THE MDSR VEGETATION FILTER

Jakub Kučera*,[1]

*jakub.kucera.1@fsv.cvut.cz
[1]Department of Special Geodesy, Faculty of Civil Engineering, Czech Technical University in Prague, Thákurova 7, 166 29 Prague, Czech Republic

Abstract
This paper deals with the software implementation of a vegetation filter using the MDSR algorithm and its subsequent testing on a variety of pre-selected point clouds.

The results of each filtering algorithm were compared visually in sections with the results of other conventionally used filtering algorithms. Comparisons were also made by calculating the root mean square deviations of the results of the conventional algorithms from the TIN surface generated from the MDSR filter results.

Keywords
Point cloud, ground filtering, classification

1 INTRODUCTION

Nowadays, the use of laser scanners and photogrammetry as reliable sources of spatial data [1] is becoming more and more widespread in the field of surveying. However, even these modern technologies have their specific limitations and problems, such as noise in the data and unwanted objects that need to be removed [2]. Manual classification of point clouds can be very laborious and tedious, so vegetation filters are often used in practice and can greatly facilitate this task. In this paper, the software implementation of the MDSR vegetation filter and its testing on selected point clouds will be described, in order to contribute to a more efficient processing of spatial data acquired by laser scanners and photogrammetry.

Professor Ing. Martin Štroner, Ph.D. and his colleagues from the Faculty of Civil Engineering, CTU in Prague, presented a new filter, namely the Multidirectional Shift Rasterization filter (MDSR) [3], which uses an innovative approach to filtering vegetation in point clouds. However, this method did not have a sufficiently reliable software implementation that would allow for an efficient analysis of the algorithm's results. For this reason, this thesis was developed to give the MDSR a solid software foundation.

The algorithm has been implemented in the C++ programming language and, in addition to its original features, it introduces new features, such as the ability to define multiple runs of the algorithm in succession or to store the number of times each point is labelled by the algorithm as ground.

Subsequently, the algorithm was tested on four clouds with different levels of terrain and vegetation complexity. These clouds included very basic terrain, very dense vegetation, a steep slope and an area with rugged terrain and a railway bridge. The results of the MDSR algorithm were then compared to the results of three other conventionally used vegetation filters, specifically the Cloth Simulation Filter (CSF) [4], the Simple Morphological Filter (SMRF) [5] and the Adaptive Triangulated Irregular Network filter (ATIN) [6], using cloud sections for visual assessment and the root mean square deviations of the cloud points from the TIN surface created from the MDSR result points.

The content of this article was originally created as part of the author’s master’s thesis named “Software Implementation of the MDSR Vegetation Filter and Its Testing on Selected Point Clouds” [7].

DOI 10.13164/juniorstav.2024.24017

2 METHODOLOGY

Multidirectional Shift Rasterization

The basic assumption of the presented algorithm for ground filtering is that the lowest point in a given point cloud area is very likely to be located on the terrain (more generally, on the surface of interest). Therefore, the algorithm divides the cloud by a horizontally oriented grid into cells of a predefined size and searches for the lowest points in them.

In order for a cell to always contain a point located on the ground, the cell size needs to be set larger than the largest object located in the cloud (and this includes places in the cloud where data is not located, such as under very dense vegetation). In practice, this requires a cell size of up to several meters. Thus, as the cell size increases, the chance that the selected lowest point will actually be on the ground increases. On the other hand, as the cell size increases, the total number of selected points characterizing the terrain decreases. Due to this, other characteristic points (local maxima such as ridges and edges) would not be recorded either.

Thus, this paper introduces two ways to densify the resulting cloud in the computation while simultaneously detecting the remaining points that are important in terms of terrain characterization:

- The first way to increase the point cloud density and at the same time to better delineate the terrain is to shift the horizontally oriented grid, which defines the position of each cell, in both the X and Y axes in very small steps compared to the chosen cell size (hence Shift Rasterization).
- The second way is to rotate the originally horizontally oriented grid around all three coordinate axes. This changes the view (hence Multidirectional) of the cloud and thus the relative position of the points in the cloud when searching for the lowest points.

The combination of these two fundamentally simple methods, rotating the view (and therefore the grid which is perpendicular to the view) and then moving the grid in small increments, is the main principle to preserve all the characteristic points in the cloud. This principle is explained in more detail in Fig. 1.

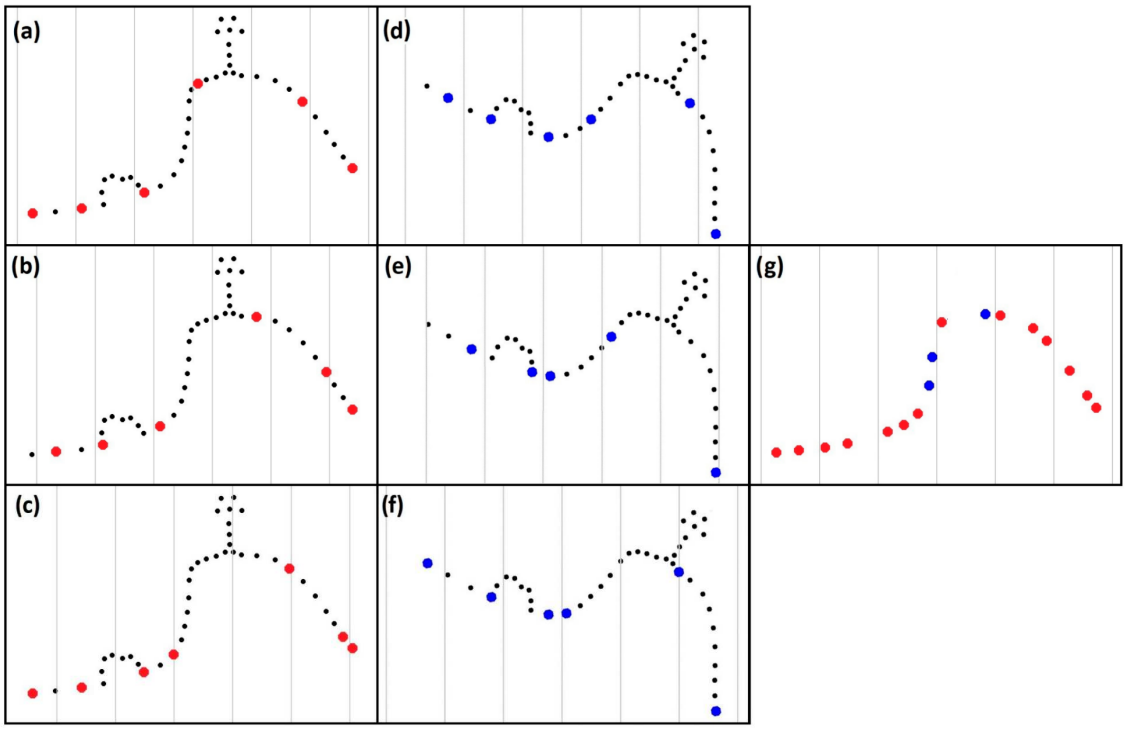

Fig. 1 Basis of functioning of the MDSR: (a, b, c) three algorithm steps without view rotation; (d, e, f) three algorithm steps with view rotation; (g) all selected points [3].

DOI 10.13164/juniorstav.2024.24017

The same principle of gradual selection of ground points using 3D visualization is shown in Fig. 2.

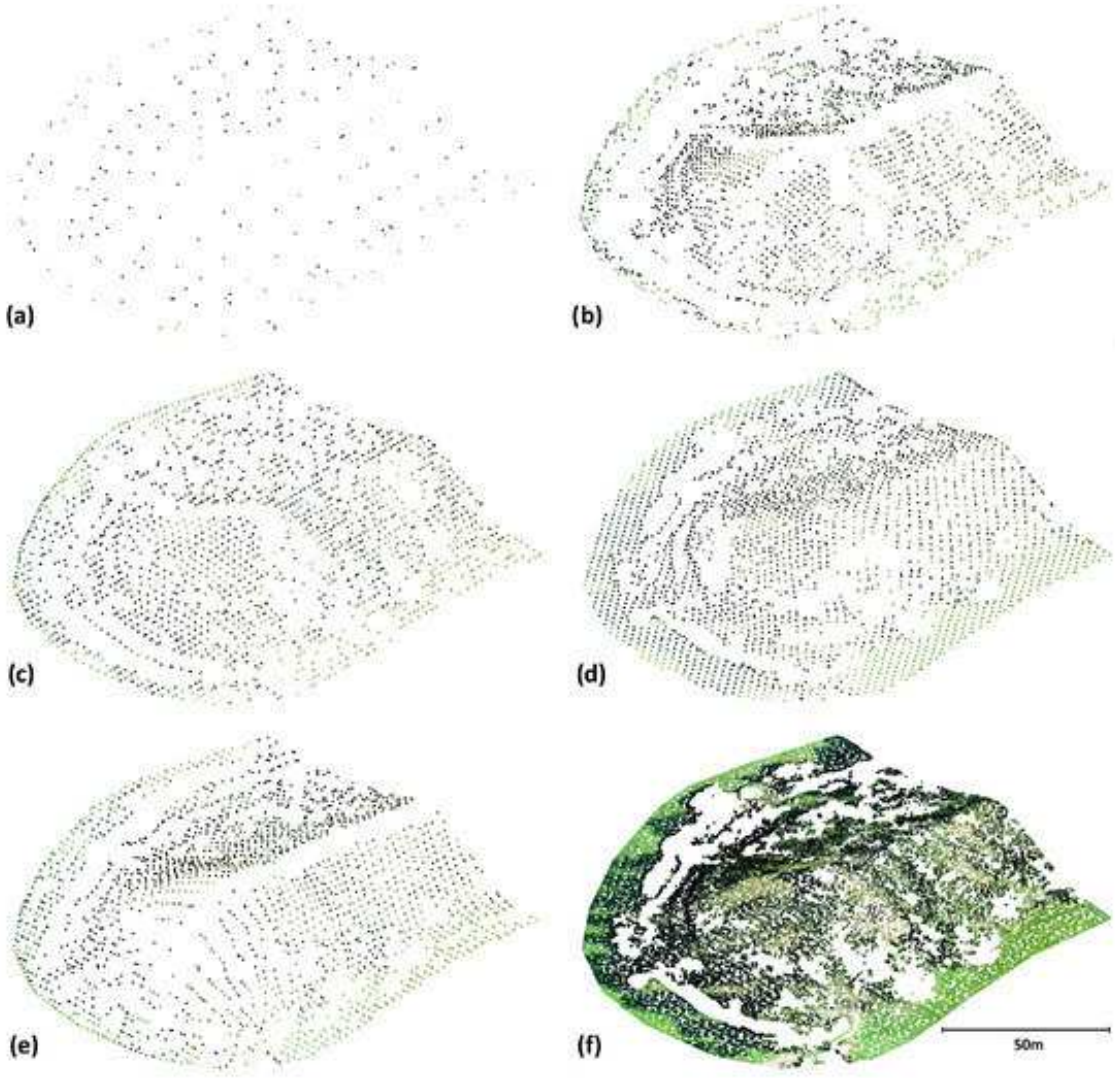

Fig. 2 Gradual selection of the ground points using MDSR: (a) without grid shift and rotation; (b) 5× grid shift, no rotation; (c, d, e) 5× grid shift, one view rotation; (f) point cloud combining all previous results [3].

Parameters

The algorithm works with five parameters in the calculation which must be defined by the user before the calculation. These criteria are relatively easy to understand. The specific values of these parameters must be set individually for each cloud. There is no one-size-fits-all setting that can filter all clouds with the same efficiency.

The first parameter, as mentioned earlier, is the cell size in the grid. This needs to be set large enough so that no unwanted relics of objects above the terrain (building walls, treetops) remain in the resulting cloud.

Another parameter is the number of shifts of the grid. This indirectly indicates the size of the step over which the grid is shifted. The shift always occurs in two directions perpendicular to each other, which are also perpendicular to the current view of the cloud. The greater the number of shifts we set, the greater the density of the resulting cloud.

The other three parameters are very similar. They are the angles of rotation about each axis. Any number of rotation angles can be defined for each axis. This number depends mainly on the ruggedness and complexity of the terrain in the processed point cloud. Specifically, rotation angles around the X-axis (alpha angles), around the Y-axis (beta angles) and around the Z-axis (gamma angles) need to be set.

DOI 10.13164/juniorstav.2024.24017

Properties of the filter

Primary testing, described in the original article by professor Štroner [3], confirmed the high efficiency of the MDSR filter for data with very dense vegetation compared to other conventional filters. In addition, the filter was able to cope with clouds with a very complex terrain.

Another advantage of the filter lies in the fact that the filter does not perform any approximation or estimation of the terrain course but works directly with the measured points.

Due to the way the filter works, the resulting cloud is usually more diluted than the input cloud. This fact can also be considered an advantage rather than a disadvantage for dense clouds.

The computational complexity, and hence the time required to run the algorithm, increases roughly quadratically with the detail the user desires. In fact, for greater resulting detail (resulting resolution), the number of grid shifts needs to be increased. The computational complexity is also dependent on the size of a single cell, and therefore on the total grid size and the number of rotation angles defined.

A disadvantage of the MDSR is the residual points at the edges of the resulting cloud which cannot be detected by the filter. However, this is not a major complication. A simple solution is to crop the cloud at the edges.

Software implementation

The main goal of this work was to create a software program that utilizes the Multidirectional Shift Rasterization algorithm (MDSR) to filter vegetation in point clouds. The program is designed to read point cloud data in the ASCII format, perform cloud filtering and save the resulting cloud in the same input format.

Efficiency and speed were the major considerations during the development process, leading to the selection of the C++ programming language which allows for low-level memory management and compilation. Parallelization of the computation flow was also implemented to further optimize speed. The OpenMP [8] programming interface was chosen for parallelization, as it enables easy implementation of shared memory multiprocessing. The source code was developed using Visual Studio Code [9], an open-source source code editor that offers features such as code suggestions and error detection.

In addition to the features described in previous studies, the implemented program includes the function of counting the number of times each point is detected as terrain. This information is stored separately for each point in a new column in the resulting text file. Utilizing a color scale, this value, ranging from 1 to the maximum number of iterations, provides an additional insight into the progress and operation of the MDSR filter.

Another added feature of the program is the ability to perform multiple runs of the filter consecutively. The program reads the input point cloud, applies the filtering, saves the resulting cloud and then reloads and performs additional filtering with new parameters. This cycle can be repeated indefinitely, allowing for an unlimited number of runs and optimizing the program's efficiency.

The implementation utilizes associative containers, specifically the set and map containers, to store information about selected terrain points. The set container stores unique values, representing the point numbers without repetition. The map container, on the other hand, stores a data value and a key for each point, enabling the storage of the number of times the algorithm selected as terrain. The use of unordered versions of these containers enhances the program's speed.

Test point clouds

The MDSR algorithm was tested on four different clouds, each containing varying degrees of complex terrain.

- The first cloud had flat terrain with low vegetation and overhead high voltage lines. It had 1,408,072 points over an area of approximately 20,000 m^2.
- The second cloud had a road surrounded by a forest with relatively steep slopes. It was the largest of the clouds, containing 10,016,451 points over an area of almost 25,000 m^2.
- The third cloud had a vertical cliff covered with dense vegetation, structures and nets. It covered approximately 1,240 m^2 and had 2,785,912 points.
- The last cloud covered an area of approximately 1,207 m^2 and contained a bridge structure with low and high vegetation. It had 830,836 points.

The data for these clouds were acquired using various instruments, such as the Leica Pegasus laser scanner, the Trimble X7 ground scanner, the Leica P40 ground-based laser scanner, and the DJI Phantom 4 RTK UAV. [3]

The selected clouds colored by relative height can be seen in Fig. 3.

DOI 10.13164/juniorstav.2024.24017

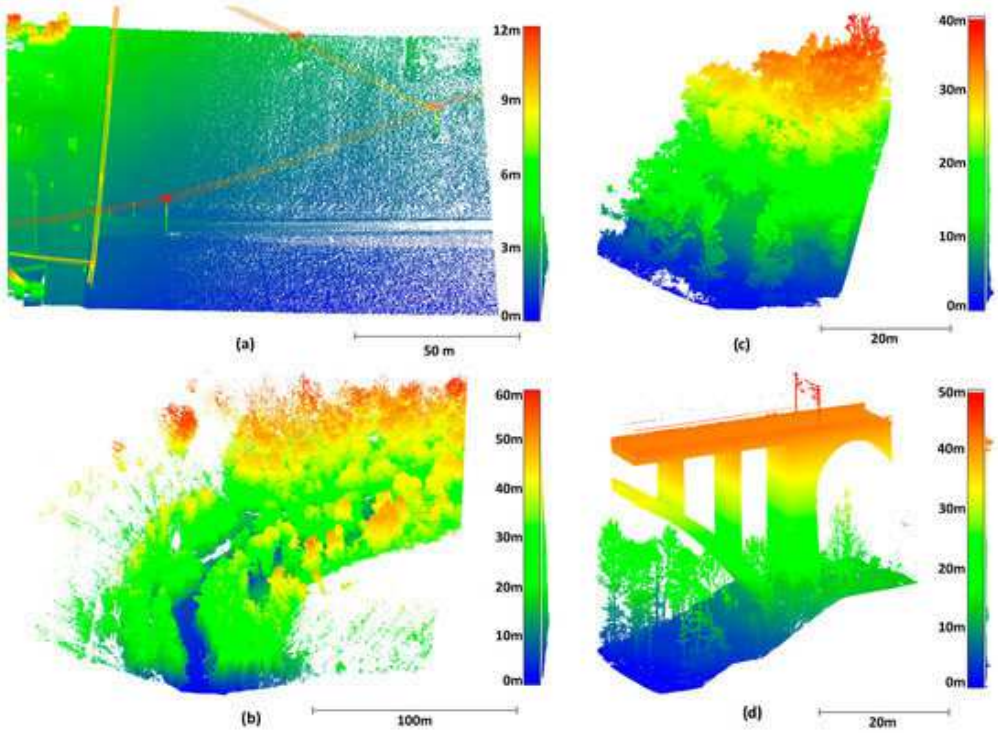

Fig. 3 Clouds tested: (a) cloud 1; (b) cloud 2; (c) cloud 3; (d) cloud 4 [7].

Testing

Due to the characteristics of the test clouds, testing the quality and effectiveness of vegetation filters using the MDSR algorithm through standard methods is a challenging task.

Due to their density and granularity, it is almost impossible to create a reliable reference terrain model against which Type I and Type II errors could be determined, and manual evaluation would be highly dependent on the individual approach of the evaluator and is thus associated with large uncertainty.

The method of comparing filtered data with control points measured by the GNSS or a total station is also inaccurate as the measured data are affected by different densities and accuracies.

Therefore, the method of visual comparison of the results of the different filtering algorithms was chosen. This comparison was made both by looking at the clouds as a whole and by looking at sections of the cloud at potentially problematic locations.

The results were also compared using the point deviations of the other clouds from the surface generated from the results of the MDSR algorithm. For this purpose, a TIN approximation of the terrain was first created from the MDSR filter results, from which unwanted edge relics were clipped. From this surface, the distances of the points above and below this surface were then calculated for each cloud, including the original cloud. Finally, the root mean square deviations (equation (1)) were calculated from the distances of the points left below or above the surface:

$$\mathrm{RMSD} = \sqrt{\frac{\sum d^2}{n}} \tag{1}$$

where d is the distance of one point from MDSR-TIN surface in meters and n is the number of points.

Points below the surface are only important for comparison with the input cloud, as these points were omitted during the identification of the lowest points. In contrast, points above the ground include all vegetation and other objects in the original cloud.

The RMS deviations of the points left above the MDSR-TIN surface are primarily used to evaluate the quality of the removal of vegetation points by the other filters.

All clouds were clipped with the same curve that was used to clip the cloud after the MDSR filter was applied, to remove points left on the edges in order to correctly calculate distances from the MDSR-TIN surface.

All cloud computation was performed in the freely available CloudCompare processing software [10].

DOI 10.13164/juniorstav.2024.24017

3 RESULTS

As mentioned earlier, the filter results are compared with those of the three other conventionally used vegetation filters, both visually and numerically. The three selected vegetation filters were the Cloth Simulation Filter (CSF) [4], the Simple Morphological Filter (SMRF) [5] and the Adaptive Triangulated Irregular Network filter (ATIN) [6].

Visual comparison

A view of the clouds as a whole (Fig. 4) and in sections (Fig. 5) was used to get a comprehensive picture of the efficiency and reliability of the filters used. The visual comparison will allow us to identify areas where successful vegetation filtering occurs as well as detect any errors or flaws more accurately in the process of identifying the terrain points of each filter.

Due to the limited length of the article, only a visual comparison of cloud 3 will be shown as it is the cloud with the most complex terrain and the densest vegetation and is thus the most difficult one in terms of ground point identification. The MDSR algorithm is primarily designed to filter such clouds.

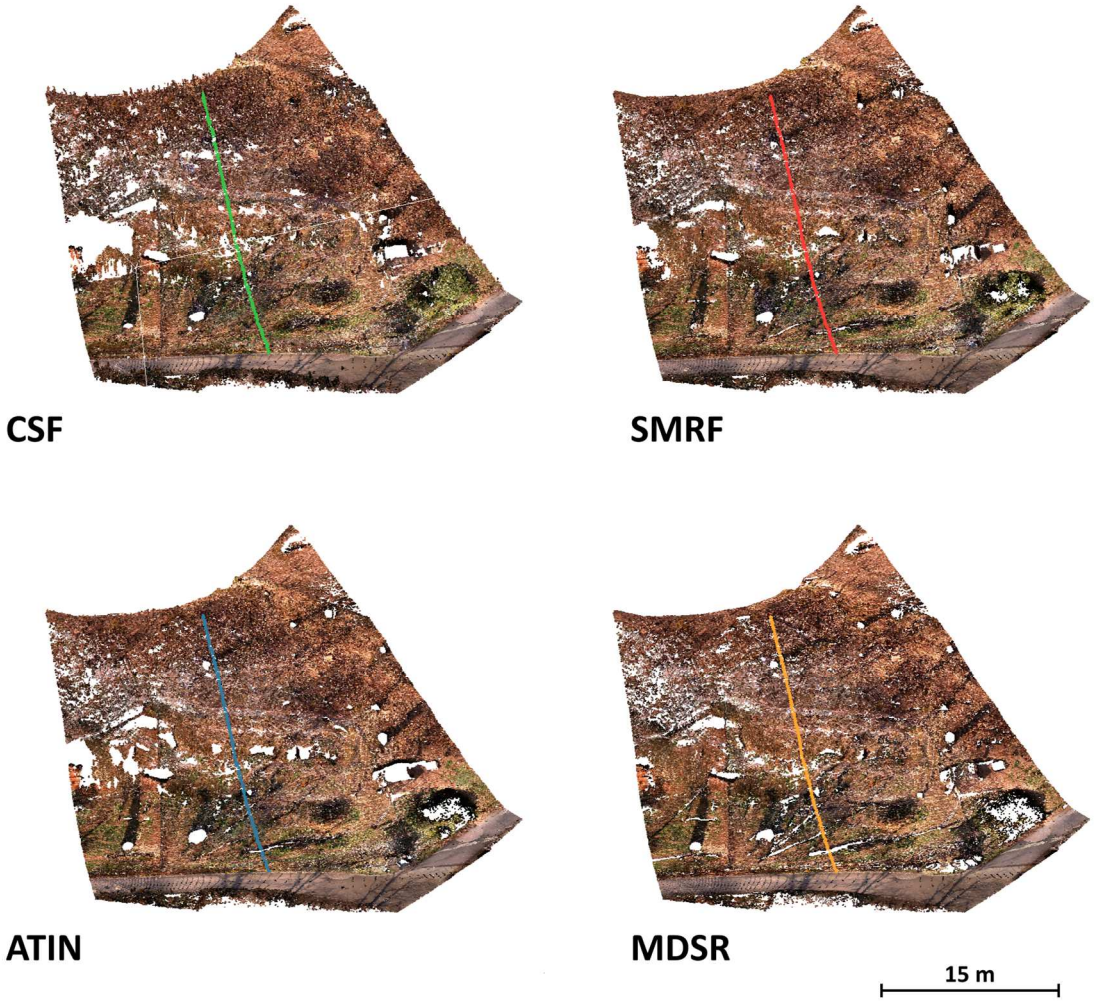

Fig. 4 Cloud 3 filtering results in whole.

DOI 10.13164/juniorstav.2024.24017

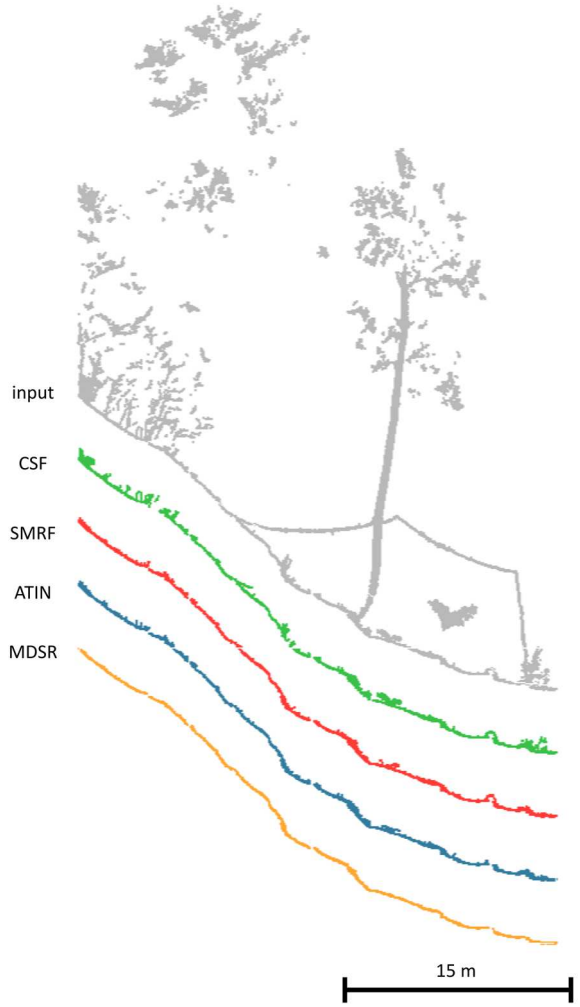

Fig. 5 Cloud 3 filtering results in section.

In the cloud sections (Fig. 5), it can be clearly seen that the best result was achieved by the MDSR filter, whose section is the smoothest and the most complete. In order to capture the best detail, a rotation angle of up to 120 gons was used to capture points below the overhang.

In this case, even the ATIN filter, whose results are similar for the other clouds, could not fully cope with the low vegetation and its remnants are visible in the section.

The CSF result shows the low vegetation left behind, which the filter cannot successfully deal with in any of the clouds tested.

Numerical comparison

The RMSDs for points of the input cloud below the MDSR-TIN surface were 6 mm for cloud 1, 11 mm for cloud 2, 7 mm for cloud 3, and 5 mm for cloud 4 (Tab. 1). These low values indicate that the MDSR filter was able to successfully detect the lowest points in the tested clouds. Similarly, the low RMSD values for points below the formed surface were achieved by the other filters. Thus, all filters were able to detect the lowest points very well.

Tab. 1 RMSD values [m] of points left above MDSR-TIN surface.

	original	**CSF**	**SMRF**	**ATIN**
1	*1.211*	*0.053*	*0.061*	*0.013*
2	*10.346*	*0.399*	*0.828*	*0.039*
3	*8.811*	*0.227*	*0.148*	*0.064*
4	*19.733*	*0.080*	*0.107*	*0.035*

DOI 10.13164/juniorstav.2024.24017

Tab. 2 RMSD values [m] of points left under MDSR-TIN surface.

	original	**CSF**	**SMRF**	**ATIN**
1	*0.006*	*0.006*	*0.003*	*0.006*
2	*0.011*	*0.012*	*0.006*	*0.013*
3	*0.007*	*0.007*	*0.003*	*0.002*
4	*0.005*	*0.006*	*0.003*	*0.008*

However, the difference occurs for the RMSD calculated for points above the MDSR-TIN surface. In this case, the RMSD values for the other conventionally used filters ranged from 1.3 to 82.8 cm (Tab. 2). These high values indicate that the other filters were not able to filter out vegetation points (and other anthropomorphic points) above the terrain as perfectly as the MDSR filter.

The same is already evident from the visual comparison where the profiles of the conventionally used filters are thicker and remnants of vegetation are visible above them.

The largest RMSD was calculated from the results of the second cloud. This large value is not only due to insufficient filtering of vegetation points, but also largely to imperfections in the input data which contain large gaps in some places and very few or almost no terrain points. If the input data is not of sufficient quality (sufficient terrain coverage), it is not surprising that the output of the vegetation filtering algorithm is not of sufficient quality either.

The best filtering algorithm is therefore clearly the MDSR. Comparably good results were then achieved by the ATIN filter.

4 DISCUSSION

The results of testing on the selected clouds demonstrated the high efficiency and robustness of the algorithm, which was able to cope with very complex terrain covered with dense vegetation. Compared to the three other conventional filters (CSF, SMRF and ATIN), the MDSR filter clearly achieved the best results. This includes both the visual comparison and in the comparison of cloud deviations from the MDSR-TIN surface. The second-best filter in our testing was the ATIN filter, which managed to filter the data with a similarly high efficiency. On the other hand, the CSF filter was the worst performer as it proved unable to reliably remove low vegetation for any of the clouds without also removing some areas of terrain.

Although the goal of the thesis was met, the MDSR filter itself is not perfect and still fails to address some of the problems commonly encountered with point clouds. For example, unwanted points remain at the edge of the resulting cloud which are not removed automatically and must be removed manually by the user. This is due to the principle of the algorithm. Or, if there were an erroneously detected outlier point in the cloud that was below the terrain contour, the MDSR filter would not be able to identify that point as erroneous. The resulting filtered cloud would therefore still contain these erroneous points.

As mentioned in [3], the MDSR filter could in the future function as part of a more complex vegetation filter that would use the MDSR only to estimate the terrain flow. Further improvements could also come from the active use of the information stored by this software implementation about the number of selections of each point during the algorithm.

5 CONCLUSION

In this paper, a software MDSR filter has been successfully implemented to enable fast and reliable identification of terrain points in point clouds.

The development of the MDSR software filter provides surveyors and cartographers with a fast and reliable way to identify terrain points in point clouds, which is crucial for a number of applications, such as the creation of digital terrain models, erosion analysis and forest management.

This implementation was done in the C++ programming language. The resulting program uses ASCII as the input and output format. The program also uses the principle of parallelization, which significantly speeds up the computation time of the program. In addition to the resulting filtered cloud, the program can also generate a log of the filtering process and the set parameters. For the points of the resulting cloud, information about the number of times each point was selected as the lowest by the algorithm is also stored. This information helps to understand and illustrate the operation of the algorithm. A higher number of times a point has been selected by the algorithm indicates greater confidence that the point is actually on the ground.

DOI 10.13164/juniorstav.2024.24017

References

[1] JAMES, David, Juergen ECKERMANN, Fawzi BELBLIDIA and Johann SIENZ. Point cloud data from Photogrammetry techniques to generate 3D Geometry. Proceedings of the 23rd UK Conference of the Association for Computational Mechanics in Engineering, Swansea, 8–10 April 2015, Swansea University. [Accessed 26 October 2023].
Available at:
https://www.researchgate.net/publication/274883792_Point_cloud_data_from_Photogrammetry_techniques_to_generate_3D_Geometry

[2] ZHOU, Lang, Guoxing SUN, Yong LI, Weiqing LI, Zhiyong SU. Point cloud denoising review: from classical to deep learning-based approaches. *Graphical Models* [online]. 2022, 121, 101140. [Accessed 26 October 2023]. ISSN 1524-0703.
Available at: https://doi.org/10.1016/j.gmod.2022.101140

[3] ŠTRONER, Martin, Rudolf URBAN and Lenka LÍNKOVÁ. Multidirectional Shift Rasterization (MDSR) Algorithm for Effective Identification of Ground in Dense Point Clouds. *Remote Sensing* [online]. 2022, 14(19), 4916. [Accessed 24 October 2023]. ISSN 2072-4292.
Avaible at: https://doi.org/10.3390/rs14194916

[4] ZHANG, Wuming, Jianbo QI, Peng WAN, Hongtao WANG, Donghui XIE, Xiaoyan WANG and Guangjian YAN. An Easy-to-Use Airborne LiDAR Data Filtering Method Based on Cloth Simulation. *Remote Sensing* [online]. 2016, 8(6), 501. [Accessed 24 October 2023]. ISSN 2072-4292.
Avaible at: https://doi.org/10.3390/rs8060501

[5] PINGEL, Thomas J., Keith C. CLARKE and William A. MCBRIDE. An improved simple morphological filter for the terrain classification of airborne LIDAR data. *ISPRS Journal of Photogrammetry and Remote Sensing* [online]. 2013, (77), 21-30. [Accessed 24 October 2023]. ISSN 0924-2716.
Avaible at: https://doi.org/10.1016/j.isprsjprs.2012.12.002

[6] NIE, Sheng, Cheng WANG, Pinliang DONG, Xiaohuan XI, Shezhou LUO and Haiming QIN. A revised progressive TIN densification for filtering airborne LiDAR data. *Measurement* [online]. 2017, (104), 70-77. [Accessed 24 October 2023]. ISSN 0263-2241.
Avaible at: https://doi.org/10.1016/j.measurement.2017.03.007

[7] KUČERA, Jakub. Software implementation of the MDSR filter and its testing on selected point clouds. Diploma thesis. Praha: Czech Technical University in Prague, Faculty of Civil Engineering [online]. 2023. [Accessed 24 October 2023]. Avaiable at: https://dspace.cvut.cz/handle/10467/110624

[8] OpenMP [online]. OpenMP, 2023 [Accessed 24 October 2023]. Avaiable at: https://www.openmp.org/

[9] Why did we build Visual Studio Code?. Visual Studio Code [online]. 2024. [Accessed 24 October 2023]. Avaiable at: https://code.visualstudio.com/docs/editor/whyvscode

[10] CloudCompare [online]. [Accessed 24 October 2023]. Avaiable at: https://www.cloudcompare.org/