Kartografické listy / Cartographic letters, 2025, 33 (1), 3-18

EDGE DETECTION USING TRIANGULAR NETWORK DECIMATION

Veronika HAJDÚCHOVÁ, Richard FECISKANIN

Edge detection using triangular network decimation

Abstract: Edge detection is an important part of various tasks in the geographic information systems, photogrammetry, and computer vision. Existing methods mostly work on 2D raster images, where different algorithms are used to examine changes or discontinuities in the color or saturation of pixels. In this article, we present a method and script that is capable of edge detection on 3D triangulated irregular network (TIN) models. This approach works with the triangle normals, calculating the angle between the normals of adjacent triangles. Angle between triangles can show inclination differences of adjacent triangles, so if the angle is smaller than given threshold value, the adjacent triangles have small plane inclination difference, which means that those triangles belong to the same plane, but when there is bigger angle, it can indicate discontinuity – edge on model.

Keywords: edge detection, digital elevation model, triangular irregular network, 3D mesh, QECD, Python

Introduction

Edge detection is one of the most important tasks in the field of computer vision and image processing. Edges provide essential information for many subsequent tasks, such as image recognition, segmentation, corner and road detection and identification of discontinuities. These tasks rely on extracting object boundaries or identifying prominent edges from original images (Jing et al., 2021; Igbinosa, 2013). Edge detection identifies areas with significant changes in color intensity (or brightness). Such discontinuities often correspond to discontinuities in depth, discontinuities in surface orientation, changes in material properties and variations in scene illumination (Das, 2016; Pu et al., 2021).

Though extensively used in computer graphics, edge detection is also relevant in geoinformatics due to the reliance on raster images, including satellite imagery and digital elevation models (DEMs). Edge detection can highlight terrain features, building footprints and boundaries of other elements. As noted by Minár and Evans (2008), Pacina (2008), identifying boundaries of elementary forms is a good starting point for land surface segmentation.

Most edge detection operators are applied to images by comparing neighbouring pixel values and their change. Main division and description of most used edge detection methods on images were introduced in Jing et al. (2022) that categorized edge detection methods to two main streams hand-crafted based methods (e.g., Sobel operator, Prewitt operator or Canny operator (Igbinosa, 2013; Miller, 2015)) and machine learning-based methods (e.g., ResNet, holistically-nested edge detection).

While these approaches target image data, the growing use of LiDAR (Light Detection and Ranging) prompted research into applying or adapting such algorithms for point clouds or on the interpolated models and whether the given algorithms are effective for edge detection, or they need to be modified or improved. For example, improvement of the Canny algorithm was presented by Xu et al. (2006) where the authors tried to modify this one algorithm in order to remove some of the shortcomings studied by them. The improved algorithm divides image into a number of sub-images

Mgr. Veronika HAJDÚCHOVÁ, Mgr. Richard FECISKANIN, Ph.D., Department of Physical Geography and Geoinformatics, Faculty of Natural Sciences, Comenius University, Ilkovičova 6, Mlynská dolina, 842 15 Bratislava, e-mail: veronika.hajduchova@uniba.sk, richard.feciskanin@uniba.sk

and detects edges with adaptive threshold values respectively. Therefore, edge points with low height values are protected and adaptation of the algorithm is also improved. Another improvement of the Canny algorithm was brought by Wang et al. (2010). Because Canny algorithm may miss some obvious edges or involve weak edges, authors used an adaptive algorithm, which is capable of performing hysteresis threshold adaptively based on finding edge region and background region in the gradient magnitude histogram plot, that was employed for extracting building outlines from digital surface model (DSM).

Recent developments include edge detection directly on the point clouds using direct or indirect methods, depending how are 3D edges extracted. As mentioned in Dolapsaki and Georgopoulos (2021), most of the existing researches use for extraction statistics and geometric methods, where the selection of edges is based on density, massiveness, but also the object of interest. Various techniques use algorithms to detect outlines of objects by sensing changes in the of surrounding elements (Weber, 2012; Dolapsaki, 2021). An interesting approach to the edge detection is the segmentation technique, where the first step is to divide the point clouds into two groups: bare ground (digital terrain model (DTM)) and non-ground elements (DSM). Subsequently, the process starts from the highest height and gradually decreases where at each iteration looks for coplanar points to create planar segments and at each elevation level then groups the points based on the selected distance (Awrangjeb and Fraser, 2013). Such techniques are effective for extracting man-made features (roofs, houses) but struggle with vegetation (Abdullah et al., 2014). Bonneto et al. (2015) proposed an innovative method for the extraction of linear features using a semi-automatic geometrical approach based on DTM. The linear features identification comes from a geometrical analysis of the DTM. The method here described to perform linear feature detection on a DTM is based on the assumption that a geological lineament can be geometrically identified as a convex or concave edge of the surface of a DTM, in the presence of structural control of the geomorphological evolution of the analysed area. The method is called semi-automatic because the user is asked for two threshold values, in order to choose which edges are significant and therefore have to be extracted. Authors in Makka et al. (2024) propose a method that performs 3D edge detection by exploiting the direction differences of normal vectors in 3D point clouds. Authors have five stage workflow starting with generation of mesh and angle computation to use also graph theory. They used least square method to create best fitting 3D line and determine the 3D edge using RANSAC variation.

Despite this, most research focuses on LiDAR or raster-based edge detection. In this paper, we focus on triangulated irregular network (TIN) models. By computing the angle between normals of adjacent triangles, we perform edge detection on 3D DTMs at various levels of decimation. This facilitates the creation of edge hierarchies, showing which features remain prominent across different levels of detail.

1. Methodology

1.1 Areas / Territories

The first study area is the Bratislava – Rača district. We selected a section in the vicinity of the old Rača (Fig. 1), where vineyards, are organized into two groups - along the slope and in terraces. The terraced vineyards were particularly interesting due to their clearly defined edges but the territory is not so steep everywhere. Various roads and other cuts can also be seen there, as well as a field hockey playground, which contains high steep wall on the one side. The selected area has an elevation difference of 130.05 m with the lowest point at 151.92 m a. s. l. and the highest at 281.97 m a. s. l. (Fig. 2).

The second location is a quarry in the village of Rohožník, western Slovakia (Fig. 3). This location was chosen because of the sharp edges of the quarry, where we want to see how the sharp walls will display themselves during the decimation and how their edges will be visible at various levels. This territory has an elevation range of 179.67 m with the altitudes from 262.64 m a. s. l. to 422.31 m a. s. l. (Fig. 4).

Fig 1. The first location Bratislava – Rača, base map ZBGIS (https://zbgis.skgeodesy.sk/mapka/en/zakladna-mapa?pos=48.215221,17.152061,16)

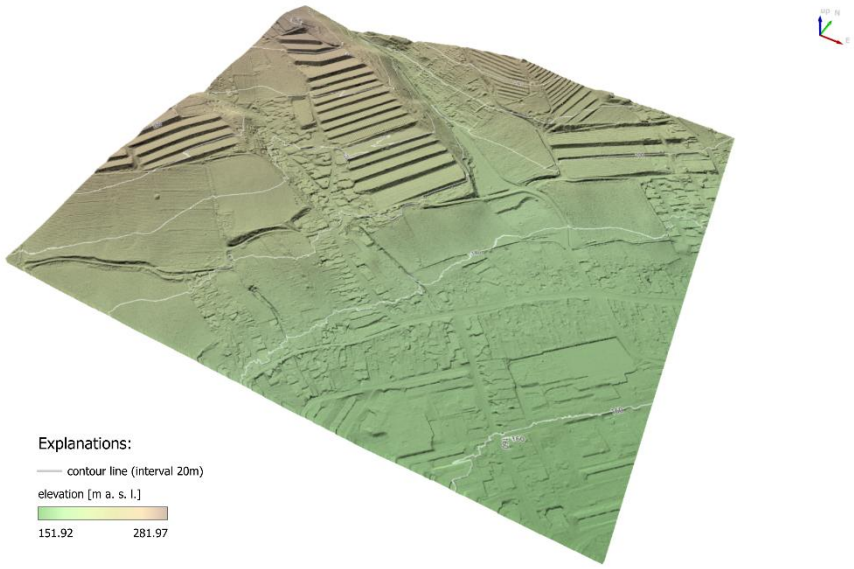

Fig 2. The first location – DTM with hillshade (1×1 km)

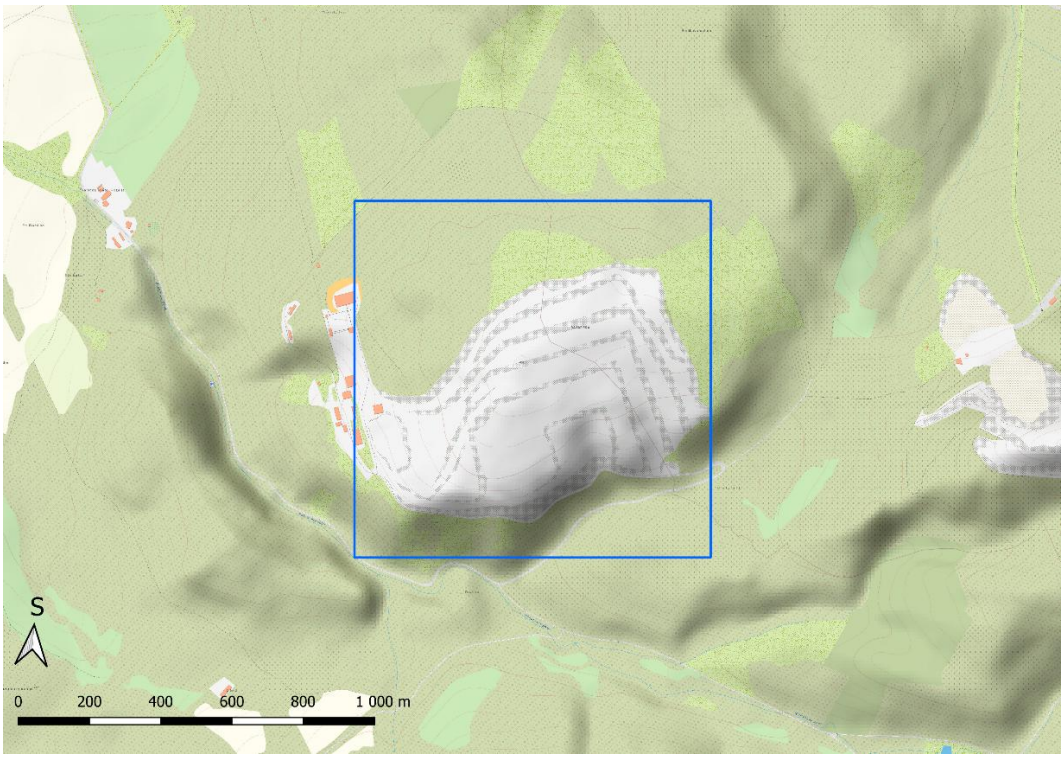

Fig 3. The second location Rohožník quarry, base map ZBGIS (https://zbgis.skgeodesy.sk/mapka/en/zakladna-mapa?pos=48.444189,17.210426,16)

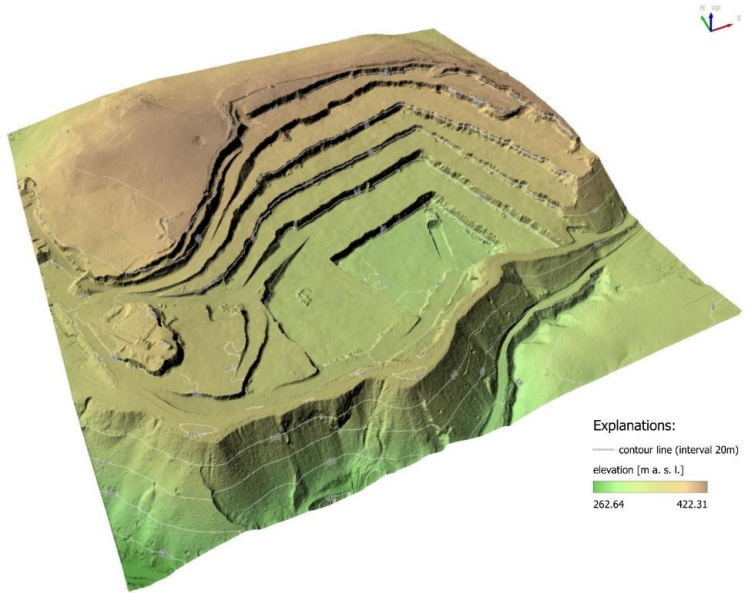

Fig 4. The second location – DTM with hillshade (1×1 km)

1.2 Workflow

Presented edge detection on TIN models is based on angle between normals of adjacent triangles, because angle between triangles can show inclination differences of adjacent triangles – planes. If the angle is smaller than given threshold value, the adjacent triangles have small plane inclination difference, so those triangles belong to the same plane, but when there is bigger angle, it can indicate discontinuity – edge on model.

We used airborne LiDAR data from Slovakia's Airborne Laser Scanning (ALS), available from the Geodesy, Cartography and Cadastre Authority of the Slovak Republic (Terrain | ZBGIS (skgeodesy.sk)). Point clouds from ALS needs to have quality criteria (Tab. 1) and they are classified up to ten categories (Tab. 2).

Tab. 1 Compulsory quality criteria of ALS

Compulsory quality criteria:	
scanning density	min. 5 points per m^2
overlap between swaths	min. 20% on 95% of their mutual coincidence
horizontal and vertical system	S-JTSK(JTSK03)+H_{Bpv}
	ETRS89-TM34+h_{ETRS89}
abs. vertical accuracy of point clouds at ellipsoidal heights ETRS89	$m_h \leq 0,15$ m
abs. positional accuracy of point clouds in ETRS89-TM34	$m_{XY} \leq 0,30$ m
abs. vertical accuracy of DTM 5.0 in h_{ETRS89}	$m_H \leq 0,20$ m
abs. vertical accuracy of DTM 5.0 in Bpv	$m_H \leq 0,25$ m

Source: https://www.geoportal.sk/en/zbgis/als/1st-cycle/

Tab. 2 Point cloud classification

Compulsory classification		**Optional classification**	
01	Unclassified	01	Unclassified
02	Ground	02	Ground
		03	Low vegetation
		04	Medium vegetation
		05	High vegetation
		06	Building
		07	Low point (Noise)
		09	Water
		17	Bridge deck
		18	Hight Noise

Source: https://www.geoportal.sk/en/zbgis/als/1st-cycle/

After obtaining these data we process them in *Cloud Compare* and *MeshLab* software, our new Python script and QGIS software.

In *Cloud Compare* software, we used only ground-classified points (class *02 Ground*) to generate a terrain models (DTMs) using *Delaunay triangulation* that is implemented in *Cloud Compare* to create the model. *Delaunay triangulation* belong to the group of shape-dependent triangulations, and it is one of the most used in the creation of a triangular network in relief modelling and thus takes precedence over other geometric triangulations. The reason why this method is so used is that it uses the maximization of the minimum angle and thus ensures that for a given set of points in general position, no other point lies in the circle described in any other triangle formed (Edelsbrunner et al., 2017; Feciskanin, 2009; Guo et al., 2021). The use of *Delaunay triangulation* has still limitations because it was developed for 2D applications. A better way to create a triangular mesh would be by using the theory of an *optimal triangle* which can be used for land surface modelling and generalization (Feciskanin and Minár 2020).

Next, we simplified the created TIN models using the *Quadric Edge Collapse Decimation (QECD)* method, which is implemented in the *MeshLab* software, and is a variation of the algorithm created by Garland and Heckbert (1997). It was based on the use of *Quadric Error Metrics* (*QEM*) simplification. The essence of this algorithm is to minimize the number of vertices and

faces of the original model, while the individual properties of the original model are still preserved. The individual simplified models created by this algorithm retain a very high fidelity compared to the original model, and all important elements are preserved even with a very large simplification (Fig. 5).

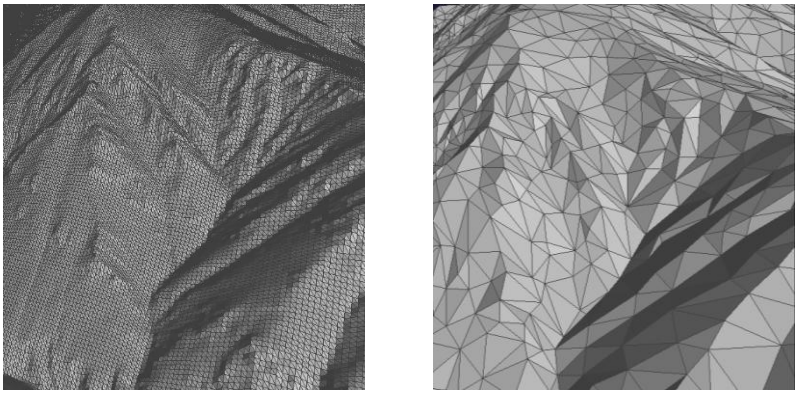

Fig 5. Example of a decimated territory using the QECD method (from 904,802 faces to 9,048 faces)

Our goal is to decimate the models and detect edges at various levels and thus determine their significance. After importing our model, we selected in the *Filters* tab *Remeshing, simplification and reconstruction* and there *Simplification: Quadric Edge Collapse Decimation*. We left the individual parameters that can be adjusted on default settings, and only changed the percentage (0-1) by which the model should be simplified compared to the initial size and preserve boundary of the mesh. After setting the percentage reduction, we got a simplified model of the selected territory, which we exported to an **.obj* file. Because each location is different and specific, the parameters for decimation must always be adapted in order to get the best possible results.

For the first location [*raca*] we created five simplified models. Because the original model was very detailed and hard for processing, we used reduction with targeted number of 1,000,000 faces – first level (Tab. 3). This was still very detailed for analysis, so we continued with percentage reduction. All levels were reduced to 0.2 from previous model. At the second location [*quarry (1)*] we used the same reduction scheme.

Because this algorithm works best with minimum levels and bigger reduction, we create new sequence in second location [*quarry (2)*] which was reduced with 0.1 three times, so we get similar number of vertices and faces as in the first sequence (Table 4).

Tab. 3 Number of vertices and faces in each level after decimation

levels		original	1	2	3	4	5
% reduction				20	20	20	20
raca	vertices	16,098,073	500,025	100,024	20,018	4,016	811
	faces	32,192,748	999,998	199,951	39,983	7,991	1,597
% reduction				20	20	20	20
quarry (1)	vertices	24,749,362	499,011	100,099	20,304	4,006	811
	faces	49,493,434	997,393	199,477	39,895	7,979	1,595

Tab. 4 Number of vertices and faces in each level after decimation

levels		original	1	2	3	4
% reduction				10	10	10
quarry (2)	vertices	24,749,362	499,011	49,896	5,003	511
	faces	49,493,434	997,393	99,738	9,973	993

Next step included creating script for loading, calculating, and exporting data from mesh. The script uses Trimesh library to load a 3D mesh from an input file and checks for duplicates, cleans mesh by removing duplicate vertices and faces and prints the number of vertices and triangles in the loaded mesh before and after cleaning. Then it identifies pairs of adjacent triangles (faces) and for each pair computes the angle between their normal using the dot product of the face normals to find cosine of the angle and converts it to degrees. If the angle exceeds the specifies threshold (threshold_angle) both faces are added to the set of selected faces and shared edges between those faces are identified and stored in list. The main result is shapefile with selected shared edges and if it is specified, the script exports the selected faces as **.obj* or shared edges as **.dxf* (Fig. 6). Script can handle bigger meshes, but it can take longer to run. Smaller meshed were run in less than a second or few seconds, but bigger meshes with 1,000,000 faces could run up to one minute (tested on computer with 32GB RAM). It also depends how big the threshold is and if we are exporting only **.shp* or other outputs. The script has arguments:

– input = input file containing 3D mesh,
– output_shp = file name for selected edges,
– threshold_angle = angle threshold for edge detection in degrees,
– crs = coordinate reference system for the shapefile, defaults to EPSG:5514 (optional),
– output_obj = file name for selected triangles (optional),
– output_dxf = file name for selected edges (optional).

Script is available on https://github.com/VeronikaH98/edge_detection.

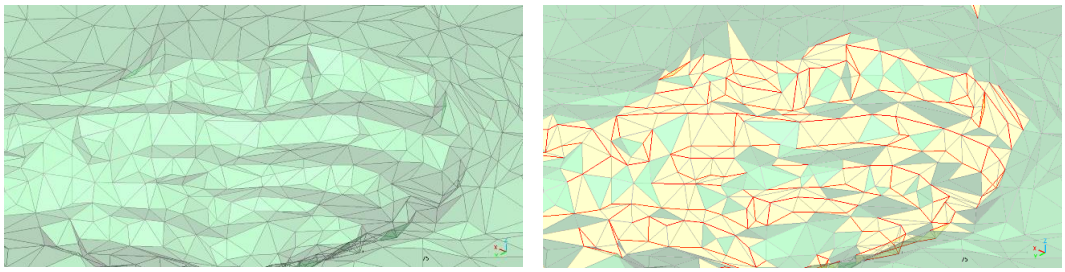

Fig 6. Illustration of loaded mesh and outputs (triangles.obj and edges.shp) from Python script

After the decimation process in *MeshLab*, we run our script across all territories on all levels of detail. Finding the optimal value for threshold angle can be challenging and often involves a trial-and-error approach. To support this decision, we generate normalized histograms and cumulative distribution functions (CDFs) for all levels in each territory to see distribution of angles, which can help to select suitable threshold value. In some cases, the threshold depends on the specific types of features that user intends to detect.

Figures 7 to 9 illustrate the normalized distribution of edge angles in three territories at all levels of decimation. Each graph combines a histogram (showing relative frequency) with a cumulative distribution function to provide clearer understanding of angle distribution. The black dashed line marks selected threshold angle used to distinguish significant geometric features. The CDF curves indicate that the vast majority of edges have relatively low angles, with rapid accumulation below the selected thresholds. In raca (Fig. 7), over 90% of edges across all decimation levels have angles below 30°, while in *quarry (1)* and *quarry (2)* (Fig. 8 and Fig. 9), approximately 94–95% of edges fall below 35°. This confirms that the thresholds isolate a relatively small set of steeper angles, which are likely to correspond to more prominent geometric features.

These observations are further supported by the summary of edge counts presented in Table 5. In raca, only 3.85% of edges at level 1 exceed the 30° threshold. Similarly, in both quarry datasets, around 6.4% of edges at level 1 exceed the 35° threshold. This trend continues at higher decimation levels, where the proportion of edges above the threshold remains consistently low. The selected thresholds thus effectively separate the sharpest edges – preserving detail while excluding smoother transitions that are less relevant for feature identification.

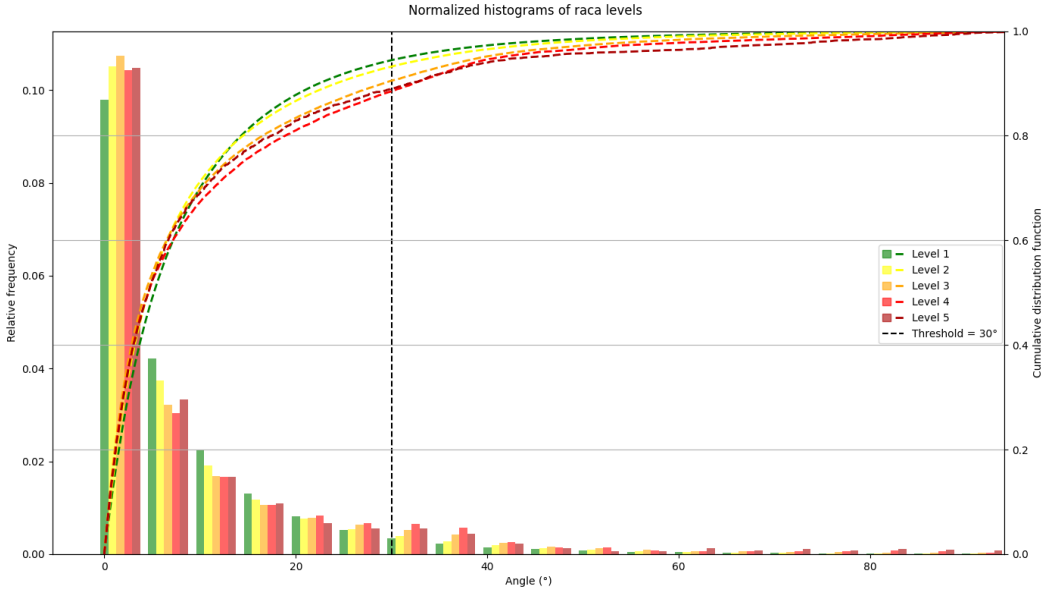

Fig. 7 Normalized histograms and cumulative distribution functions of angles for edges across five levels of decimation in the first location – *raca*. The black dashed line marks the 30° threshold

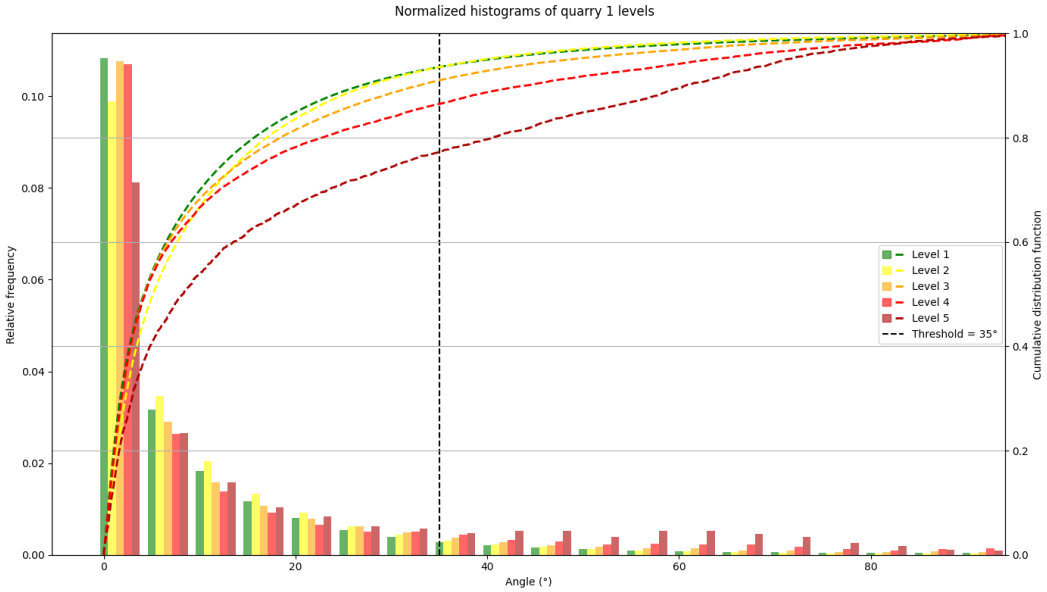

Fig. 8 Normalized histograms and cumulative distribution functions of angles for edges across five levels of decimation in the second location – *quarry (1)*. The black dashed line marks the 35° threshold

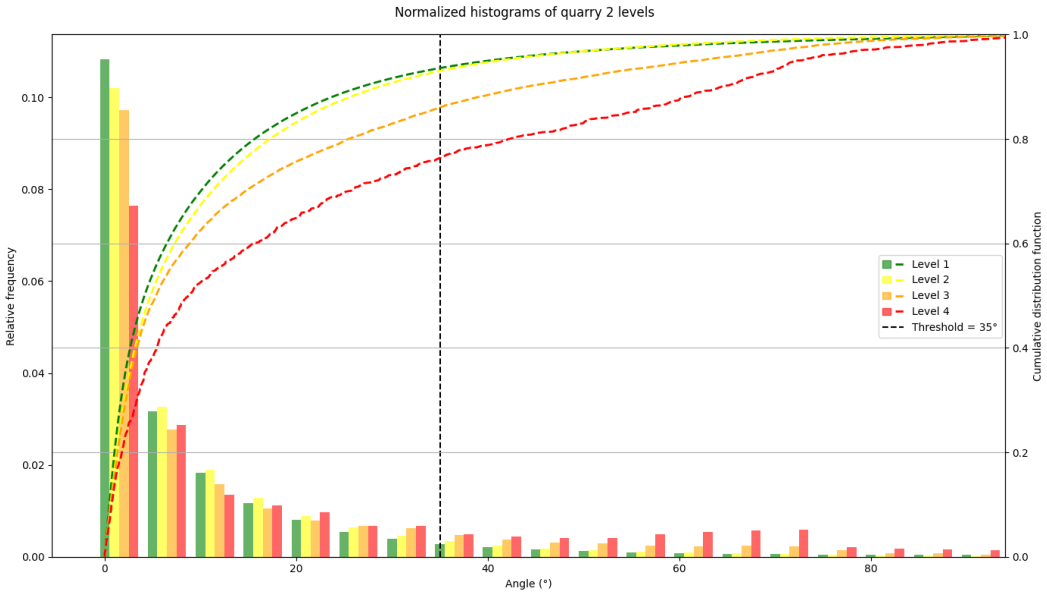

Fig. 9 Normalized histograms and cumulative distribution functions of angles for edges across five levels of decimation in the second location, second sequence – *quarry (2)*. The black dashed line marks the 35° threshold

Overall, both the graphical and numerical results validate the chosen threshold angles. They ensure a balance between retaining important topographic or structural details and eliminating noise or insignificant geometry. Selecting a lower threshold would include too many minor edges, while a higher one would risk discarding meaningful sharp features. The adopted values of 30° for raca and 35° for the two quarry datasets are therefore considered appropriate and effective for the intended analysis.

Tab. 5 Total number of edges and selected edges (depends on threshold) on each level across all locations

	levels	1	2	3	4	5
raca	total edges	1,048,575	298,480	59,509	17,683	3,501
	edges > 30°	40,349	14,417	4,106	1,458	288
quarry (1)	total edges	1,048,575	293,551	55,962	14,847	3,238
	edges > 35°	67,291	18,828	5,022	2,005	735
quarry (2)	total edges	1,048,575	148,325	14,658	1,440	
	edges > 35°	67,291	10,336	2,046	340	

2. Results and discussion

This section presents the outcomes of the decimation analysis for all three locations. In the figures, the levels are distinguished by colors: level 2 in yellow, level 3 in orange, level 4 in red and level 5 in burgundy.

The first location was selected mainly because of the interesting, terraced shape of the vineyards where we wanted to test how the given edges are detected. The threshold angle was set to 30° in this territory. Looking at the entire territory (Fig. 10), we can say that the individual edges appear where we expected them, especially on the individual terraces of the vineyards, where they are very prominent throughout the entire period of decimation until the last level, where some are

not merged together. As we can see, almost all the terraces were so sharp and significant that even after several times of decimation, they were very visible and did not merge with the surroundings (Fig. 11). With lesser threshold angle, we would be probably able to capture all of terraces. Equally prominent are the edges next to the roads, the stream in the valley, which is in a few meters deep cut, but also the field hockey stadium in the southeast of the territory, which is separated from the family houses and gardens by a 3 to 4 meter high vertical wall over it. It was this wall that interested us even during decimation and comparison with individual levels.

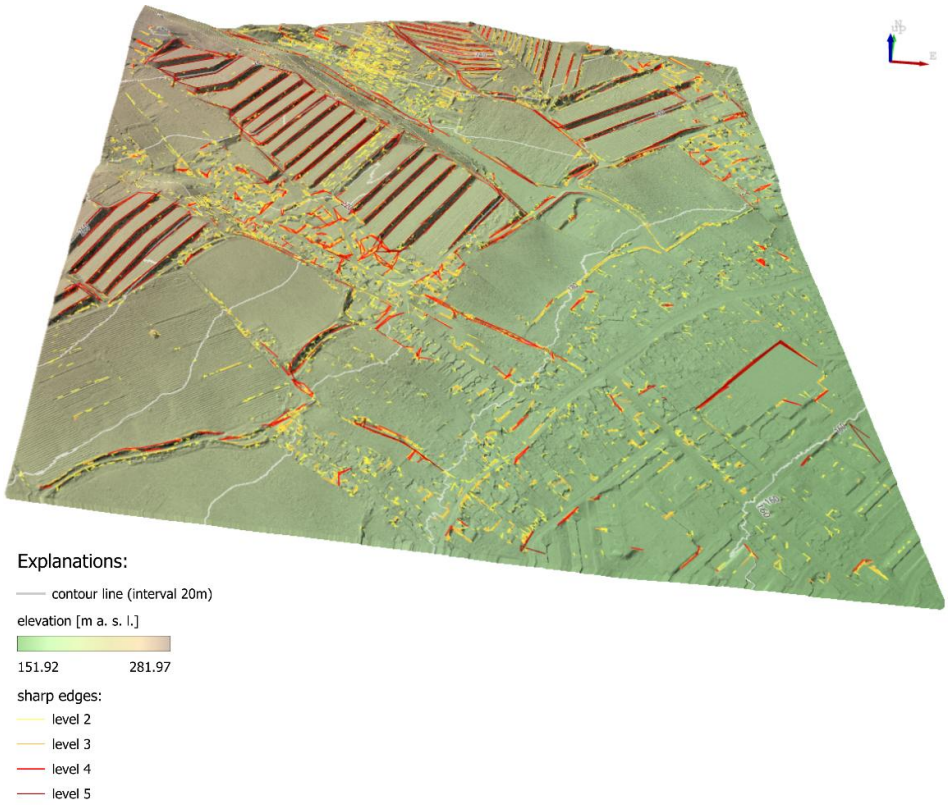

Fig. 10 Detected edges (30°) on each level in the first location – output of the edge detection algorithm applied to the study area, highlighting how edges on different levels of detail are captured

The second location captures how decimation and edge detection behaves on a relatively sharp terrain in a quarry, where a threshold value of 35° was set. As we expected, the decimation algorithm was able to preserve the main features of the quarry even to the lowest level and thus increased the angle of individual quarry walls, which enabled very accurate detection of the upper and lower edge of the wall (Fig. 12). In the zoomed-in images (Fig. 13), detected edges are not clearly visible, even though they are detected. The visualization in the *Cloud Compare* environment did not allow us to insert a line layer on the model, so it may appear as if individual edges are missing, but they are actually located beneath the model. This is only an aesthetic issue; the edge detection was performed correctly.

In the second sequence of the second location, we wanted to compare how the decimation algorithm can deal with prominent elements even with a larger set value, i.e., a more simplified surface already after the first level. Similar to the first sequence, we chose a threshold of 35° here. We can see that the decimation took place in such a way that significant elements remained in each level again (Fig. 14), but if we compare the first and second sequence, we can see that the corners of the

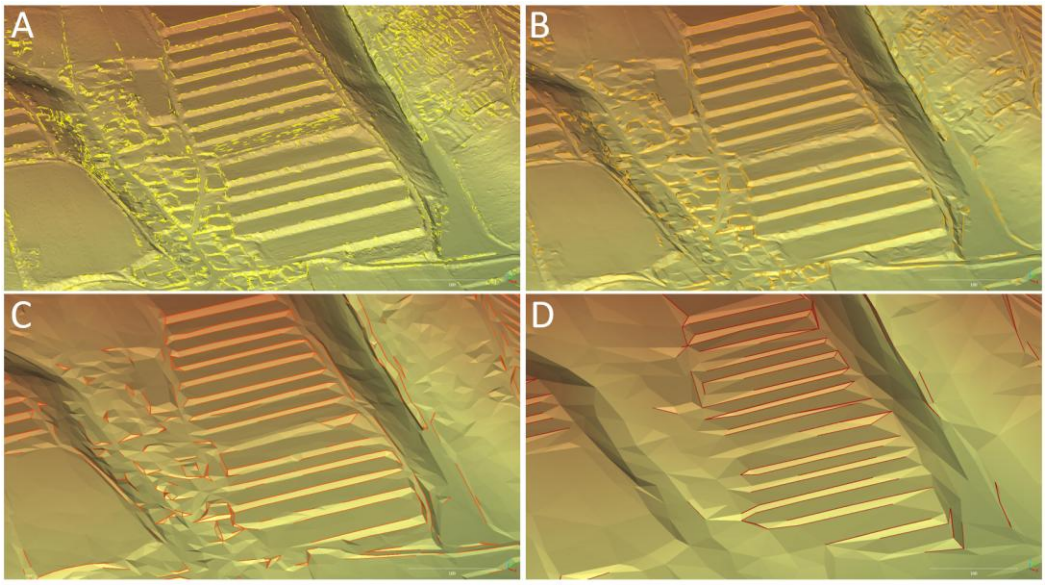

Fig. 11 Detail of the detected edges in the first location. Closer look at vineyard terraces: A – level 2, B – level 3, C - level 4, D – level 5

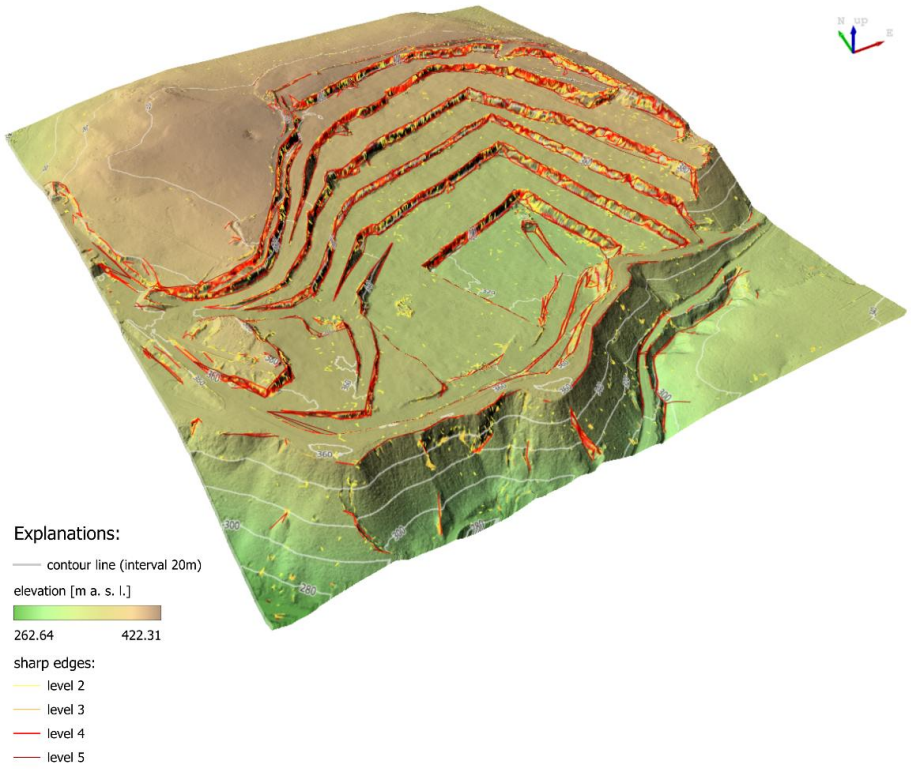

Fig. 12 Detected edges (35°) on each level in the second location – output of the edge detection algorithm applied to the study area, highlighting how edges on different levels of detail are captured

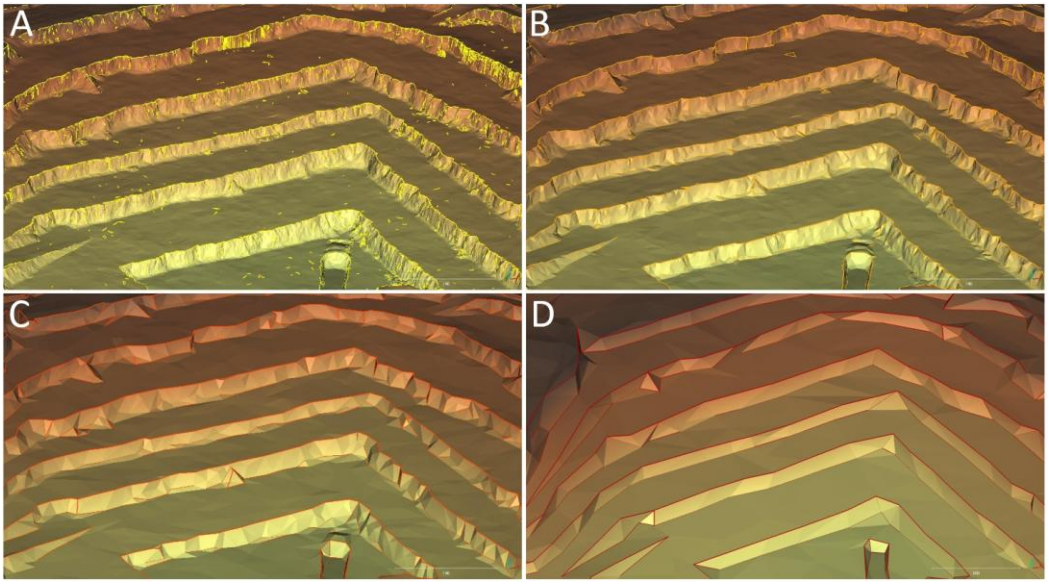

Fig. 13 Detail of the selected edges in the second location. Closer look at quarry walls: A – level 2, B – level 3, C - level 4, D – level 5

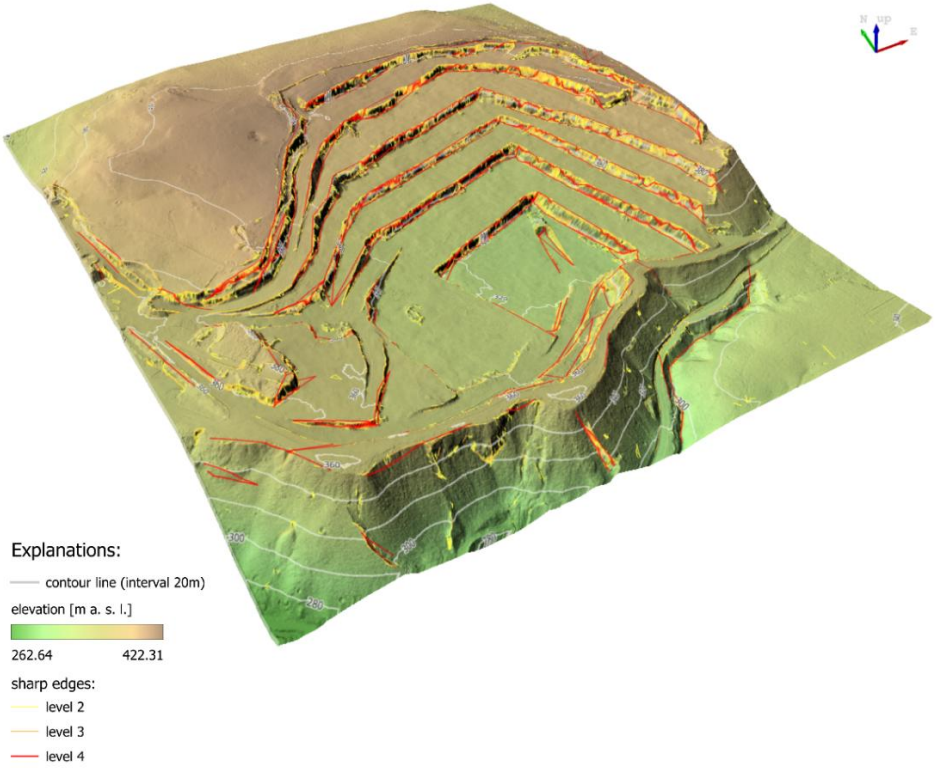

Fig. 14 Detected edges (35°) on each level in the second location – second sequence

individual levels in the quarry are sharper already in the first stages of decimation, and thus the individual contours of the walls are better identifiable (Fig. 15). At the fourth level of the second sequence, the number of edges and faces reached below the fifth level of the first sequence, so we can say that the initially larger decimation set has better results, which was also pointed out earlier in the text.

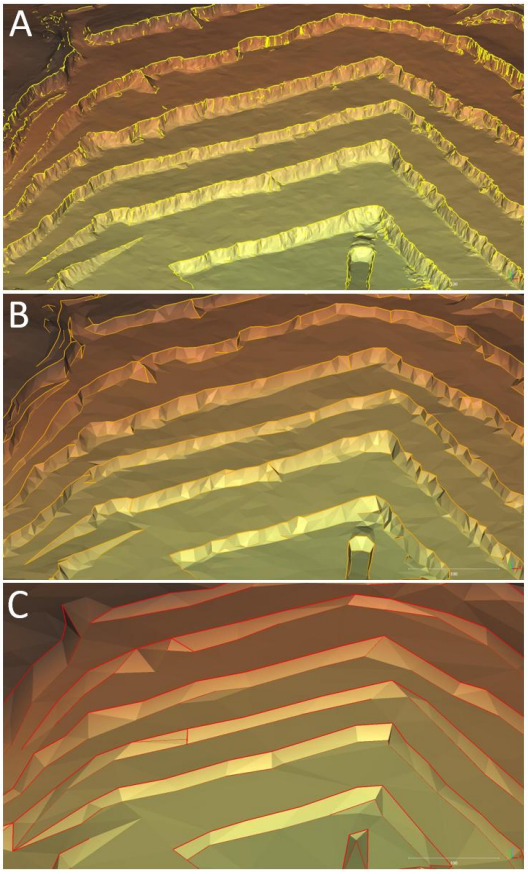

Fig. 15 Detail of the selected edges in the second location – second sequence. Closer look at quarry walls: A – level 2, B – level 3, C – level 4

The presented procedure for processing the LiDAR point clouds into TIN models and their subsequent decimation, on which the detection of significant edges is possible using the calculation of the angle between two neighbouring triangle normals, appears to be of high quality for identifying and comparing sharp edges in 3D models. This approach can be applied in various fields, including urban planning and terrain modelling. The presented script developed for this workflow is versatile, offering clean and optimized outputs in multiple formats, and it is capable of removing duplicates, calculating face angles, and exporting results. It can help extract and analyse features like steep slopes, cliffs, or other significant terrain changes by identifying faces with large angles and can be useful in CAD and GIS applications. Also, in geospatial analysis by integrating processed 3D data into GIS systems for further analysis and visualization or in architectural and structural engineering by analysing and extracting critical geometric features from structural models.

However, despite these advantages, the procedure has some limitations. One key challenge is the dependence on threshold selection, which requires a careful, often trial-and-error approach that might not generalize well across different datasets. The accuracy of edge detection can also be influenced by data quality, especially in regions with noisy or incomplete point clouds. Another limitation is the computational load when processing large or very dense meshes, which can result in longer processing times or memory issues. Furthermore, while the method was tested here on DTMs, applying it to DSMs or complex urban scenes may require additional preprocessing and filtering to ensure reliable results.

Future work could focus on automating the selection of the threshold angle based on dataset-specific metrics and exploring the method's performance on DSMs and building models. Despite these challenges, the presented approach offers a valuable tool for analysing and extracting hierarchical edge structures in 3D data.

Conclusion

In this paper, we presented a script for edge detection on TIN models, which calculates the angle between two adjacent triangles and based on a specified threshold, save and exports the identified edges. This procedure was tested on several territories at different levels of detail, providing insight into hierarchy of individual edges throughout the entire decimation process. Although classical algorithms such as Sobel or Canny operator achieve high-quality results on raster images, this method gives priority to 3D meshes and offers complementary information by identifying 3D structural features. While the script performs well, selecting an appropriate threshold angle remains a challenge, and further testing is needed to assess its effectiveness on DSMs or complex urban models. Overall, this approach shows promise for extending 3D edge detection beyond traditional 2D applications and supports more detailed and hierarchical analysis of terrain and built environments.

References

ABDULLAH, S., AWRANGJEB, M., LU, G. (2014). Automatic segmentation of LiDAR point cloud data at different height levels for 3D building extraction. In: *IEEE International Conference on Multimedia and Expo Worksh*ops, ICMEW, Chengdu, China, 1-6. DOI: 10.1109/ICMEW.2014.6890541

AWRANGJEB, M., FRASER, C. (2013). Rule-based segmentation of LIDAR point cloud for automatic extraction of building roof planes. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, II-3/W3, 1-6. DOI: 10.5194/isprsannals-II-3-W3-1-2013

BONETTO, S., FACELLO, A., FERRERO, A.M., UMILI, G. (2015). A tool for semi-automatic linear feature detection based on DTM. *Computers & Geosciences*, 75, 1-12. DOI: https://doi.org/10.1016/j.cageo.2014.10.005

DOLAPSAKI, M.M.; GEORGOPOULOS, A. (2021). Edge Detection in 3D Point Clouds Using Digital Images. *ISPRS International Journal of Geo-Information*, 10(4), 229. DOI: https://doi.org/10.3390/ijgi10040229

DAS, S. (2016). Comparison of Various Edge Detection Technique. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 9, 2, 143-158. DOI: http://dx.doi.org/10.14257/ijsip.2016.9.2.13

EDELSBRUNNER, H., GLAZYRIN, A., MUSIN, O.R., NIKITENKO, A. (2017). The Voronoi functional is maximized by the Delaunay triangulation in the plane. *Combinatorica*, 37 (5), 887-910. DOI: https://doi.org/10.1007/s00493-016-3308-y

FECISKANIN, R. (2009). *Optimalizácia nepravidelných trojuholníkových sietí pre modelovanie georeliéfu*. Dizertačná práca. Prírodovedecká fakulta, Masarykova Univerzita v Brne. [online] [cit. 2024-10-01]. Available at: <https://is.muni.cz/th/knx26/FeciskaninTextDP.pdf>

FECISKANIN, R., MINÁR, J. (2020). An optimization of triangular network and its use in DEM generalization for the land surface segmentation. In: *Geomorphometry 2020 Conference Proceedings, Perugia, Italy*. DOI: 10.30437/GEOMORPHOMETRY2020_2

GARLAND, M., HECKBERT, P. (1997). Surface simplification using quadric error metrics. In: *Proceedings of the 24th annual conference on Computer graphics and interactive techniques (SIGGRAPH '97). ACM Press/Addison-Wesley Publishing Co., USA*, 209-216. DOI: https://doi.org/10.1145/258734.258849

GUO, Y., HUANG, X., MA, Z., HYE, Y.Q., ZHAO, R., SUN, K. (2021). An Improved Advancing-front-Delaunay Method for Triangular Mesh Generation. In: *Advances in Computer Graphics, 38th Computer Graphics International Conference, CGI 2021*. DOI: 10.1007/978-3-030-89029-2_37

IGBINOSA, I. (2013). Comparison of Edge Detection Technique in Image Processing Techniques. *International Journal of Information Technology and Electrical Engineering*, 2, 1, 25-29. [online] [cit. 2024-10-03]. Available at: <https://www.researchgate.net/publication/272023819_Comparison_of_Edge_Detection_Technique_in_Image_Processing_Techniques>

JING J., LIU, S., WANG, G., ZHANG, W., SUN, CH. (2022). Recent advances on image edge detection: A comprehensive review. *Neurocomputing*, 503, 259-271. DOI: https://doi.org/10.1016/j.neucom.2022.06.083

MINÁR, J., EVANS, I.S. (2008). Elementary forms for land surface segmentation: The theoretical basis of terrain analysis and geomorphological mapping. *Geomorphology*, 95, 3-4, 236-259.

MAKKA, A., PATERAKI, M., BETSAS, T., GEORGOPOULOS, A. (2024). 3D Edge Detection based on Normal Vectors. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLVIII-2/W4-2024, 295-300. DOI: 10.5194/isprs-archives-XLVIII-2-W4-2024-295-2024

PACINA, J. (2008). *Metody pro automatické vymezování elementárních forem georeliéfu jako součást Geomorfologického informačního systému*. Disertační práce. Plzeň (Západočeská univerzita, Fakulta aplikovaných věd).

PU, M., HUANG, Y., GUAN, Q., LING, H. (2021). RINDNet: Edge Detection for Discontinuity in Reflectance, Illumination, Normal and Depth. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. [online] [cit. 2024-10-02]. Available at: <https://openaccess.thecvf.com/content/ICCV2021/papers/Pu_RINDNet_Edge_Detection_for_Discontinuity_in_Reflectance_Illumination_Normal_and_ICCV_2021_paper.pdf>

WANG, Z., LI, H.-Y., WU, L.-X. (2010). QEM-based simplification of building footprints from Airborne LiDAR data. In: *IEEE International Geoscience and Remote Sensing Symposium, Honolulu, HI, USA*, 1186-1189. DOI: 10.1109/IGARSS.2010.5654122

WEBER, CH., HAHMANN, S., HAGEN, H., BONNEAU, G.-P. (2012). Sharp feature preserving MLS surface reconstruction based on local feature line approximations. *Graphical Models*, 74, 6, 335-345 DOI: 0.1016/j.gmod.2012.04.012

XU, J., WAN, Y., ZHANG, X. (2006). A method of edge detection based on improved canny algorithm for the lidar depth image. *Geoinformatics: Remotely Sensed Data and Information*, 6419, 64190O–64190O–9). DOI: http://doi.org/10.1117/12.712923

R e s u m é

Detekcia hrán pomocou zjednodušenia trojuholníkovej siete

Tento článok predstavuje novú metódu a skript na detekciu hrán na nepravidelnej trojuholníkovej sieti (TIN), ktorá sa odlišuje od tradičných a existujúcich prístupov zameraných prevažne na 2D rastrové obrazy. Metóda je založená na výpočte uhlov medzi normálami susediacich trojuholníkov, pričom väčší uhol signalizuje diskontinuitu alebo hranu.

Hlavným cieľom je aplikácia tejto metódy na digitálne modely reliéfu (DMR) na rôznych úrovniach zjednodušenia, čo umožňuje vytvárať hierarchie hrán a identifikovať najvýraznejšie prvky. Metodika zahŕňa využitie údajov z leteckého laserového skenovania (LLS) zo Slovenska. Proces spracovania dát zahŕňa softvér CloudCompare na generovanie DMR pomocou Delaunayovej triangulácie z klasifikovaných bodov terénu. Následne sú TIN modely zjednodušené v softvéri MeshLab pomocou metódy Quadric Edge Collapse Decimation (QECD), ktorá aj pri veľkom zjednodušení modelu dokáže verne zachovať jeho pôvodné vlastnosti. Vyvinutý Python skript pracuje v niekoľkých krokoch: načíta 3D sieť, identifikuje a odstráni duplikáty, vypočíta uhly medzi normálami susediacich trojuholníkov a identifikuje spoločné hrany, ak uhol presiahne špecifikovanú prahovú hodnotu. Výstupy je možné exportovať ako **.shp* alebo **.dxf* pre vybrané hrany a voliteľne aj ako **.obj* pre vybrané plochy trojuholníkov. Skript je voľne dostupný na GitHub-e a spúšťa sa z príkazového riadku.

Výber optimálnej prahovej hodnoty uhla je náročný a často zahŕňa metódu pokusu a omylu, pričom na podporu tohto rozhodnutia boli použité normalizované histogramy a kumulatívne distribučné funkcie (CDF). Pre lokalitu Rača bola prahová hodnota nastavená na 30°, zatiaľ čo pre kameňolom bola použitá hodnota 35°. Tieto prahové hodnoty efektívne izolovali strmšie uhly zodpovedajúce významným hranám.

Výsledky z testovaných lokalít (Rača, kameňolom Rohožník) potvrdili efektívnu detekciu terénnych prvkov ako terasy viníc a steny lomu, ktoré zostali viditeľné aj po niekoľko násobnom zjednodušení. Väčšie po-

čiatočné zjednodušenie viedlo v niektorých prípadoch k ostrejším hranám. Prezentovaná metóda je využiteľná v GIS a CAD aplikáciách napríklad pre analýzu terénu, no medzi obmedzenia patrí závislosť na výbere prahovej hodnoty, kvalita vstupných dát a výpočtová náročnosť pre rozsiahle siete. Budúci výskum sa zameria na automatizáciu výberu prahových hodnôt a aplikáciu na zložitejšie modely. Celkovo metóda posúva 3D detekciu hrán a umožňuje detailnejšiu hierarchickú analýzu terénu.

Prijaté do redakcie: 13. apríl 2025

Zaradené do tlače: jún 2025