3D Edge Detection based on Normal Vectors

A. Makka[1*], M. Pateraki[1, 2], T. Betsas[1], A. Georgopoulos[1]

[1]Lab. of Photogrammetry, School of Rural, Surveying & Geoinformatics Engineering, National Technical University of Athens
[2]Institute of Communication and Computer Systems, National Technical University of Athens
rs18050@mail.ntua.gr, mpateraki@mail.ntua.gr, betsasth@mail.ntua.gr, drag@central.ntua.gr

Commission II

KEY WORDS: Edge detection, point cloud, 3D mesh, graph theory, least squares, RANSAC

ABSTRACT: Edge detection is supported by extensive research and is part of different photogrammetric and computer vision tasks across numerous application areas. While 2D edge detection may achieve high accuracy results from several automated methods, the automation of edge detection in 3D space remains a challenge. Existing methods are often computationally demanding and heavily parameterized, leading to a lack of adaptability. In real-world applications 3D edges, representing the object boundaries and break lines, are crucial, particularly in fields such as computer vision, robotics and architecture. In this context, we present a method that automates 3D edge detection in 3D point clouds exploiting the normal vectors' direction differences to detect finite edges, which are further pruned and grouped to edge segments and fitted to indicate the presence of a 3D edge.

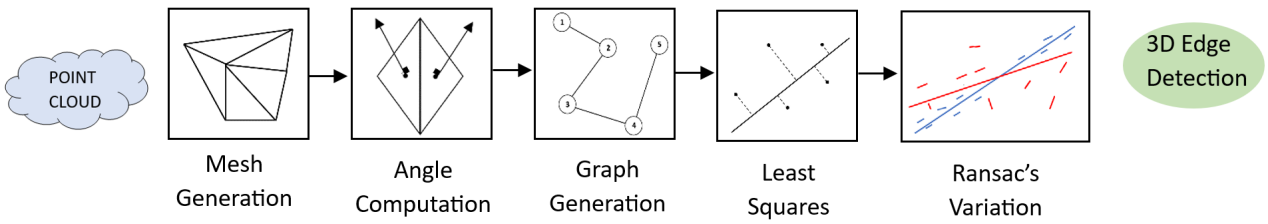


Figure 1 . The five stage workflow of the proposed method for 3D edge detection.

1. INTRODUCTION

Edge detection in 2D images is a well-investigated topic in the photogrammetric and computer vision communities. Line features from images are extracted with a variety of methods, either exploiting classical pipelines or deep neural networks that have recently emerged. (e.g. Abdellali et al., 2021, Zhang et al., 2021). 3D edge extraction has attracted increasing interest over the last few years. More specifically, the analysis of 3D data and the extraction of line segments can be used in multiple tasks, such as 3D building documentation, construction, restoration, Structure from Motion (SfM) which has many applications in augmented reality, visual localization and mapping, etc. All the aforementioned applications require high-accuracy 3D line vectors, which are usually extracted by delineating break lines in 3D mesh models and point clouds. Nowadays, this process is performed manually and is time-consuming. However, the research community still finds it challenging to provide a computationally efficient and geometrically accurate approach for automated 3D edge detection and vectorization in point clouds and 3D mesh models.

This paper proposes a method that exploits the normal vectors' direction differences to identify edges in unorganised 3D point clouds. It follows by further grouping and merging the detected finite edges into 3D edges. The results of this method have also been assessed qualitatively to test the efficiency and accuracy of the method.

2. RELATED WORK

Numerous 2D or 3D edge detection algorithms have been proposed in the recent past to speed up the process of delineating 3D line features in 3D models or 3D dense point clouds or modelling surface discontinuities by matching edge features in 2D images (Pateraki and Baltsavias, 2004). This is also highly relevant for generating 3D vector drawings, for which the state-of-the-art process is time consuming, laborious and requires specialists from several scientific fields. The detection may be performed in two ways. Firstly, detect the edges on the 2D images and project them into 3D models or point clouds. 2D edge detection is a fundamental computer vision problem and various traditional edge detection operators proposed in the literature exploit the gradient magnitude to detect pixels that may lie on an edge, while further search and aggregate those pixels sharing a similar gradient angle to a 2D edge.

The most notable handcrafted 2D line detectors are LSD (Grompone et al, 2008), EDLines (Akinlar and Topal, 2011) and several extensions such as ELSED (Suarez et al, 2022). These methods are in general fast but lack robustness in challenging imaging conditions producing noisy lines.

Recently, a few methods proposed deep learning architectures to detect 2D edges (Poma et al., 2020; Su et al., 2021, Pautrat et al., 2023). Several efforts have been reported, which are based on image segmentation (Dhankhar & Sahu, 2013, Lu et al., 2019, Xie et al., 2020), which aim to cluster the pixels or points into groups with similar geometric or spectral characteristics without considering semantic meaning. Furthermore, 3D edge detection

may be realized by detecting the edges directly in the 3D environment, mainly the point clouds. Several techniques have been proposed, e.g., model fitting, analytical geometry, semantically enriched images, etc. Certain researchers (Rodríguez Miranda et al. 2008, Canciani et al. 2013,) have tried to produce vectors, i.e. linear features, directly from the point clouds either manually or semi-automatically. However, this procedure is time consuming and extremely strenuous and demands loads of computer power. Consequently, the solution lies in the automation of vector detection directly in the point cloud (Briese & Pfeifer, 2008). Nguatem et al. (2014) used predefined templates of windows and doors to detect their 3D boundaries exploiting plane intersection. Mitropoulou and Georgopoulos (2019) first segmented 3D point clouds into planes and then detected the 3D edge points by applying plane intersection. Liakopoulou (2022) performed plane intersection based on planes defined by edges on images and their exterior orientation parameters. Bazazian et al. (2015) first find the nearest neighbours of 3D points and then for each group of 3D points, the covariance matrix is calculated. Finally, the eigenvalues and eigenvectors of each matrix are examined to detect the sharp 3D edges by deciding whether the point belongs to a plane or to an edge. Lu et al. (2019) also exploit the eigenvalues and eigenvectors of the calculated covariance matrix of the points' neighbourhood. The 3D point cloud is segmented into planes, using the region growing and merging method. Afterwards, the 3D points of each fitted plane are projected onto it, to create images. Finally, a 2D contour detection algorithm is applied and the detected 2D contours are projected back into 3D space. Additionally, Dolapsaki and Georgopoulos (2021), proposed a 3D edge detection method, which exploits the relationships of analytical geometry and the properties of planes in combination with digital images. These planes are defined by edges detected on images and the exterior orientation elements of the images. Finally, the detected 3D edge points inevitably lie on these planes.

Detecting edges in 3D space can also be performed by using 2.5D data e.g., range images, and depth images. Bao et al. (2015) proposed an approach which first creates range images from a given point cloud, then applies the Canny operator (Canny, 1983) on them and finally projects the 2D edges into 3D space. Recently, Betsas & Georgopoulos (2022) have proposed a method which uses semantically enriched images to detect straight edges after an SfM/MVS process of these images.

3. METHODOLOGY

The following subsections provide brief overviews of key concepts that are of central importance for the development of the proposed method in Section 4

3.1 3D surface representation

A triangulated mesh is generated from the unorganised point cloud data, which serves as a 3D surface representation. To achieve this, the Ball Pivoting Algorithm (BPA) (Bernardini et al. 1999) is exploited as it can handle datasets consisting of millions of points with short execution times and has shown good performance on noisy point clouds acquired from laser scanning. The BPA, as the name suggests, uses a ball of ρ-radius that interpolates the point cloud to create the triangular mesh. However, during this process, some gaps may be formed on the surface. Therefore, the BPA allows the use of different user-specified radii to avoid these gaps on the surface. The algorithm starts with the smallest radius and the process is repeated sequentially with the larger radii to fill the gaps. The BPA is provided as a built-in function in the OPEN3D library (https://www.open3d.org). To generate the geometric mesh representation, we calculate the mean distance from 30 neighbouring points within the sphere of a predefined radius using OPEN3D's built-in function. Then, this distance is multiplied by one, two, four and eight to determine the radii which are used in the BPA algorithm.

3.2 Edge Detection in 3D Point Clouds

The normal vector is computed for every triangle of the mesh, and the angle between the normal vectors of two adjacent triangles is subsequently calculated. This value, representing the inclination differences of adjacent planes, is then assigned to the common edge of the two triangles. More specifically, a small angle between adjacent triangles indicates small plane inclination differences, which means that the triangles possibly belong to the same plane, while larger angles may indicate the presence of a 3D discontinuity, i.e., an edge.

3.3 Graph Generation

In brief, a graph is a data structure that consists of nodes and edges. Understanding the relations among the elements of unordered data i.e., which do not lie on a regular grid, like 3D point clouds, is complicated. Transforming the unordered data to a graph provides a way to understand the relations among them, by using primarily the edges of it. The triangulated network which was created in the previous step consists of points and edges. Thus, it could be treated as a graph. Each edge of the triangulated network belongs to at most two -adjacent- triangles. To enhance the information carried by the edges, the inclination differences between the normals of their adjacent triangles is inherited to them. Hereafter, the valuable information of the inclination angles could be used throughout the workflow, diversely as a property of each edge.

Using these data an undirected cartesian graph (G) was generated. The constructed graph (G) contains all the edges of the triangulated network. However, the goal is to extract only the 3D edges of the object, which is a subset of the graph's edges. To achieve this, the calculated angles between the adjacent triangles, which were previously inherited to each edge of the graph, are exploited. Firstly, a lower θ1 and an upper θ2, threshold values are defined. The edges that do not satisfy the above threshold are eliminated from the graph (G) resulting in a first estimation of objects' 3D edges in the form of 3D finite edges. During this effort the threshold values θ1, θ2 are defined as π/6 and 6π/7, respectively i.e., the goal is to identify the edges that lay in between the two threshold values using the following inequality.

$$\frac{\pi}{6} < angle < \frac{6\pi}{7} \quad (1)$$

After eliminating those finite edges the initial cartesian graph (G) is converted to a line graph L(G), where each vertex of L(G) refers to an edge of G (Figure 2). By computing the angle between each three consecutive vertices of the L(G) we can find which finite edges belong to the same final edge and group them together. More specifically, if the angle value is greater than 5π/6, the 3D finite edges are parts of the same final edge and are grouped together.

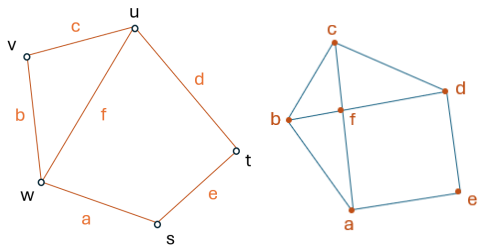


Figure 2. Conversion of graph (G) to line graph L(G).

3.4 Least Squares

The graph generation step of the previous section provided us with groups of *finite edges*. In order to combine them into a single edge segment, we used the Least Squares method to estimate the best-fitting 3D line. However, as the edges are primarily line segments it is necessary to detect the endpoints of every best-fitting parametric line to transform it into a segment. To do so, we project the initial *finite edges* onto the parametric line and keep as endpoints of the line segment the ones that are furthest apart.

3.5 RANSAC variation

From the line segments that are estimated from the least squares method, the ones that represent different parts of the same edge should be combined, excluding line segments attributed to the presence of random noise and surface abnormalities (roughness). Therefore, we exploit the RANdom SAmple Consensus algorithm (RANSAC), which manages to estimate a model from data in the presence of outliers. In classical RANSAC, a single data object (e.g point, line segment), which is significantly distant from the model is automatically labelled as an outlier, without checking if the segments between the model and this segment are inliers or not. However, we aim at expanding the edge segments as much as possible in order to detect a prominent 3D edge and eliminate possible gaps. We are proposing a variation of the RANSAC algorithm, where we use as a model one single segment and try to expand each of its endpoints using inlier segments. This process is performed for a single endpoint each time. For reasons of computational complexity, we outright marked segments extremely distant or misaligned as outliers, as we want the perpendicular distance to the model and the angle between the initial model and the new ones to remain mostly constant. The remaining segments were characterised as "possibly extending endpoints of the model". Therefore, the algorithm checks for each of these segments in ascending order of distance to the model, to mark them as inliers, while updating the edge endpoints. The model is updated with every new inlier and therefore its length is constantly increasing. In this way, a segment, which has a large distance from the initial model can be marked as an inlier for the updated one. Therefore, we are not missing inliers due to distance limitations from the initial model, which leads to the initial model being expanded to its maximum extent.

4. IMPLEMENTATION

The proposed method exploits the normal vectors' direction differences to detect edges in 3D point clouds. A five-stage workflow was adopted to achieve this scope.

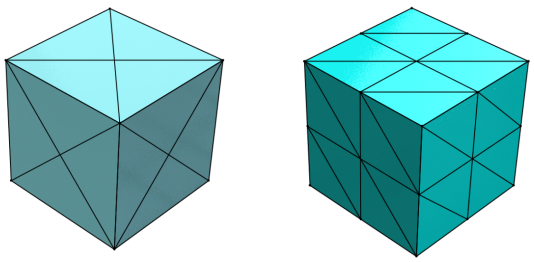
- **Triangulated Mesh** generation.
- **Angle computation** between adjacent normals.
- **Graph theory** implementation to organise finite edges into groups.
- **Least Squares** to merge the finite edges of groups into edge segments.
- **Ransac's Variation** to determine the 3D edge.

The first step (section 3.1) applies to generating the triangulated mesh surface using the BPA. The input data used for this is a 3D point cloud. In the second stage of the algorithm's implementation, the corresponding normal vector for each triangle of the triangulated surface is computed and the angle between the normal vectors for every pair of adjacent triangles of the triangulated mesh is calculated. In the third stage of the algorithm (section 3.3), two thresholds are used to prune edges that do not depict surface discontinuities. With these optimizations, the triangulated network is converted to an undirected Cartesian Graph (G) (Section 3.3) During the Graph's generation, edges within certain thresholds are kept. Then, the initial Graph (G) was converted into a Line Graph L(G) (Section 3.3). At last, the Line Graph is reverted into a set of grouped edges from the initial mesh, where each group corresponds to one single edge. During the fourth step of the implementation, the Least Squares method (section 3.4) is used to combine each group of the finite edges into line segments.
In the last stage, the line segments were concatenated as described in section 3.5 with other almost-parallel line segments close to each other using our proposed variation of RANSAC.

5. EXPERIMENTAL RESULTS

5.1 Simulated Data

To first assess the algorithm performance on noise-free data we used simulated data of 3D points generated on cube faces of different densities. The cube models contain well-defined edges and are therefore suitable for validating the assumptions of our methodology. The results do not depend on the point cloud density but only on the mesh quality. Provided that the cube has an optimal mesh and the plane surfaces are smooth, the edges may be detected even from the third stage of the algorithm's implementation without the utilisation of the Least squares and RANSAC stage. Figure 3, depicts two of the different cube models with varying density of 3D points and the triangulated surface representation, used in the experiments. A denser sampling with fifty 3D points in each face was also exploited.



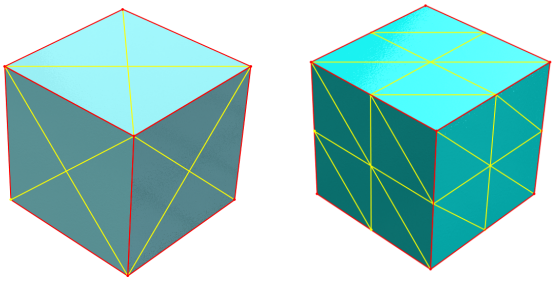


Figure 3. Top row: Triangular mesh of 3D points defined on cube faces with varying density. Bottom row: Detected edges (in red) and pruned ones (in yellow).

5.2 Real Data

A dense point cloud dataset representing the Temple of Demeter in Naxos, Greece was used to test the algorithm's efficiency in real data. The 3D points were extracted from images using the Multi-View Stereo process. The Temple's point cloud consists of more than three million points and has a sufficient amount of noise (Figure 4). All five stages of the algorithm implementation were needed for the detection of the edges in this dataset. The initial four stages i.e., the mesh generation, the computation of the mesh's edge values, the edge grouping through the graphs (Figure 5) and the least squares implementation (Figure 6) had a good execution time. However, RANSAC slowed down the process as the developed implementation needed a significant amount of time to run using large point clouds. The reason is because it performs thousands of iterations between the different line segments to find the best models and therefore to eliminate the noise of the data. In order to bypass that problem we tested the RANSAC variation using a smaller point cloud (Figure 7).

The proposed algorithm identifies the edges of an object, by automatically calculating the inclination angle between the adjacent triangles of the mesh. The only parameters that the user should define are those for the BPA algorithm (Section 3.1). Additionally, the user could define the threshold values θ1, θ2 (Section 3.3) to identify different edges, but this is not a necessary step. Hence, the use of the proposed algorithm is not highly parameterized and so it can be used by non-experts. On the one hand, the proposed algorithm tries to identify the object's edges with the best possible accuracy, in respect to the given data by including the least squares and RANSAC steps (Figure 7 and Figure 8). On the other hand, the execution time increases for large and medium point clouds and decreases with small point clouds i.e., less than 40000 points). The execution times in Table 2 are until the least squares step i.e., excluding RANSAC. With RANSAC the execution time increases for large and medium point clouds. The presented results in Figure 8, are gathered by terminating the RANSAC manually and so an improvement of the implementation of the RANSAC algorithm should be further investigated and optimized.

Figure 4. The dense point cloud of Demeter's temple.

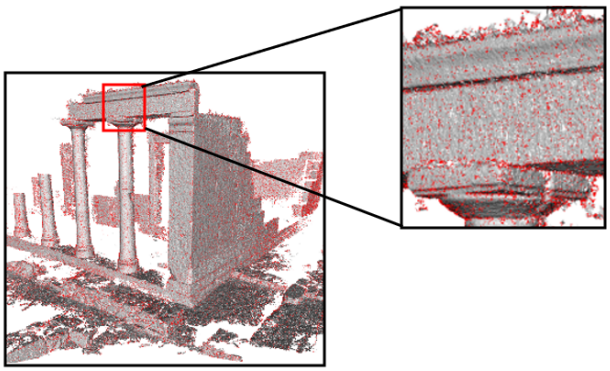


Figure 5. Visualisation of finite edges detection after graphs' implementation on the temple dataset.

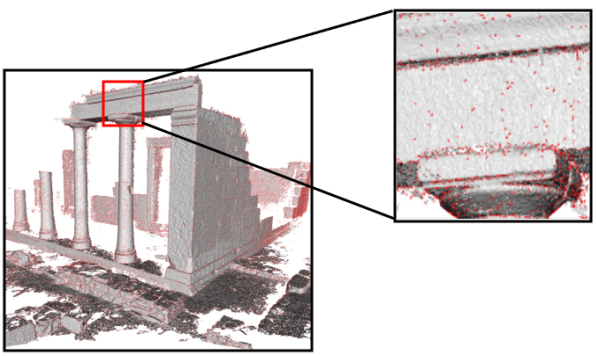


Figure 6. Visualisation of edge segments after least squares implementation on the temples algorithm.

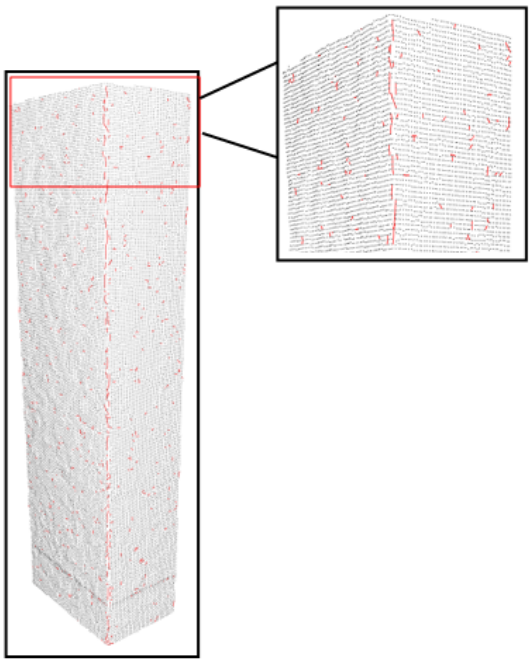


Figure 7 . Edge segments after least squares implementation.

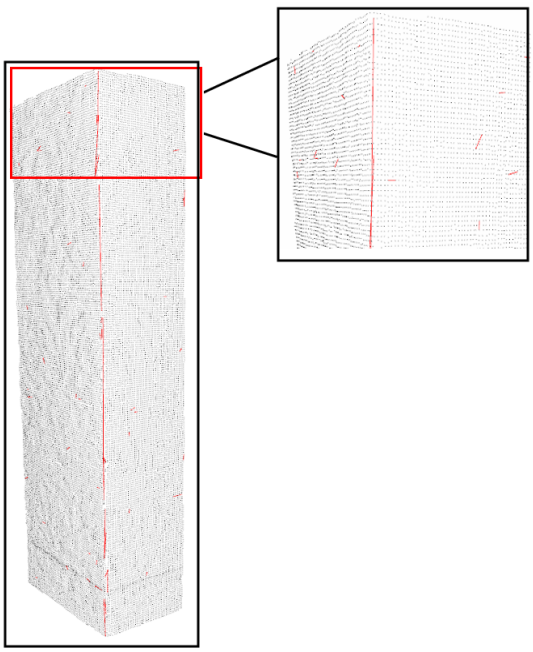


Figure 8. Final Edge detection after Ransac's implementation.

Experiment	Execution Time (seconds)
Temple of Demeter in Naxos (≈3.100.000 3D points, Fig 6)	≈ 48000 (≈ 13 mins, Including BPA)
Part of the Temple of Demeter in Naxos (≈ 32.000 3D points, Fig 7 and Fig. 8)	≈ 12 (Including BPA)
Cube 1 (Fig. 3)	< 1 (without BPA)
Cube 2 (Fig. 3)	< 1 (without BPA)

Table 2: Execution time for each experiment until the least squares step.

Furthermore, the proposed workflow gives promising results against noise. To be more specific, the experiments using the small dense point cloud depicted in Figure 7 and Figure 8 proves that the algorithm could handle noisy, real-world point clouds to detect the dominant 3D edges, i.e., the detected small edges attributed to micro-surface abnormalities depicted in Figure 7, have been almost eliminated in Figure 8. To sum up, the proposed algorithm has to be improved in terms of execution time but provides promising results against noise.

6. Conclusions

In this paper, we propose a method that performs 3D edge detection by exploiting the direction differences of normal vectors in 3D point clouds. The proposed approach was tested on both simulated and real-world data with promising results in terms of accuracy. However, a further optimization of the developed method should be performed to improve the execution time of the algorithm. Finally, the proposed method gives good results using small point clouds, demonstrating the effectiveness of the main idea.

Acknowledgements

This work was partially funded by the European Union's Horizon 2020 research and innovation programme FELICE (GA No 101017151) and Horizon Europe programme SOPRANO (GA No 101120990).

References

Abdellali, H., Frohlich, R., Vilagos, V., Kato, Z., 2021. L2D2: Learnable line detector and descriptor. In Intl. Conf. on 3D Vision (3DV).

Akinlar, C., Topal, C., 2011. EDLines: Real-time line segment detection by edge drawing. In Proc. *International Conference on Image Processing (ICIP)*, Brussels, Belgium, 2011, pp. 2837-2840.

Bazazian D., Casas J. and Ruiz-Hidalgo R., 2015, Fast and Robust Edge Extraction in Unorganized Point Clouds, In *Digital Image Computing: Techniques and Applications (DICTA),* Int. Conference IEEE, (https://imatge.upc.edu/web/sites/default/files/pub/cBazazian15.pdf).

Bao, T., Zhao, J., Xu, M., 2015. Step edge detection method for 3D point clouds based on 2D range images. 126(20), 2706–2710. https:// linkinghub.elsevier.com/ retrieve/pii/S0030402615005586.

Bernardini, F., Mittleman, J., Rushmeier, H., Silva, C., Taubin, G., 1999. The Ball-Pivoting Algorithm for Surface Reconstruction. IEEE Transactions on Visualization and Computer Graphics, vol. 5, no. 4, pp. 349-359.

Betsas, T. and Georgopoulos, A.: 3D Edge Detection and Comparison using Four-Channel Images, Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., XLVIII-2/W2-2022, 9–15, https://doi.org/10.5194/ isprs-archives-XLVIII-2-W2-2022-9-2022, 2022.

Betsas, T., Georgopoulos, A., 2022. Point-Cloud Segmentation for 3D Edge Detection and Vectorization. *Heritage* 5.4 (2022): 4037-4060.

Briese, C. and Pfeifer, N., 2008. Towards automatic feature line modelling from terrestrial laser scanner data. *International Archives of Photogrammetry & Remote Sensing and Spatial Information Science*, Volume XXXVII Part B5 pp. 463-468.

Canciani, M., Falcolini, C., Saccone, M. and Spadafora, G., 2013. From Point Clouds to Architectural Models: Algorithms for Shape Reconstruction. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Volume XL-5/W1, 20133D-ARCH 2013 - 3D Virtual Reconstruction and Visualization of Complex Architectures, 25 – 26 February 2013, Trento, Italy.

Canny, J. F., 1983. Finding edges and lines in images. Technical report, Massachusetts Inst of Tech Cambridge Artificial Intelligence Lab., DSpace@MIT Home, https://dspace.mit.edu/handle/1721.1/6939

Dhankhar, P., Sahu, N., 2013. A Review and Research of Edge Detection Techniques for Image Segmentation. International Journal of Computer Science and Mobile Computing - IJCSMC Vol.2, No. 7, pp. 86-92, ISSN 2320–088X.

Dolapsaki, M.M.; Georgopoulos, A. Edge Detection in 3D Point Clouds Using Digital Images. *ISPRS Int. J. Geo-Inf.* 2021, *10*, 229. https://doi.org/10.3390/ijgi10040229

Grompone Von Gioi, R., Jakubowicz, J., Morel, J.M., Randall, G., 2008. LSD: A fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32(4):722– 732

Kirsch, R. A., 1971. Computer determination of the constituent structure of biological images. Computers and biomedical research, 4(3), 315–328.

Kroon, D., 2009. Numerical optimization of kernel-based image derivatives. Short Paper University Twente, 3.

Liakopoulou, A.M., 2022. Development of a plane intersection algorithm for 3D edge determination. https://dspace.lib.ntua.gr/xmlui/handle/123456789/56782, Diploma Thesis, Lab of Photogrammetry, NTUA (in Greek).

Lu, X., Liu, Y., & Li, K. (2019). Fast 3D Line Segment Detection from Unorganized Point Cloud. *ArXiv, abs/1901.02532*.

Marr, D., Hildreth, E., 1980. Theory of edge detection. Proceedings of the Royal Society of London. Series B. Biological Sciences, 207(1167), 187–217.

Mitropoulou, A., Georgopoulos, A., 2019. An automated process to detect edges in unorganized point clouds. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, *4*, 99-105.

Nguatem, W., Drauschke, M., Mayer, H., 2014. Localization of Windows and Doors in 3d Point Clouds of Facades. II-3, 87– 94.

Pateraki, M., Baltsavias, E., 2004. Surface discontinuity modelling by LSM through patch adaptation and Use of edges. In: Proc. of the ISPRS Congress, IAPRS, Vol. 35, Part B3, pp. 522-527.

Poma, X. S., Riba, E., Sappa, A., 2020. Dense extreme inception network: Towards a robust CNN model for edge detection. Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 1923–1932.

Prewitt, J. M., 1970. Object enhancement and extraction. Picture Processing and Psychopictorics, B. Lipkin and A. Rosenfeld, Eds., New York: Academic Press, 1970, pp. 75-149.

Pautrat, R., Barath, D., Larsson, V., Oswald, M., Pollefeys, M., 2023. DeepLSD: Line Segment Detection and Refinement with Deep Image Gradients. In Proc. EEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, Canada,

Rodríguez Miranda, Á., Valle Melón, J. M. and Martínez Montiel J. M., 2008. 3D Line Drawing from Point Clouds using Chromatic Stereo and Shading. Digital Heritage. *Proceedings of the 14th International Conference on Virtual Systems and Multimedia VSMM 2008*. ISBN: 978-963-8046-99-4 pp. 77-84

Sobel, I., Feldman, G. et al., 1968. A 3x3 isotropic gradient operator for image processing. a talk at the Stanford Artificial Project in, 271–272.

Su, Z., Liu, W., Yu, Z., Hu, D., Liao, Q., Tian, Q., Pietikainen, M., Liu, L., 2021. Pixel difference networks for efficient edge detection. Proceedings of the IEEE/CVF International Conference on Computer Vision, 5117–5127.

Suárez, I., Buenaposada, J., Baumela, L. 2022. ELSED: Enhanced line segment drawing. *Pattern Recognition*.

Wang, Y., Ewert, D., Schilberg, D., Jeschke, S., 2014. Edge extraction by merging the 3D point cloud and 2D image data. *Automation, Communication and Cybernetics in Science and Engineering 2013/2014*, 773-785.

Xie, Y., Tian, J., Zhu, X. X., 2020. Linking points with labels in 3D: A review of point cloud semantic segmentation. IEEE Geoscience and Remote Sensing Magazine, 8(4), 38–59.

Zhang, H., Luo, Y., Qin, F., He, Y., Liu, X., 2021. Elsd: Efficient line segment detector and descriptor. In International Conference on Computer Vision (ICCV),