm{TPR} = \mathrm{TP}/(\mathrm{TP} + \mathrm{FN}) \tag{5}$$

$$\mathrm{FPR} = \mathrm{FP}/(\mathrm{FP} + \mathrm{FN}) \tag{6}$$

$$\mathrm{precision} = \mathrm{TP}/(\mathrm{TP} + \mathrm{FP}) \tag{7}$$

$$\mathrm{F1\ score} = 2\mathrm{TP}/(2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}) \tag{8}$$

The TPR represents the proportion of positive instances that are correctly classified as positive, indicating the model's ability to correctly identify true positives. On the other hand, the FPR measures the proportion of negative instances that are incorrectly classified as positive, highlighting the instances that were falsely identified. Precision measures the accuracy of positive predictions. Finally, the F1 score is a harmonic mean of precision and recall (=TPR), providing a balanced measure of a classifier's performance.

4. RESULTS

4.1 Random Forest Training

The results for the grid search training of the RF are presented in Table 2. For the number of trees, 100 and 200 trees were trained as choices to check if the number of decision trees does influence the test score values. The maximum number of features was defined by the square root (sqrt) and logarithm function (with base of two – log2) to reduce the randomly selected feature count to decorrelate the built trees. In our case, the sqrt function created ten features and the log2 function created seven features. Both functions were used to also detect differences in the testing score. Furthermore, and most importantly, the maximum depth parameter varied to assess the influence of the complexity of the built decision trees. Beginning with lower values of two and five, the values were then increased to observe the models testing score and to build more complex models.

The number of trees and the maximum number of drawn features did not influence the performance of the model much. However, changing the maximum depth of the RF did influence the outcome most. The best test scores were achieved when building trees with a maximum depth of 25. The time used to train the models was shorter if less trees were trained, using the log2 function and growing shallow decision trees.

Mean fitting time [min]	Max depth	Max features	Number of trees	Mean test score (F1-score)
3.9	2	sqrt	100	0.688
7.9	2	sqrt	200	0.686
2.8	2	log2	100	0.678
5.2	2	log2	200	0.676
9.2	5	sqrt	100	0.736
18.9	5	sqrt	200	0.736
6.7	5	log2	100	0.732
12.5	5	log2	200	0.732
17.9	10	sqrt	100	0.816
35.6	10	sqrt	200	0.817
12.5	10	log2	100	0.805
23.1	10	log2	200	0.805
30.1	20	sqrt	100	0.953
60.1	20	sqrt	200	0.954
20.7	20	log2	100	0.951
38.3	20	log2	200	0.952
31.5	25	sqrt	100	0.966
63.1	25	sqrt	200	0.967
21.4	25	log2	100	0.966
39.9	25	log2	200	0.966

Table 2. Results of the grid search hyperparameter tuning with sqrt = 10 features and log2 = 7 features.

After the training, the feature importance was retrieved. To distinguish between valuable and negligible values, the values were normalized by the maximum of all values. Afterwards, values greater than 0.2 were chosen as valuable. The kept features are displayed in Figure 4.

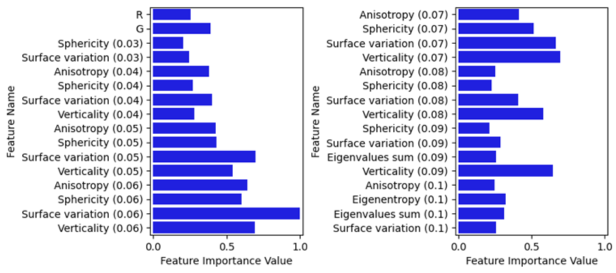

Figure 4. Feature importance with neighborhood radius in meter in parenthesis. Only features with values greater than 0.2 are shown.

Anisotropy, sphericity, surface variation and verticality were the most important features across all scales. The surface variation had the greatest influence up to a search radius of 6 cm and the verticality had the greatest influence for a radius of 7 cm. The red (R) and green (G) colors also played a minor role in the classification process. Features for scales (i.e., search radii) of one and two centimeters were neglectable.

4.2 PointNet++ Training

To use the best model for the classification, different grouping strategies and point numbers were used. The above-mentioned implementation provided single scale grouping (SSG) and multi scale grouping (MSG). To check the influence of K on the classification result, three different values (256, 4096, 8192) were tested.

Grouping	Batch size	Epochs	K	Training time [min]
SSG	16	32	8192	23.5
MSG	16	32	8192	24.2
SSG	16	32	4096	27.1
MSG	16	32	4096	27.5
SSG	16	32	256	128.2
MSG	16	32	256	139.2

Table 3. Parameter setup for PointNet++ model training with single- and multi-scale grouping (SSG/MSG).

During the training, training loss values and evaluation loss values were exported and plotted against the epoch number (Figure 5).

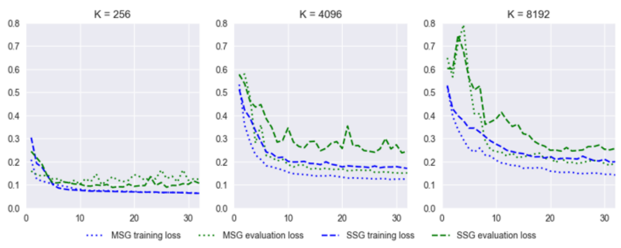

Figure 5. Training and evaluation loss values for SSG and MSG models for different neighbor points K.

An overall decreasing loss value with decreasing K became obvious. There was a difference in using SSG and MSG for the 4096 and 8192 points variants of the model. Thereby, MSG performed better. For the 256 variant, both grouping methods achieved similar loss values.

4.3 Performance on test sets

The assessment of the best RF and PointNet++ classifiers was performed on both test datasets, i.e., point clouds from the same experiment but at different times and point clouds from different experiments. On the test set from the same experiment, spatio-temporal correlations and overfitting must be assumed. When the classifier was assessed on the dataset, which was from a different rainfall simulation (containing three experiments with one time slice each), this is not the case. The rule-based algorithm was also evaluated on two time slices, which were part of the time series the algorithm was trained on, thus again potentially leading to too optimistic performance metrics due to correlations.

Quality metrics: Quality metrics were calculated with equations 5, 7 and 8. The RF and PointNet++ model achieved higher values of the accuracy metrics (Precision 0.92 and Recall 0.93 and Precision 0.97 and Recall 0.95, respectively) when applied to the same experiment compared to different rainfall simulations. Table 4 and Table 5 show the calculated metrices for each classifier and class.

Model	Class	Precision	Recall	F1-Score
Rule-based	Ground	-	-	-
	Vegetation	-	-	-
RF	Ground	0.91	0.91	0.91
	Vegetation	0.67	0.67	0.67
Point-Net++	Ground	0.96	0.91	0.93
	Vegetation	0.72	0.84	0.78

Table 4. Accuracy metrics on the validation set from different time series. The best metrics are marked grey.

Model	Class	Precision	Recall	F1-Score
Rule-based	Ground	0.97	0.96	0.96
	Vegetation	0.87	0.90	0.88
RF	Ground	0.92	0.93	0.92
	Vegetation	0.78	0.78	0.78
Point-Net++	Ground	0.97	0.95	0.96
	Vegetation	0.85	0.91	0.88

Table 5. Accuracy metrics on the validation set from the same time series. The best metrics are marked grey.

Confusion matrix: The ground truth and predicted labels are compared and summarized in confusion matrices (Figure 6).

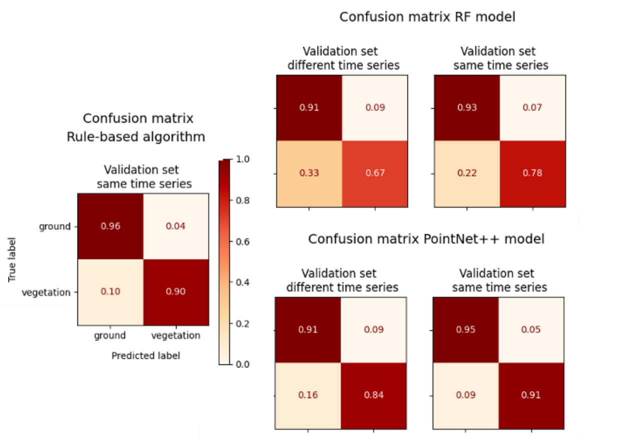

Figure 6. Confusion matrices of best classifier of the rule-based algorithm and the RF and PointNet++ models, calculated for the test set from the same experiment as the training set and for the other rainfall simulations.

It is noticeable that the values of the false negatives were always greater than the false positives; for values of the rule-based and PointNet++ model two times larger and for the RF model even three times larger. This means that there are relatively more ground points that are predicted as vegetation than there are vegetation points that are predicted as ground.

Misclassifications: To evaluate the spatial distribution of miss classification, plotting the classification results as well as its errors onto the point cloud is a helpful visualization (Figure *7*gure 7, 8 and 9). The Figures present the test dataset from different experiments (three point clouds at the left) and from the same rainfall simulation (two point clouds at the right). The predicted false ground points are shown in purple and the false vegetation points in blue.

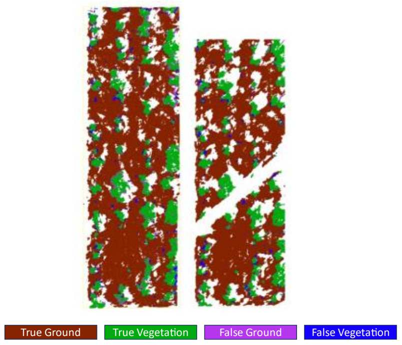

Figure 7. Rule-based classification results mapped on input point clouds of the test dataset from the same experiment as the training set.

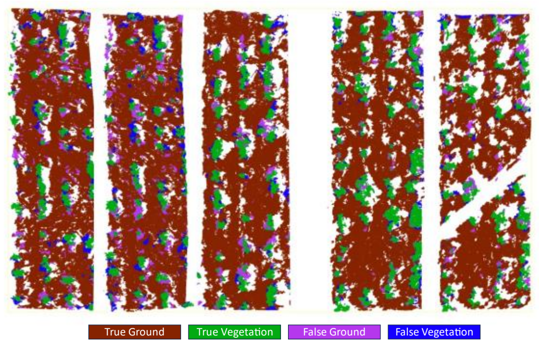

Figure 8. Classification results for the RF classifier mapped on input point clouds of the test dataset from different experiments (three point clouds on the left) and from the same experiment (two point clouds on the right).

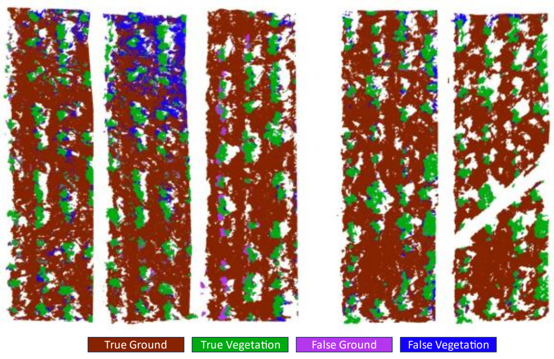

Figure 9. Classification results for the PointNet++ classifier mapped on input point clouds of the test dataset from different experiments (three point clouds on the left) and from the same experiment (two clouds on the right).

The point clouds of the two right-hand time slices in 7 till 9 originate from the same rainfall simulation experiment. The line of missing points in the rightmost time slice was due to different light conditions during the rainfall experiment. The shadows casted by the sprinkler's legs resulted in very dark areas in the images and thus the point matching failed. The misclassifications in the Figures indicate that the false ground and false vegetation points are clustered around the true vegetation points. Furthermore, some plots (Figure ***9***igure 9, second point cloud from left) are largely misclassified as vegetation in some regions, which weakens the information content in those sections.

5. DISCUSSION

The aim of this study was to minimise the false negative values (FN or False Ground) of vegetation points (i.e., vegetation classified as ground) because this observation can negatively influence the soil erosion process assessment during the later processing of the point clouds. For example, a DEM might be derived from the soil points and some remaining vegetation points could then be considered as eroded soil during the differencing of DEMs.

Overall, the detection of ground points works well because in that case every metric was above 90%. Differences between the classifier's quality become obvious for the vegetation classification. For point clouds from different time series (different rainfall experiments with different vegetation structure), the metrics for the quality of the vegetation classification were lower than for point clouds taken from the same time series, highlighting the influence of spatio-temporal correlations and overfitting to the plot from which the training data was provided. The RF classifier performed worst on both datasets. The PointNet++ results were close to the results achieved with the rule-based classification.

Looking onto the spatial distribution of the misclassification, a differentiation between both test datasets must be made to point out different behaviours of the ML classifiers on seen and unseen data. The performance of the rule-based classifier was only assessed on one dataset. It shows a very good classification of ground and vegetation on both point clouds. False ground points were mostly detected at the fringes of larger vegetation clusters, which were left over after the filter process. False vegetation was also present at those fringes, but also appeared as little clusters on bare soil, which cut holes in the point cloud after the filtering. The PointNet++ models showed on the same test set similar patterns of misclassified fringes on the vegetation clusters, which was similar to the performance of the rule-based approach. The deep learning-based approach was a very reliable classifier in detecting vegetation with a very low amount of false ground classification and a similar amount of false vegetation compared to the rule-based classifier. Using the classifier on unseen and spatially uncorrelated data like the first test set, the rate of misclassification increased for both, false ground and false vegetation. Still a considerable amount of vegetation clusters was found, but the amount of false ground classification rose on many of those clusters. To bypass heavily misclassified regions (Figure 9), gathering more training data in those time series and training a model with an extended training set might create a more generalized model. Furthermore, other deep-learning based architectures might be considered; such as the graph-based model DGCNN (Wang, 2018), which is another approach for encoding features and their neighborhood considering the edges between points within a neighborhood.

The RF classifier produced far more misclassifications. It performed relatively decent on the spatially correlated dataset from the same time series, finding many vegetation clusters. But it also classified many tiny clusters as ground, alluding to unfiltered vegetation in the process. The amount of false ground in comparison to false vegetation was considerably higher relative to the other classifiers.

Applying the RF model onto the point clouds from the other experiments (Figure 8, left) revealed that many vegetation clusters were not found and therefore classified as ground; the false ground values overtook the false vegetation values, which was misaligned to the actual aim of the classifier. To overcome this limitation, similar to the PointNet++ classifier, more training data in those timeseries must be included to generate a more generalizing model. Furthermore, more handcrafted features, which are even more expressive – e.g., relative height or color ratios, might further improve the classifier for vegetation detection. Also, improved models like Probabilistic RF (not yet investigated for point cloud classification; Reis et al., 2018) to model uncertainties of features and labels could be investigated. Further improvements of the RF could be the consideration of correlations between trees, only selecting the trees with the lowest correlation to form a forest (as proposed in Xue et. al., 2020).

6. CONCLUSION

For this study, three algorithms were trained and tested on different point clouds from different rainfall simulations to filter vegetation and ground in DEM time series calculated with SfM+MVS. The rule-based classifier was the most precise for all candidates. However, the downside of this approach is the time-consuming crafting of filter masks, which are only usable for that single experiment. PointNet++ is a promising alternative to the rule-based method, as it performed similar (partially even better) and it might be transferable to other rainfall experiments if trained with more examples. Another advantage is the sole input of point clouds, which avoids the need to design and calculate expressive point features. RF is a widely used classification algorithm, but it failed in this specific application and was outmatched by the other classifiers. Future work should focus on improving and implementing other deep learning-based algorithms. Furthermore, the amount of training data should be investigated to find the sweet spot between ground truth generation and classification quality.

ACKNOWLEDGEMENTS

We thank Lea Epple and our colleagues from the TU Bergakademie Freiberg for the support during the rainfall experiments. The work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation – 405774238).

REFERENCES

Blanch, X., Eltner, A., Guinau, M., Abellan, A., 2021: Multi-Epoch and Multi-Imagery (MEMI) photogrammetric workflow for enhanced change detection using time-lapse cameras. *Remote Sensing*, 13(8). https://doi.org/10.3390/rs13081460.

Breiman, L., 2001: Random Forests. *Machine Learning,* 45(1), 5-32. https://doi.org/10.1023/A:1010933404324.

Candido, B., Quinton, J., James, M., Silva, M., de Carvalho, T., de Lima, W., Beniaich, A., Eltner, A. 2020: High-resolution monitoring of diffuse (sheet or interrill) erosion using structure-from-motion. *Geoderma,* 375, 114477. https://doi.org/10.1016/j.geoderma.2020.114477.

Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H.,Xiao, J., Yi, L., Yu, F., 2015: ShapeNet: An Information-Rich 3D Model Repository. *Graphics*. https://doi.org/10.48550/arXiv.1512.03012.

Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T., Nießner, M., 2017: ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. *Computer Vision and Pattern*. https://doi.org/10.48550/arXiv.1702.04405.

Eltner, A., Kaiser, A., Abellan, A., Schindewolf, M., 2017: Time lapse structure-from-motion photogrammetry for continuous geomorphic monitoring. *Earth Surface Processes and Landforms,* 42(14), 2240-2253. https://doi.org/10.1002/esp.4178.

Eltner, A., Schneider, D., Maas, H.-G., 2016: Integrated Processing of High Resolution Topographic Data for Soil Erosion Assessment Considering Data Acquisition Schemes and Surface Properties. *International Archives of the Photogrammtry, Remote Sensing and Spatial Information Sciences,* XLI-B5, 813-819. https://doi.org/10.5194/isprs-archives-XLI-B5-813-2016.

Epple, L., Kaiser, A., Schindewolf, M., Bienert, A., Lenz, J., Eltner, A. 2022: A Review on the Possibilities and Challenges of Today’s Soil and Soil Surface Assessment Techniques in the Context of Process-Based Soil Erosion Models. *Remote Sensing*, 14(10), 2468. https://doi.org/10.3390/rs14102468.

Erickson, B. J., Kitamura, F., 2021: Magician’s Corner: 9. Performance Metrics for Machine. Radiology: Artificial Intelligence, 3. https://doi.org/10.1148/ryai.2021200126.

Hänsel, P., Schindewolf, M., Eltner, A., Kaiser, A., Schmidt, J., 2016: Feasibility of High-Resolution Soil Erosion Measurements by Means of Rainfall Simulations and SfM Photogrammetry. *Hydrology,* 3 (38). https://doi.org/10.3390/hydrology3040038

Kaiser, A., Erhard, A., Eltner, A. 2018: Addressing uncertainties in interpreting soil surface changes by multi-temporal high resolution topography data across scales. *Land Degradation & Development*, 9(8), 2264-2277. https://doi.org/10.1002/ldr.2967.

Kohavi, R., Provost, F., 1998: Glossary of terms. Special issue of applications of machine learning and the knowledge discovery. *Machine Learning*, 30, 271-274.

Kromer, R., Walton, G., Gray, B., Lato, M., Group, R., 2019: Development and optimization of an automated fixed-location time lapse photogrammetric rock slope monitoring system. *Remote Sensing*, 11(16). https://doi.org/10.3390/rs11161890.

Malinowski, R., Heckrath, G., Rybicki, M., Eltner, A. 2023: Mapping rill soil erosion in agricultural fields with UAV-borne remote sensing data. *Earth Surface Processes and Landforms*, 48(3), 596-612. https://dx.doi.org/10.1002/esp.5505.

Martins, J., Marcato Junior, J., Paetzig, M., Sant’Ana, D., Pistori, H., Liesenberg, V., Eltner, A. 2023: Identifying plant species in kettle holes using UAV images and deep learning techniques. *Remote Sensing in Ecology and Conservation*, 9(1), 1-16. https://doi.org/10.1002/rse2.291.

Onnen, N., Eltner, A., Heckrath, G., Van Oost, K. 2020: Monitoring soil surface roughness under growing winter wheat with low altitude UAV sensing. *Earth Surface Processes and Landform*s, 45(14), 3747-3759. https://doi.org/10.1002/esp.4998.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011: Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825-2830.

Qi, C., Su, H., Mo, K., Guibas, L., 2016: PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *Computer Vision and Pattern Recognition*. https://doi.org/10.48550/arXiv.1612.00593.

Qi, C., Yi, L., Su, H., Guibas, L., 2017: PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *Computer Vision and Pattern Recognition*. https://doi.org/10.48550/arXiv.1706.02413.

Reis, I., Baron, D., Shahaf, S., 2018: Probabilistic Random Forest: A machine learning algorithm for noisy datasets. *The Astronomical Journal*, 157(1). https://doi.org/10.3847/1538-3881/aaf101.

Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J. M., 2018: Dynamic Graph CNN for Learning on Point Clouds. *Computer Vision and Pattern Recognition*. https://doi.org/10.48550/arXiv.1801.07829

Weinmann, M., Jutzi, B., Hinz, S., Mallet, C., 2015: Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. *ISPRS Journal of Photogrammetry and Remote Sensing,* 105, 286-304. https://doi.org/10.1016/j.isprsjprs.2015.01.016.

Weinmann, Ma., Jutzi, B., Mallet, C., Weinmann, Mi., 2017: Geometric features and their relevance for 3D point cloud classification. *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.,* IV(1W1), 157-164. https://doi.org/10.5194/isprs-annals-IV-1-W1-157-2017.

Xue, D., Cheng, Y., Shi, X., Fei, Y., Wen, P., 2020: An Improved Random Forest Model Applied to Point Cloud Classification. *IOP Conf. Ser.: Mater. Sci. Eng.,* 768(7). https://doi.org/10.1088/1757-899X/768/7/072037.

Yan, X., 2019: Pointnet/Pointnet++ Pytorch. https://github.com/yanx27/Pointnet_Pointnet2_pytorch