# PROCESSING UAV AND LIDAR POINT CLOUDS IN GRASS GIS

V. Petras[a]*, A. Petrasova[a], J. Jeziorska[a,b] H. Mitasova[a],

[a] Department of Marine, Earth, and Atmospheric Sciences, North Carolina State University - (vpetras, akratoc, hmitaso @ncsu.edu)
[b] Department of Geoinformatics and Cartography, University of Wroclaw - (jajezior@ncsu.edu)

**Commission VII, SpS10 - FOSS4G: FOSS4G Session (coorganized with OSGeo)**

**KEY WORDS:** 3D rasters, decimation, sampling, binning, LAS, PDAL, PCL, Kinect

**ABSTRACT:**

Today's methods of acquiring Earth surface data, namely lidar and unmanned aerial vehicle (UAV) imagery, non-selectively collect or generate large amounts of points. Point clouds from different sources vary in their properties such as number of returns, density, or quality. We present a set of tools with applications for different types of points clouds obtained by a lidar scanner, structure from motion technique (SfM), and a low-cost 3D scanner. To take advantage of the vertical structure of multiple return lidar point clouds, we demonstrate tools to process them using 3D raster techniques which allow, for example, the development of custom vegetation classification methods. Dense point clouds obtained from UAV imagery, often containing redundant points, can be decimated using various techniques before further processing. We implemented and compared several decimation techniques in regard to their performance and the final digital surface model (DSM). Finally, we will describe the processing of a point cloud from a low-cost 3D scanner, namely Microsoft Kinect, and its application for interaction with physical models. All the presented tools are open source and integrated in GRASS GIS, a multi-purpose open source GIS with remote sensing capabilities. The tools integrate with other open source projects, specifically Point Data Abstraction Library (PDAL), Point Cloud Library (PCL), and OpenKinect libfreenect2 library to benefit from the open source point cloud ecosystem. The implementation in GRASS GIS ensures long term maintenance and reproducibility by the scientific community but also by the original authors themselves.

## 1. INTRODUCTION

Current methods of acquiring data to represent terrain or surface are often associated with large number of points in unordered point clouds. These points are collected non-selectively in case of lidar devices or generated during imagery processing as in the case of imagery collected by unmanned aerial vehicles (UAV). Not all of these points are necessarily important for creating a digital elevation model (DEM) of a given area (Brasington et al., 2012). Thus omitting the unnecessary points is desired since it is computationally challenging to process the acquired point clouds (Rychkov et al., 2012). Even with increasing hardware power, we are collecting larger points clouds which are more challenging to process.

There are two basic approaches for processing large number of points. The first approach is binning which creates a raster from a point cloud. Binning is a powerful analytical method which can be extended into 3D space in order to reveal vertical structure of vegetation (Gorte and Winterhalder, 2004). The second approach is decimation, also referred to as thinning or sampling, which reduces number of points of a point cloud. In this research, we are testing two decimation techniques, count-based decimation (Pirotti and Tarolli, 2010) and grid-based decimation, to see how they perform when applied to different types of data. Our objective is to determine how many points we can remove from a point cloud and still preserve enough details in the digital elevation model. Additionally, we compare the performance of the two techniques to evaluate whether there is a need for both techniques.

Geospatial methods can be implemented either as standalone tools or integrated into a larger software package. We want the implementation of our methods to be accessible in the long-term, and available for further review and improvement. Furthermore, the methods should be easily used with other geospatial processing tools. For these reasons, we use GRASS GIS (Neteler et al., 2012), a free, libre and open source geospatial information system, for our geospatial processing. We also implemented the methods presented in this paper, namely count-based decimation, grid-based decimation and three dimensional binning, in GRASS GIS.

*Corresponding author

## 2. DATA

For this study we are using four different point cloud types: airborne lidar and ground-based lidar point clouds, points from a low-cost indoor scanner (the Microsoft Kinect), and a point cloud derived from data obtained from UAV imagery using structure from motion (SfM) technique. Our study site is a rural area south of Raleigh, North Carolina, USA. The data for our study area, the Sediment and Erosion Control Research and Education Facility (SECREF) at the Lake Wheeler Road Field Laboratory of North Carolina State University, were collected in 2013 by Wake county with airborne lidar. The point cloud was classified by the data provider. We used only points classified as ground (class 2) for our study. The ground-based lidar measurements were done in 2009 on a small part of the SECREF site (Starek et al., 2011) using Leica Geosystems ScanStation 2. The measurement was done from two sites and the point clouds were merged together. The data obtained by the Kinect scanner capture a scaled physical model molded from sand. The 0.37 m $\times$ 0.35 m mold was derived from the airborne lidar data for the SECREF site. The data generated by SfM from UAV imagery are from a location west of the SECREF site and not from the SECREF site itself due to the limits on where the UAV could be flown (Jeziorska et al., 2016).

To understand the spatial distribution of points in the point cloud we look at the number of points per raster cell. Figure 1 shows

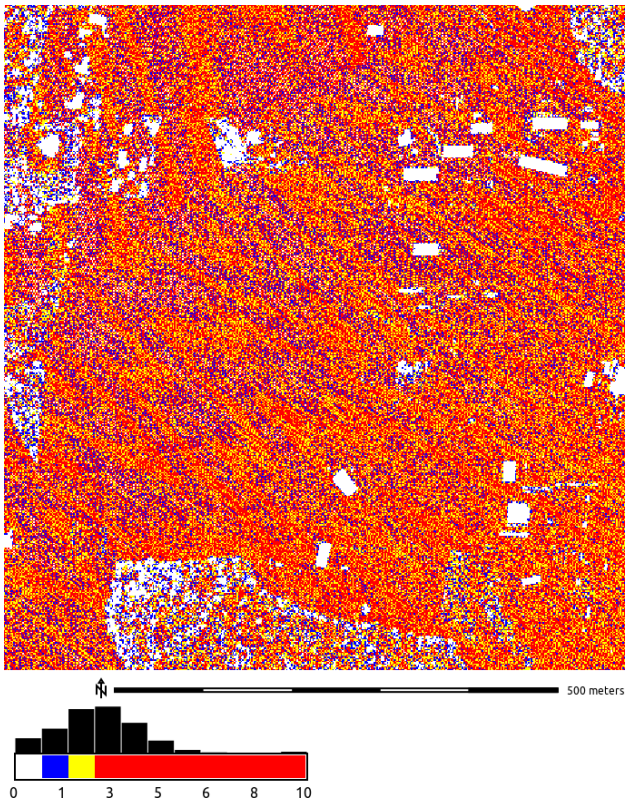


Figure 1: The number of points per cell for point cloud obtained using airborne lidar (raster resolution 1.5 m)

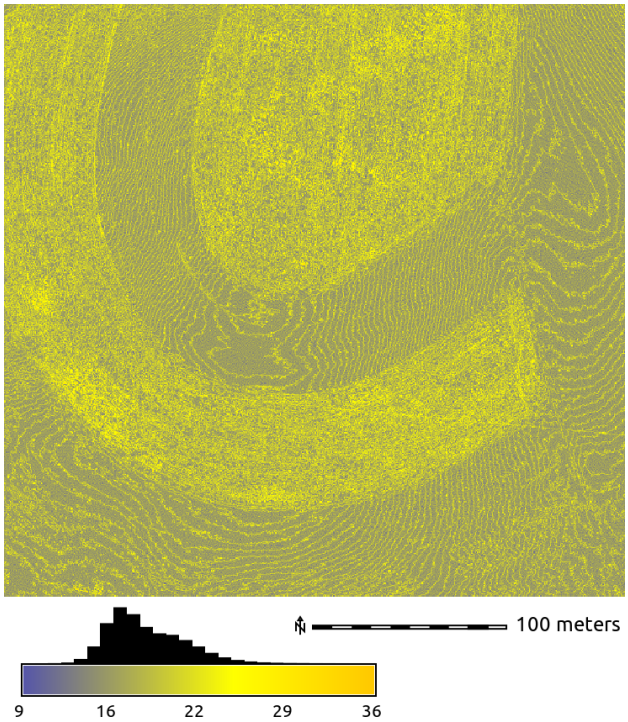


Figure 2: The number of points per cell for point cloud obtained using UAV imagery (raster resolution 0.5 m)

uniform distribution of the points classified as ground in the airborne lidar dataset. The only non-uniformities we can see are caused by presence of buildings (places without any points), vegetation (places with lower point density) and the scanning pattern (lines of slightly higher and lower density). Figure 2 shows the spatially variable number of points per cell for the the UAV dataset which is primarily caused by the variable vegetation cover (areas with and without crops) and by artifacts from the SfM processing step. Figure 3 shows the number of points per cell for the ground-based lidar measurement. The highest density of points is in 10 m radius around the ground-based lidar stations. Figure 4 reveals the regular, grid pattern of the point cloud acquired by the Kinect scanner.

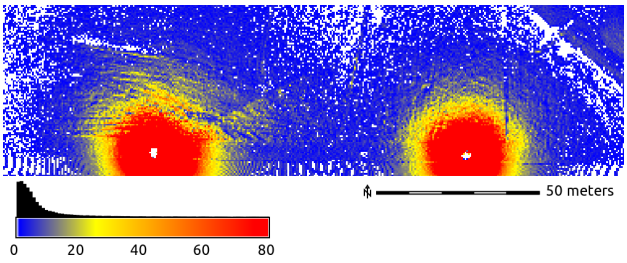


Figure 3: The number of points per cell for point cloud obtained using ground-based lidar (raster resolution 0.5 m). Note that the color table uses red color for values from 80 up to the maximum which is 18 thousand points per cell.

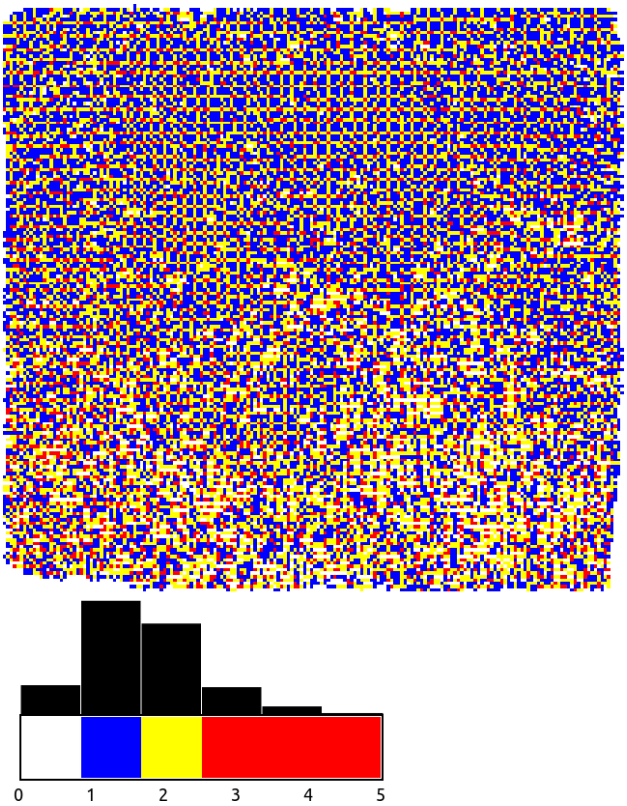


Figure 4: The number of points per cell for point cloud obtained using Kinect scanner (0.37 m × 0.35 m, raster resolution 0.002 m)

## 3. APPROACH

In this section, we first reiterate why we choose an open source implementation in GRASS GIS. We follow with an overview of how point clouds are processed in GRASS GIS highlighting newly implemented 3D binning and integration with other open source projects. Then, we explain the binning process in detail

which is also important as a base for the grid-based decimation technique. Further, we discuss both newly implemented techniques for decimating point clouds, namely count-based decimation and grid-based decimation. Finally, we discuss our approach for comparing decimated point clouds.

### 3.1 Open source implementation

We want to assure that the newly developed methods and other methods used in this study can be reviewed. To review a method's implementation, its source code must be publicly accessible. In order to build or improve upon the source code, the code should be available under an open source[1] license. Furthermore, we want to ensure that all methods we are using and developing will remain accessible in the long term. GRASS GIS has long history of preserving algorithms over several decades (Chemin et al., 2015) with changes and updates to reflect new hardware, platforms, types of data and user needs. The implementation of the methods in GRASS GIS ensures long-term accessibility thanks to an active community which has now worked on GRASS GIS for more than 30 years. We can also see a wide variety of tools being constantly[2] added. In order to build complex analytic workflows, we can take advantage of the wide range of geospatial tools offered by GRASS GIS such as remote sensing (Neteler and Mitasova, 2008) or spatio-temporal data processing tools (Gebbert and Pebesma, 2014), and use them together with our newly developed methods.

### 3.2 Processing point clouds in GRASS GIS

The first step in the typical point cloud processing workflow in GRASS GIS is to explore the point cloud by counting the number of points per raster cell using *r.in.lidar* module[3] with different resolutions. We determine the spatial distribution and density of points of different classes and the optimal resolution for further analysis.

Further processing typically consists of computing digital elevation model, height above surface, and several other statistics based on point count, height of points, and intensity of points. This functionality is available in *r.in.lidar* module for 2D rasters and in *r3.in.lidar* for 3D rasters. Both 2D and 3D rasters can be further processed by corresponding processing tools. The 3D raster can also be decomposed into series of 2D rasters to take advantage of the standard tools for image processing and classification available in GRASS GIS.

To create a smooth surface based on a point cloud, we can import the points to GRASS GIS using *v.in.lidar* module which creates vector points. These points can be later interpolated to create a smooth surface typically representing digital elevation model. Several interpolation methods are available for example, inverse distance weighting implemented in *v.surf.idw* module, bicubic and bilinear spline interpolation with Tykhonov regularization (Brovelli et al., 2002) both implemented in *v.surf.bspline* module, and regularized spline with tension technique (Mitasova et al., 2005) implemented in *v.surf.rst* module. The aforementioned lidar modules can limit the import through a number of parameters including spatial extent, area, and return number or class if these are available.

[1] We understand open source as defined by the Open Source Initiative at `opensource.org/osd`.

[2] At the time of writing last added GRASS GIS add-on module called *r.randomforest* was added less than a week ago.

[3] The individual programs, tools, plug-ins, and functions in GRASS GIS are called modules.

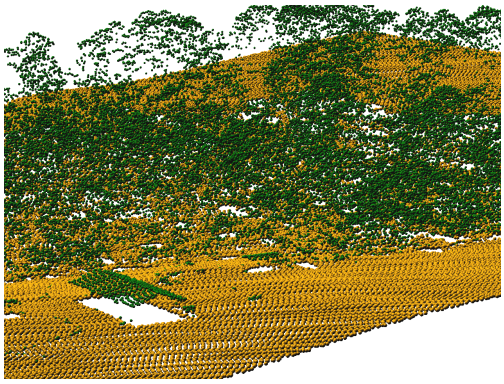

Figure 5: Point cloud obtained using airborne lidar in a small selected area with a patch of trees in the middle and a building (same as in Figure 6). Points classified as ground by progressive morphological filtering from PDAL are in yellow; all the other points are in green.

Besides GRASS GIS functionality we can take advantage of methods and tools implemented in other open source packages. We present a prototype of *v.in.pdal* module which integrates several features from the Point Data Abstraction Library (PDAL) including progressive morphological filter for ground point classification (Zhang et al., 2003) from the Point Cloud Library (PCL). This method can be used as an alternative to the multiscale curvature classification algorithm (Evans and Hudak, 2007), implemented in *v.lidar.mcc* add-on[4] module, and the edge-based lidar data filtering method (Brovelli et al., 2004), implemented in *v.lidar.edgedetection*, *v.lidar.growing*, and *v.lidar.correction* modules in GRASS GIS. An example result from the progressive morphological filter is in Figure 5.

For reading data from the low-cost indoor scanner, Microsoft Kinect, we use a module called *r.in.kinect* which uses OpenKinect libfreenect2 library (Xiang et al., 2016) to communicate with the device. This module is used in Tangible Landscape system which is a collaborative modeling environment for analysis of terrain changes coupling a scanner, projector and a physical 3D model with GRASS GIS (Petrasova et al., 2015).

### 3.3 Binning

Binning is the conversion of points into a regular grid. The binning of points with X and Y coordinates starts with the overlay of a grid of bins over the points. A bin can be a rectangle, hexagon or generally any shape which can create continuous grid. When creating a raster from a point cloud, the bin is a raster cell which is square or rectangular. The value associated with a bin is the count of points falling into the given bin (Lewin-Koh, 2011). The analogous concept in univariate statistics (one dimension – 1D) is a histogram, so the result of this binning can be called 2D (two dimensional) histogram. We use binning to count the number of points per raster cell to see the density of points.

The concept of binning can be extended when the points have another value associated with them. For lidar data this value can be the Z coordinate or intensity. The value for a bin is computed as univariate statistics from the values of all points in the bin. For example, computing the mean value of Z coordinates yields a raster representing the digital elevation model. Another example is the range of Z coordinates which can be used as a rough estimate of vegetation height.

[4] Add-on modules are not part of the standard installation of GRASS GIS but can be installed separately using the *g.extension* module.

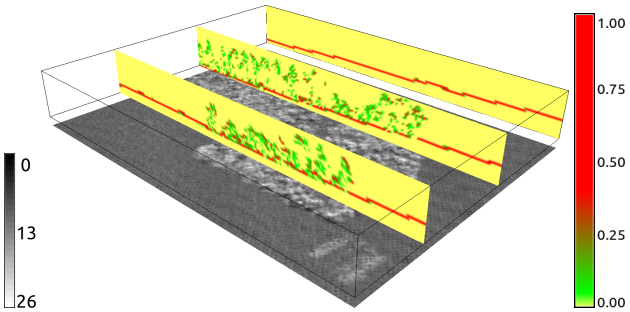


Figure 6: The result of 3D binning a point cloud obtained using airborne lidar (using all classes) in a small selected area with a patch of trees in the middle. The visualization shows slices of the resulting 3D raster. The variable visualized in yellow, green, and red is the proportional count of points per vertical column where yellow means 0% of points from a vertical column are in the given 3D cell (voxel) and red means that more than 40% of points from a given vertical column are in the given 3D cell. The horizontal plane is colored in gray scale according to the number of points per 2D cell which is a result of 2D binning at 2 m resolution.

Binning into a 3D (three dimensional) grid of cubes (rectangular cuboids) creates a 3D raster. Same statistics can be computed as in the case of 2D raster but not from the Z coordinate as it is used to determine position in the 3D grid in the same way as X and Y coordinates are used. By dividing bin values in each vertical column by value of a corresponding bin from a 2D binning result, we obtain the proportional (relative) value of a given variable that is not influenced by relative point density. The example result for the selected section of the airborne lidar dataset is shown in Figure 6 which represents vegetation structure as vertical slices of a 3D raster.

### 3.4 Count-based decimation

Count-based decimation is the removal (or preservation) of every $n$-th point based on its ordinal number in the set of points. Depending on the dataset structure and selected $n$, such point selection may be biased and does not ensure spatialy homogenious decimation. The advantage of this method is, however, its simplicity resulting in very fast point processing.

Specifying value $r$ to "remove every n-th point" or value $k$ to "keep every n-th point" is reasonably straighforward. For example, it is clear that removing every third point results in removal of one third of points. However, when we need to preserve a given percentage $p$ of points, specifying the right values $r$ and $k$ gets more complicated, because we can remove only certain fractions of points. Some percentage values can be easily converted to $r$ and $k$ values as visible in the Table 1. For a specific percentage $p$ of points the following equations can be used to obtain approximate values of $r$ and $k$. For $p >= 0.5$ the value $r$ is:

$$r = \left\lfloor \frac{1}{1-p} + 0.5 \right\rfloor \tag{1}$$

where $\lfloor a \rfloor$ is the whole part of $a$ obtained by trimming the decimal part of the number $a$.

For $p < 0.5$ the value $k$ is:

$$k = \left\lfloor \frac{1}{p} + 0.5 \right\rfloor \tag{2}$$

| $p$ | $r$ | $p$ | $k$ |
|---|---|---|---|
| $16.\overline{6}\%$ | 6 | $66.\overline{6}\%$ | 3 |
| 20.0% | 5 | 75.0% | 4 |
| 25.0% | 4 | 80.0% | 5 |
| $33.\overline{3}\%$ | 3 | $83.\overline{3}\%$ | 6 |
| 50.0% | 2 | 90.0% | 10 |

Table 1: Values of $r$ and $k$ for selected fractions of removed points $p$ in percent

### 3.5 Grid-based decimation

2D grid-based decimation of points creates a subset of spatially uniformly distributed points with representative heights. This subset of points can later be used to interpolate smooth digital terrain model and topographic parameters. When binning vector points to raster, Z coordinates of points in a grid cell are counted, summed, averaged or processed in other ways. To perform the grid-based decimation, we average the X and Y coordinates of all the points in the given grid cell to get a position of a new point. Average is defined as a sum of all values divided by the number of values. However, considering the often large numbers representing horizontal coordinates, summing all the X or Y coordinates of all points in a grid cell could result in truncating the sum and loosing precision. For this reason, we use cumulative moving mean[5] which requires us to store only one value rescaled by the number of values we processed up to that point. Cumulative moving mean $c$ in $i$-th step is defined as:

$$c_i = c_{i-1} + \frac{x_i - c_{i-1}}{i} \tag{3}$$

where $c_i$ is cumulative mean in the current step, $c_{i-1}$ is cumulative mean in the previous step, $x_i$ is the current value and $i$ is the current step number.

### 3.6 Comparing decimated point clouds

To compare the two newly implemented decimation techniques, we decimated the given point cloud using count-based decimation and grid-based decimation with different settings influencing the number of preserved points. We interpolated a surface from each decimated point cloud using regularized spline with tension technique (Mitasova et al., 2005) as implemented in the *v.surf.rst* module. We are interested in how the distinct terrain features are preserved when lower number of points is used. To identify these features we use local relief model (LRM) extraction algorithm (Hesse, 2010), implemented in the *r.local.relief* add-on module which identifies terrain features based on their difference to terrain trend. The example of LRM enhancing digital elevation model visualization is shown in Figure 7.

We decimated each point cloud using the count-based method several times while changing number of removed points from less progressive removal to more progressive removal. Similarly, we decimated the point clouds using the grid-based method with coarser resolution each step. After interpolating and computing the local relief models, we correlated the local reliefs with the local relief from the original point cloud using the *r.covar* module which computes correlation matrix of given rasters.

[5] Wikipedia: Moving average, last revision September 18, 2015, accessed November 18, 2015

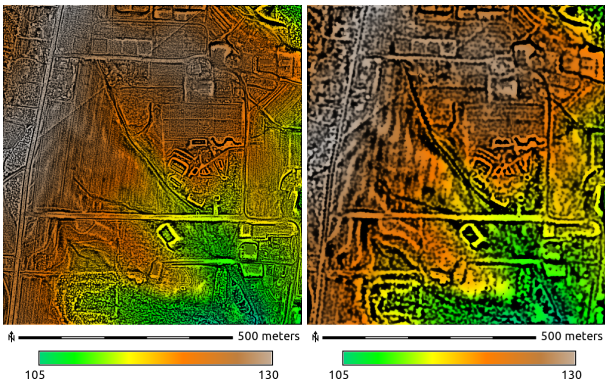


Figure 7: An example of a terrain interpolated from airborne lidar point cloud. The left image shows terrain interpolated from the original point cloud. The right image shows terrain interpolated from the point cloud decimated using count-based decimation with preserving every $10^{th}$ point. The colors shows elevation while shading is done using the local relief model. Darker areas denote terrain features below the trend surface, while lighter areas denote terrain features above the trend surface.

## 4. RESULTS

We compared the two newly implemented decimation techniques, count-based decimation and the grid-based decimation by correlating LRMs as described in Section 3.6. Figures 8, 9, 10, and 11 show how the correlation coefficient is affected by the percentage of removed points during the decimations for the four datasets described in Section 2. We performed all the interpolations and local relief model computations at resolution 0.5 m with the exception of the scaled sand model dataset where we used resolution 0.001 m.

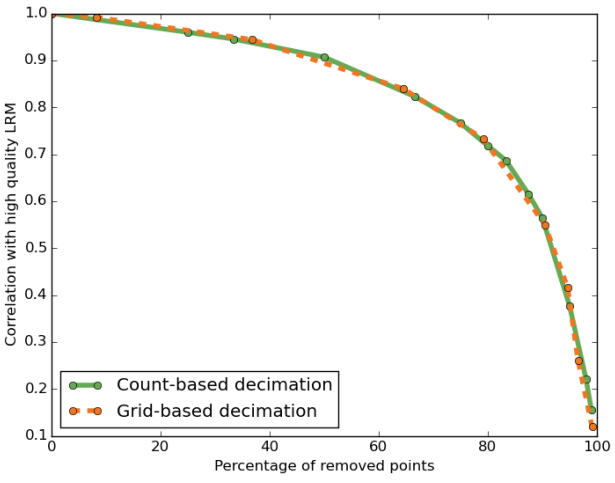


Figure 8: Comparison of the performance of count-based and grid-based decimation applied to a point cloud obtained using airborne lidar

Figure 8 shows that the correlation of the original LRM and the LRM from the count-based and grid-based decimated point clouds derived from the airborne lidar dataset decays in a very similar way as the number of points decreases. The correlation decay for the UAV dataset is much slower at the given resolution compared to the airborne lidar dataset. Figure 9 shows that the results are similar for count-based and grid-based decimations. The grid-based decimation performed worse than the count-based

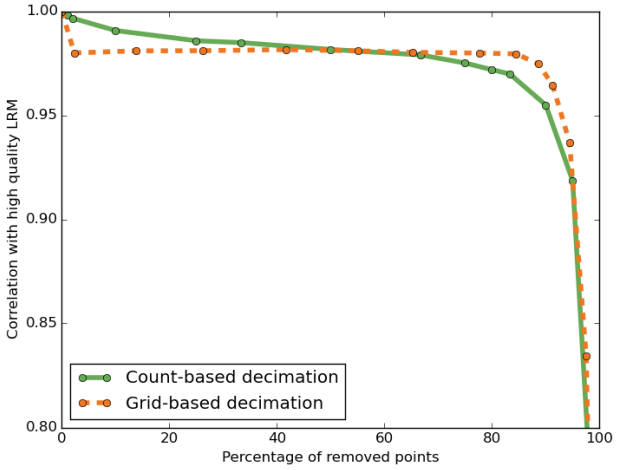


Figure 9: Comparison of the performance of count-based and grid-based decimation applied to a point cloud obtained using UAV imagery

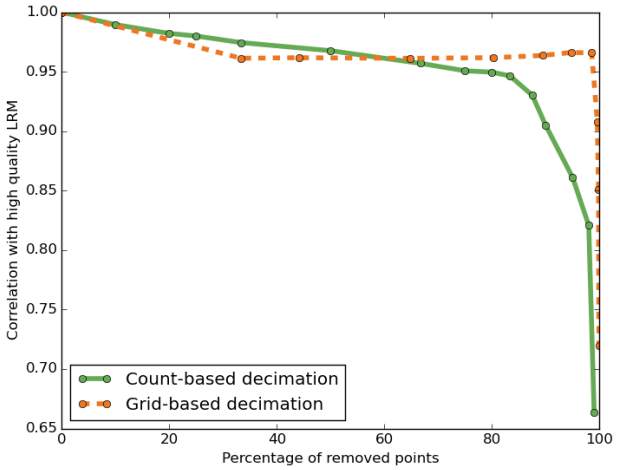


Figure 10: Comparison of the performance of count-based and grid-based decimation applied to a point cloud obtained using ground-based lidar

decimation with a low number of removed points, but performed better with a high number of removed points. For the ground-based lidar dataset, the correlation for the grid-based decimation is high even with a large number of removed points as Figure 10 shows. With the Kinect scanner dataset, the differences between count-based and grid-based decimations are also not significant, although the correction for grid-based decimations decays in more predictable way. The behavior is similar to UAV dataset and also partially to the airborne lidar dataset as is visible in Figure 11.

Figures 8, 9, 10, and 11 show that we can remove significant number of points (60-90%) from the original point clouds and still preserve large number of features according to the correlation with LRM of the original dataset. This applies to both count-based and grid-based decimations. These results are consistent with the previous studies on lidar data (Singh et al., 2015) which also showed that a significant portion of the points can be removed. In the case of ground-based lidar, we can even remove more than 95% points using grid-based decimation and still get high correlation as Figure 10 shows. This is caused by the extremely high density of points near the scanning sites.

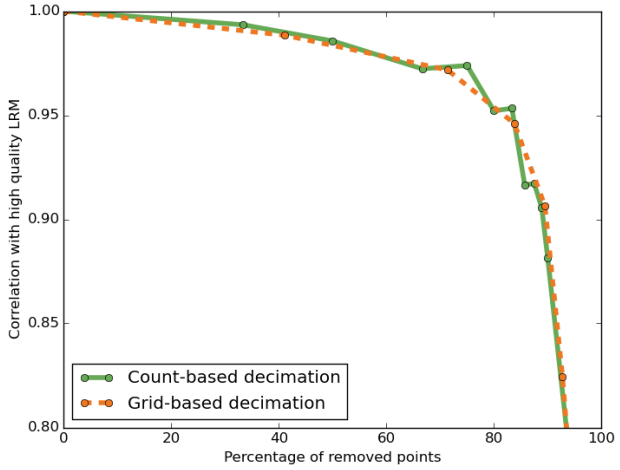


Figure 11: Comparison of performance of count-based and grid-based decimation applied to a point cloud obtained using the Kinect scanner

## 5. DISCUSSION

In the count-based decimation, the order of points in the input dataset influences which points are removed. This is simple to implement, but it can bias the sampling. The closest alternative decimation method to count-based decimation is the removal of a certain percentage of points $p$ where points are removed randomly by comparing a (pseudo) random number for each point with the probability $p$. This decimates the point cloud independent of the order of the points but the processing cost is higher due to random number generation. The count-based decimation, on the other hand, only requires the count of processed points be kept. The low processing cost of the count-based decimation is even more pronounced in comparison with the grid-based decimation which requires us to keep track of the cells with points and to compute mean for each cell.

The presented implementation of the count-based decimation limits the choices user have in decimating between 30% and 70% of the points, namely the choices are skip every third point, skip every second point, or preserve every third point. This may limit what we can infer from the graphs in previous sections about Figure 10. The assumption is that this is not an issue for the actual users, as one needs fine settings close to 0% and especially close to 100% of removed points while coarser changes are sufficient around 50% of removed points.

## 6. CONCLUSIONS

We presented a suite of tools in GRASS GIS for processing point clouds from different sources, namely airborne lidar, SfM applied to UAV imagery, ground-based lidar, and the Kinect scanner. We tested the tools with datasets representing each of these categories with focus on the newly developed tools for decimation of point clouds. The tests show that count-based decimation yields same results as the grid-based decimation for the tested point clouds obtained by airborne lidar, UAV, and the Kinect scanner. Therefore, it is more advantageous to use faster and simpler count-based decimation for point clouds when we can assume a spatially uniform distribution of points. When point clouds have spatially variable point density, such as point clouds from ground-based lidar scans, we can benefit from grid-based decimation which ensures that areas with extremely high density of points contain only the desired number of points after the decimation. At the same time, all points are preserved in areas with extremely low point density. The grid-based decimation can thus reduce the processing time of subsequent steps while keeping the quality of the outputs. This shows us the need for both presented decimation techniques to be included in GRASS GIS alongside with the tools currently available. Additionally, we presented two new GRASS GIS modules which integrate PDAL point cloud processing tools and employ 3D binning to create a 3D raster representation of vertical vegetation structure.

## SOFTWARE AVAILABILITY

GRASS GIS is free, libre and open source[6] software licensed under GNU General Public License 2 or higher (GNU GPL) and is available for download online. The count-based decimation was implemented by Vaclav Petras as part of GRASS GIS module *v.in.lidar*. The grid-based decimation also authored by Vaclav Petras was experimentally included as part of *r.in.lidar* but in the future it will be part of a separate module. These extended versions of modules are currently available in the development version of GRASS GIS and we expect them to be part of the next minor release, namely version 7.2. The *r3.in.lidar* module newly developed by Vaclav Petras will be part of the same release. The *v.in.pdal* module source code is included in the development version of GRASS GIS. All the other presented tools are already available in the latest stable release of GRASS GIS (currently 7.0.3). Both the latest release and the daily build of the development version of GRASS GIS can be obtained from the official GRASS GIS website[7] hosted by Open Source Geospatial Foundation (OSGeo). GRASS GIS module *r.in.kinect* developed by Anna Petrasova licensed under GNU GPL 3 is available in a Git repository[8] hosted on GitHub. The libfreenect2 library is part of OpenKinect[9] project and is available in one of the project source code repositories. PDAL[10] and PCL[11] are licensed under BSD licenses and both the source code and the installable binaries are available from the corresponding project websites.

## ACKNOWLEDGEMENTS

We are grateful to the GRASS development team for maintaining and developing GRASS GIS software package. We would like to acknowledge especially Markus Metz who implemented significant part of the presented functionality related to processing lidar data in GRASS GIS. We would also like to thank to GRASS GIS users, especially to Douglas J. Newcomb, Markus Neteler, and William W. Hargrove, for feedback and testing. The UAV imagery was collected by NextGen Air Transportation (NGAT), Institute for Transportation Research and Education, North Carolina State University.

## REFERENCES

Brasington, J., Vericat, D. and Rychkov, I., 2012. Modeling river bed morphology, roughness, and surface sedimentology using high resolution terrestrial laser scanning. *Water Resources Research*. W11519.

[6] GRASS GIS fulfills the definition of open source by Open Source Initiative and also free software definition by Free Software Foundation.

[7] `grass.osgeo.org`

[8] `github.com/ncsu-osgeorel/r.in.kinect`

[9] `openkinect.org`

[10] `pdal.io`

[11] `pointclouds.org`

Brovelli, M. A., Cannata, M. and Longoni, U. M., 2004. LIDAR data filtering and DTM interpolation within GRASS. *Transactions in GIS* 8(2), pp. 155–174.

Brovelli, M., Cannata, M. and Longoni, U., 2002. Managing and processing LIDAR data within GRASS. In: *Proceedings of the GRASS users conference*, Vol. 29.

Chemin, Y., Petras, V., Petrasova, A., Landa, M., Gebbert, S., Zambelli, P., Neteler, M., Löwe, P. and Di Leo, M., 2015. GRASS GIS: a peer-reviewed scientific platform and future research repository. In: *Geophysical Research Abstracts*, Vol. 17, p. 8314.

Evans, J. S. and Hudak, A. T., 2007. A multiscale curvature algorithm for classifying discrete return LiDAR in forested environments. *Geoscience and Remote Sensing, IEEE Transactions on* 45(4), pp. 1029–1038.

Gebbert, S. and Pebesma, E., 2014. A temporal GIS for field based environmental modeling. *Environmental Modelling & Software* 53, pp. 1–12.

Gorte, B. and Winterhalder, D., 2004. Reconstruction of laser-scanned trees using filter operations in the 3d raster domain. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 36(Part 8), pp. W2.

Hesse, R., 2010. LiDAR-derived Local Relief Models–a new tool for archaeological prospection. *Archaeological Prospection* 17(2), pp. 67–72.

Jeziorska, J., Mitasova, H., Petrasova, A., Petras, V., Divakaran, D. and Zajkowski, T., 2016. Overland flow analysis using time series of sUAS-derived elevation models. *Submitted to ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*.

Lewin-Koh, N., 2011. Hexagon binning: an overview. http://cran.r-project.org/web/packages/hexbin/vignettes/hexagon_binning.pdf (2 Mar. 2016).

Mitasova, H., Mitas, L. and Harmon, R. S., 2005. Simultaneous spline approximation and topographic analysis for lidar elevation data in open-source GIS. *Geoscience and Remote Sensing Letters, IEEE* 2(4), pp. 375–379.

Neteler, M. and Mitasova, H., 2008. *Open Source GIS: A GRASS GIS Approach*. Vol. 773, Springer, New York.

Neteler, M., Bowman, M. H., Landa, M. and Metz, M., 2012. GRASS GIS: A multi-purpose open source GIS. *Environmental Modelling & Software* 31(0), pp. 124130.

Petrasova, A., Harmon, B., Petras, V. and Mitasova, H., 2015. *Tangible Modeling with Open Source GIS*. Springer.

Pirotti, F. and Tarolli, P., 2010. Suitability of lidar point density and derived landform curvature maps for channel network extraction. *Hydrological Processes* 24(9), pp. 1187–1197.

Rychkov, I., Brasington, J. and Vericat, D., 2012. Computational and methodological aspects of terrestrial surface analysis based on point clouds. *Computers & Geosciences* 42, pp. 6470.

Singh, K. K., Chen, G., McCarter, J. B. and Meentemeyer, R. K., 2015. Effects of LiDAR point density and landscape context on estimates of urban forest biomass. *ISPRS Journal of Photogrammetry and Remote Sensing* 101, pp. 310–322.

Starek, M. J., Mitasova, H., Hardin, E., Weaver, K., Overton, M. and Harmon, R. S., 2011. Modeling and analysis of landscape evolution using airborne, terrestrial, and laboratory laser scanning. *Geosphere* 7(6), pp. 1340–1356.

Xiang, L., Echtler, F., Kerl, C., Wiedemeyer, T., Lars, Gordon, R., Facioni, F., Wareham, R. and et al., 2016. libfreenect2: Release 0.2 RC2.

Zhang, K., Chen, S.-C., Whitman, D., Shyu, M.-L., Yan, J. and Zhang, C., 2003. A progressive morphological filter for removing nonground measurements from airborne LIDAR data. *Geoscience and Remote Sensing, IEEE Transactions on* 41(4), pp. 872–882.

## APPENDIX

This section lists selected commands (GRASS GIS module calls) we used to create results presented in this paper. These commands can be used in the GRASS GIS command line or they can be executed using graphical user interface (GUI). Each command starts with the name of the module followed by individual parameters separated by spaces. Parameters are in format *key=value* where *key* can be found in the GUI form of the given module and *value* is the number or text written into the associated input field. Some special parameters are specified as letter prefixed by a dash, for example *-b* where *b* is a name associated with a check box in the module GUI form. Dash followed by multiple letters is the same as writing each letter with a separate dash.

To count the number of points per raster cell, we used *r.in.lidar* module with *method=n*:

```
r.in.lidar input=input output=name method=n
```

To decimate the point cloud we used the *v.in.lidar* module with parameters *skip=num* or *preserve=num* where *num* is the number of points to skip or preserve in the dataset. This is how the command looks like without any decimation applied:

```
v.in.lidar -bcr input=points.las output=points
```

In case of airborne lidar dataset we used also *class_filter=2* to select only points classified as ground. We experimentally integrated the implementation of grid-based decimation into the *r.in.lidar* module and enabled it using *vector_output* parameter.

```
r.in.lidar -b input=input vector_output=points
```

To get different decimations we used the parameter *resolution* which sets the resolution of the grid used for decimation. For interpolation we used the *v.surf.rst* module:

```
v.surf.rst input=points elevation=elevation
```

There are several parameters which influence the resulting surface, we used *npmin=120*, *tension=20*, and *smooth=2*. For the purpose of this study, we disabled, using *dmin=0*, decimation performed by the *v.surf.rst* module internally. In certain cases, we had to leave it enabled to reduce number of points considered for the interpolation. The local relief model was computed using the *r.local.relief* module:

```
r.local.relief input=elevation output=lrm
```

In addition to the *output* parameter which is useful for further processing, *shaded_output* can be used to obtain colored local relief model with shading which is advantageous for visualization. To create the relative point density 3D raster we used *r3.in.lidar* with the *proportional_n* parameter:

```
r3.in.lidar input=points.las proportional_n=output
```

Additional parameters for the *r3.in.lidar* module are *n*, *sum*, *mean*, and *proportional_sum*. To classify points as ground, we used *v.in.pdal*:

```
v.in.pdal -j input=points.las output=ground
```

The parameter *-j* extracts only the ground points. To get points which are not ground, we used the same command, but with *-k* parameter to classify ground points and *class_filter=0* to extract only the points not classified as ground. To get points scanned by the Kinect scanner, we used *r.in.kinect* module with parameter *vector* to get the raw point cloud:

```
r.in.kinect vector=points numscan=1 zexag=3
```

We used points from one scan, the physical model was 3 times exaggerated, and we applied smoothing using *smooth_radius=0.009*. Additionally, we set parameters related to calibration of the particular scanning setup. Since the binning implemented in *r.in.lidar* requires the data to be in the LAS format, we needed to convert some of the data using the *v.out.lidar* module into the LAS format:

```
v.out.lidar input=points output=points.las
```

The data for the physical model were in their original scale, so we had to use unusually low scaling for their storage in the LAS format, namely we used parameter *las_xyscale=0.00001* and also *las_zscale=0.00001*.

In addition to processing, we did all 2D and 3D geospatial visualization in GRASS GIS.