3D visibility analysis for planning of TLS networks

Pauline SPEIDEL, Volker SCHWIEGER, Germany

Key words: visibility analysis, terrestrial laser scanning, voxel, ray casting

SUMMARY

Terrestrial laser scanning (TLS) is a well-established method of capturing complex 3D objects on a large scale with high resolution. Often, more than one scan is required to capture the whole object. In this case, an optimal design of the TLS network would be very helpful. A key challenge therefore is ensuring the completeness of the resulting point cloud. This requires precisely determining visible and hidden object faces from different viewpoints. This paper presents a voxel-based algorithm for 3D visibility analysis. First, the triangulated mesh is voxelized, with edges and faces being approximated using the extended Bresenham algorithm. Afterwards, a visibility check is performed based on the principle of ray casting. The occupied voxels are examined to determine their visibility from a defined viewpoint. For this purpose, the voxels are transformed into spherical coordinates so that the relationship between the instrument viewpoint and the voxel is described directly in geometric terms and occlusions can be detected via the radial component. Normal vectors are also incorporated into the analysis, enabling a reliable distinction to be made between real and apparent self-occlusions, to prevent misclassifications caused by staircase artefacts on sloping faces.
This approach provides a foundation for the efficient planning of TLS networks and the optimisation of the number and coordinates of scan viewpoints.

3D visibility analysis for planning of TLS networks

Pauline SPEIDEL, Volker SCHWIEGER, Germany

1. INTRODUCTION AND MOTIVATION

In recent decades, there has been a significant increase in the use of measurement methods for the large-scale acquisition of object geometries, with terrestrial laser scanning (TLS) being one possible implementation (Xu, et al., 2021). A key aspect of many TLS projects is capturing the entire surface of the object, which usually requires several individual scans from different viewpoints in complex scenes due to occlusions. In this context, 'completeness' refers to the visibility of all surfaces, edges, and corners of the object in the overall point cloud, as well as sufficient overlap between the individual scans for registration. If a sensitivity analysis (Wujanz, et al., 2016) is also required to detect expected deformations, factors such as the recording rate and angle step size play an important role alongside completeness. The aim is therefore to find a multi-criteria optimisation method for the first- and second-order geodetic design of a TLS network. One way to make TLS more efficient in the future is to optimise scanner positions in terms of completeness, precision, cost-effectiveness and reliability.
Though there are already various approaches to planning the positioning of a TLS network, many of these studies are limited to a two-dimensional view. For example, Jia & Lichti (2019) describe a computer-aided planning system that minimizes the number of instrument sites and registration targets while maximizing reconstruction quality. However, their calculations are based on hierarchical resolutions and a weighted greedy algorithm and are therefore limited to two-dimensional analysis due to computational complexity. An approach for planning TLS scans in 3D environments is proposed by Noichl, et al. (2024). Their method is based on a visibility and coverage analysis of triangulated mesh representations of a scene. The combinatorial algorithm developed by Wujanz & Neitzel (2016) uses a three-dimensional (3D) model as input but focuses primarily on reducing the number of sites without considering the resulting point cloud quality. Additionally, to reduce complexity, two-dimensional (2D) maps are derived from three-dimensional (3D) models, requiring all sites to be simulated. Soudarissanane (2016) draws parallels between determining optimal TLS viewpoints and the art gallery problem in computer science, which is also two-dimensional. These studies demonstrate that a two-dimensional view alone is insufficient for describing the complete geometry of an object or environment. Since TLS provides three-dimensional measurements, it makes sense to take advantage of this and determine the optimal viewpoints in three-dimensional space as well, in order to take the complete geometry of the scene into account. However, a three-dimensional view increases computational complexity significantly due to more complex relationships.

A fundamental step in determining optimal scanning positions is the reliable identification of the faces of a three-dimensional object that are visible from a specific perspective. Analysing

visibility in complex 3D environments is essential for evaluating the completeness of the object in a point cloud, ensuring high-quality object capture. However, the question of visibility from a defined viewpoint is relevant not only for sensitivity analyses, but has also been investigated in a variety of other fields. Visibility analysis is also central to urban security strategies, for example, to minimize the risk of terrorist attacks (Aleksandrov, et al., 2019). Furthermore, visibility is crucial in technological applications such as autonomous vehicle navigation (Morales, et al., 2014).
Despite its wide range of applications, determining visibility in complex, three-dimensional scenes efficiently and accurately remains challenging. In particular, sensitivity analysis based on area-wide detection by terrestrial laser scanning requires the accurate identification of hidden and visible object areas to increase scanning efficiency and obtain a complete point cloud.

In this study, objects are divided into three-dimensional voxels to solve the 3D visibility problem. Within this voxel space, computer vision methods such as ray casting can be used to classify voxels (Hughes, 2014) as visible or non-visible from a chosen viewpoint, identifying the visible areas of the scene.

2. THEORETICAL BASICS

For a better understanding of the following algorithm, the underlying data structures will first be explained. In addition, basic methods known from computer vision and used in the present 3D visibility analysis will be introduced.

2.1 Mesh

A mesh or polygon network is a central data structure for representing two- and three-dimensional objects, which is used in particular in computer graphics, simulation and geometric modelling. It consists of a finite number of polygons, usually triangles or quadrangles, which can be used to efficiently approximate complex surfaces. The basic elements of a mesh are vertices, edges and faces, which define the geometry and topology of the network separately (Bommes, et al., 2013). Vertices V are points in space with known coordinates, edges connect these points, and faces F form the polygons to describe the topology. A mesh is closed ("watertight") if only internal edges exist.
Important structural properties are orientation, manifoldness and vertex degree. The orientation of a mesh is determined by the normal vectors of the faces, which consistently point outwards. This is relevant for backface culling, among other things (Hughes, 2014). Given that normal vectors remain constant on planar faces, this property is important in the approximation of curved surfaces, such as when approximating a sphere with an icosahedron. A mesh is considered manifold if it is closed and each edge is assigned to exactly two adjacent faces.

Deviations from this result in edge-like meshes with open structures. The vertex degree, i.e. the number of edges meeting at a point, plays a central role in the detection of edge structures.
Triangulated meshes are the most common form. They consist of triangles that are always planar and convex and allow for unambiguous interpolations. Their homogeneous structure enables efficient operations such as subdivision for smoothing, edge collapse for simplification, and edge swap for optimising triangle quality (Bommes, et al., 2013). High-quality triangulated meshes are characterised by triangles that are as equilateral as possible and large minimum interior angles.

2.2 Voxel

Voxels offer a fundamental representation option for volumetric data in digital image processing and computer graphics. As the three-dimensional counterpart to two-dimensional pixels, they represent data on a regular, straight-line grid in 3D space (Ososinski & Labrosse, 2014). Voxels are cubes of uniform size arranged on a grid and can thus be regarded, according to Hughes (2014) as a special form of the finite element method, in which continuous geometries are broken down into polyhedral volume elements. The position of a voxel is implicitly determined by its indexing in the grid, while additional attributes such as material properties or physical quantities (e.g. density, temperature) are stored as occupancy (Ray, et al., 1999).
The regular structure enables constant memory access and efficient hierarchical organisation, which is particularly relevant for large data sets. It also facilitates the implementation of fast ray casting algorithms, as intersections between rays and voxel edges can be easily calculated. Another advantage is the mesh-free representation: complex topologies can be represented without explicit face connections; neighbourhoods are created solely by shared faces, edges or corners of neighbouring voxels. This allows the direct extraction of geometric quantities such as volume or surface areas. Point clouds can also be processed efficiently using voxel rasterization by assigning a grid element to each point. This enables a uniform representation independent of the original point distribution (Xu, et al., 2021).
However, voxel-based representation is associated with significant memory requirements, as each volume element is stored explicitly. In addition, the discretization of continuous structures leads to approximation errors, loss of detail and artefacts. A finer resolution reduces these effects, but significantly increases memory requirements and computational effort. Therefore, a trade-off between accuracy and efficiency remains necessary (Ray, et al., 1999). For many applications it is necessary to combine voxel-based methods with compression techniques or adaptive resolution.

2.3 Bresenham algorithm

The Bresenham algorithm, developed by Jack Bresenham (Bresenham, 1965), is an efficient approach to rasterizing lines in discrete grid structures, originally developed for a two-dimensional space. While in the 2D version the line is continued step by step along the main

axis and an integer decision criterion determines horizontal or diagonal steps, this method can be extended to a three-dimensional space (Liu & Cheng, 2002). The basic idea is to project the line segment onto two of the three coordinate planes.

Given is a line segment with endpoints $P_S(x_S, y_S, z_S)$ and $P_e(x_e, y_e, z_e)$ in a voxel grid, where the coordinate differences are defined as $\Delta x = x_e - x_a$, $\Delta y = y_e - y_a$ and $\Delta z = z_e - z_a$. If $|\Delta x|$ is greater than the other gradients, the main direction of iteration is along the x-axis. Two decision variables are defined for the projections in the xy and xz planes:

$$\begin{aligned} \Delta_{yx,1} &= 2\Delta y - \Delta x \\ \Delta_{zx,1} &= 2\Delta z - \Delta x \end{aligned} \tag{1}$$

These are updated recursively:

$$\begin{aligned} \Delta_{yx,i+1} &= \begin{cases} \Delta_{yx,i} + 2\Delta y & \text{if } \Delta_{yx,i} < 0 \\ \Delta_{yx,i} + 2\Delta y - 2\Delta x & \text{if } \Delta_{yx,i} \geq 0 \end{cases} \\ \Delta_{zx,i+1} &= \begin{cases} \Delta_{zx,i} + 2\Delta z & \text{if } \Delta_{zx,i} < 0 \\ \Delta_{zx,i} + 2\Delta z - 2\Delta x & \text{if } \Delta_{zx,i} \geq 0 \end{cases} \end{aligned} \tag{2}$$

Depending on the sign of these variables, a decision is made whether, in addition to the x-coordinate, the y-coordinate and/or z-coordinate should also be increased by one. The procedure can be adapted analogously if $|\Delta y|$ or $|\Delta z|$ represents the largest coordinate change or if negative gradients exist.

The Bresenham algorithm generally enables efficient and robust rasterization of spatial line segments in voxel grids. Due to the purely integer operations, the 3D Bresenham algorithm remains highly performant despite the extension and is particularly suitable for voxel-based visibility analyses. Nevertheless, the approximated line may contain discretization errors.

2.4 Ray casting

Ray casting is a fundamental technique in computer graphics that is primarily used to determine the visibility and representation of objects. In rendering, a ray is emitted for each pixel through its centre into the 3D scene to determine the nearest intersection with an object. This intersection determines the visible surface as well as its colour and lighting (De Berg, et al., 2008), (Hearn & Baker, 1997). Ray casting can thus be interpreted as a precise variant of the depth buffer, with the main difference being direct ray tracing instead of comparing depth values. Disadvantages arise from artefacts caused by the finite size of the pixels. These can be reduced by super sampling (Jin, et al., 2009). Unlike ray casting, ray tracing continues to track rays after the first impact, allowing reflections, refractions and shadows to be simulated. This increases

the realism, with the expense of computing power (Hughes, 2014). However, this additional effort is unnecessary for this application of 3D visibility analysis, since only the initial point of laser scanner impact is of interest and multipath effects are not considered.

In addition to rendering, ray casting is also used for voxel-based visibility analysis. Here, the ray is moved step by step through a voxel grid until a collision point with an occupied voxel is detected (Ray, et al., 1999). Efficient discrete algorithms are used for traversal, in particular the DDA approach by Amanatides & Woo (1987) or an extended Bresenham algorithm that maps the line of sight as a discrete 3D line. While octrees can contribute to memory optimisation, a uniform voxel grid greatly simplifies the calculation.

The efficiency of a ray casting system depends largely on the memory organisation and the traversal strategy, as the memory address and possible intersections must be calculated for each voxel (Ray, et al., 1999). A decisive advantage lies in the high degree of parallelisation: each ray is processed independently, so large scenes or voxel grids can be analysed with a significant increase in performance.

3. ALGORITHM

The implemented algorithm can perform a visibility analysis for both 3D planning data, such as CAD models, and for existing point clouds. In both cases, the data is discretised using a voxel representation. For 3D models, these should be provided as a three-dimensional triangulated mesh described by vertices and faces. This data structure enables the transformation of the spatial body into voxels. In the case of real point clouds, a voxel representation is likewise generated.
The subsequent visibility analysis, based on the voxel representation, is independent of the type of input data and uses the ray casting principle to determine which voxels are visible from a freely selected viewpoint. Figure 1 and Figure 2 provide a schematic representation of the processes involved in the discretization algorithm. Figure 3 shows the visibility analysis process.

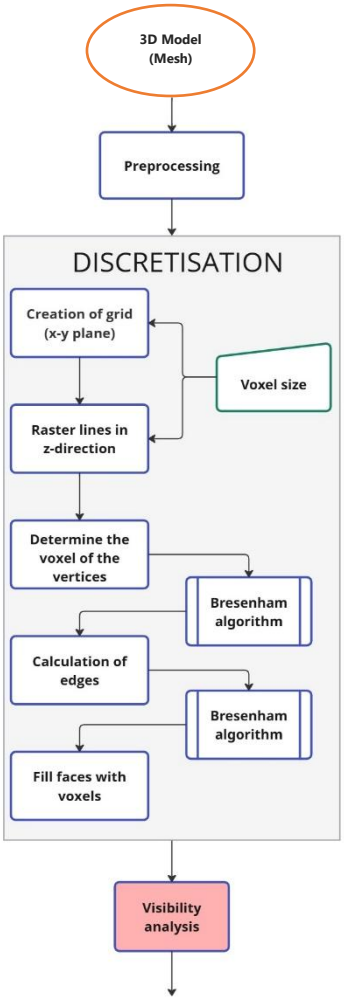


Figure 1: Workflow of voxel-based discretisation of meshes

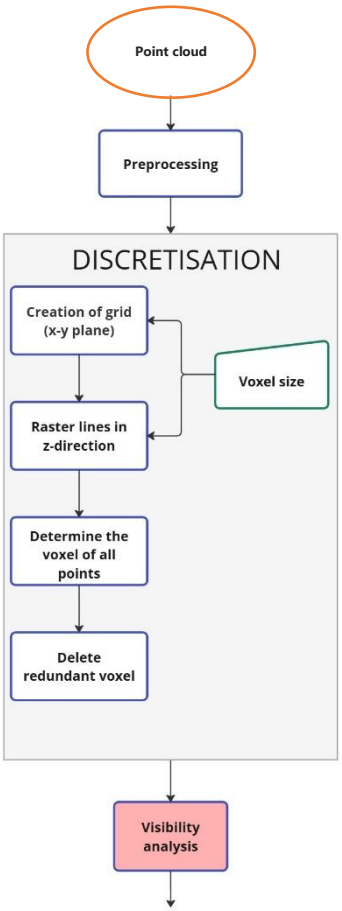


Figure 2: Workflow of voxel-based discretisation of point clouds

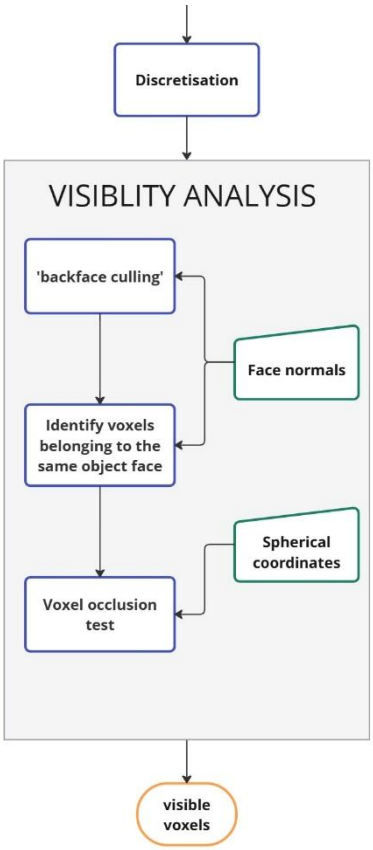


Figure 3: Workflow of 3D visibility analysis

3.1 Discretization of 3D-meshes using voxels

The discretization of the three-dimensional object signifies a pivotal step in which the continuous geometry of the object or scene is divided into a finite number of discrete elements. The utilisation of voxels leads to the uniform subdivision of space into cubes of equivalent dimensions. These regular structures can be processed and stored efficiently. This property renders them well suited for parallelisation.

3.1.1 Voxelization of the vertices

The initial step is to identify the voxels in which the vertices of the 3D mesh and the selected instrument position are located. In order to accomplish this objective, a bounding box must be determined to enclose the entire scene and thereby specify the extent of the voxel grid. The grid resolution depends on the selected voxel size. In order to reduce the memory requirements

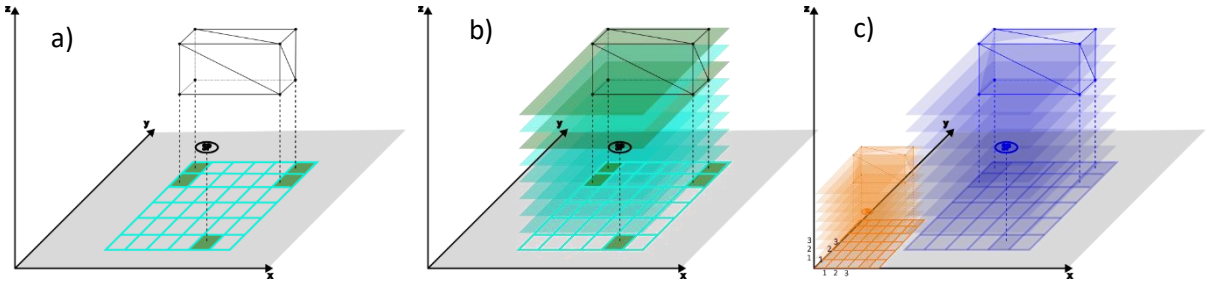


Figure 4: Creation of the voxel grid and identification of the voxels containing the vertices

during the discretization of the 3D object, the position and the height are first separated when determining the voxels of the vertices. Initially, a two-dimensional grid is formed in the x-y plane, with a structure that corresponds to pixels. The vertices are projected onto this uniform grid to determine the position of the corresponding voxel, as shown schematically in Figure 4a.

The next step is to expand to a three-dimensional view (sketched in Figure 4b). To do this, the respective z-values of the vertex voxels are interpolated using a discrete grid line that represents the vertical extension of the voxel grid. In the final step, as shown in Figure 4c, the voxel grid is transformed. To achieve a voxel size of 1, scaling is necessary. This enables the voxels to be stored in an integer matrix. This simplifies the algorithms used later, such the extended Bresenham algorithm. The voxel grid is transformed so that the initial voxel, containing the instrument view point, is located at the origin with the coordinates (0,0,0). As explained in section 3.2, this is necessary for the final visibility analysis.

3.1.2 Approximation of edges in voxel space

Once the voxels of the vertices have been identified, they are connected by discrete edges according to the topological structure of the mesh. This is done using the three-dimensional extended Bresenham algorithm, which is described in section 2.3. The edges are defined as vectors between two vertices and are approximated by voxels. The algorithm is based on the main direction of movement (axis with the largest coordinate difference) and an error term that measures deviations from the ideal line. If this error exceeds a threshold value, a corrective

movement is made along another axis. This is repeated iteratively until the target voxel is reached.

3.1.3 Filling the faces with voxels

For the complete representation of the 3D object in discrete voxel space, the faces described by the discrete edges must be filled with voxels according to the topology from the description. First, the longest edge of the respective triangle is determined, which then functions as the main edge. This is in this example, shown in Figure 5, the lower edge consisting of eight voxels. For each voxel of the main edge, a line is mapped to an opposite edge point, again using the extended Bresenham algorithm to discretely approximate the connecting line. This is visualized in Figure 5 by the blue arrows.

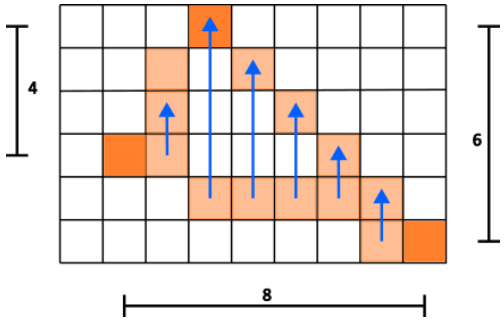


Figure 5: Filling the faces with voxels

3.2 Voxilization of a point cloud

When a point cloud is used as the data basis for a visibility analysis, no complex discretization is necessary because the data is already in a discrete form. First, a voxel grid is placed in the scene, depending on the chosen voxel size. Subsequently, for each point in the point cloud, the corresponding voxel is determined, similarly to the voxelization of the vertices described in section 3.1.1. To reduce the amount of data, all redundant voxels are removed. This process is also shown in Figure 2.
The voxel-based representation presents an effective alternative to mesh representations of 3D point clouds and can simplify the processing and analysis of 3D data, as explained by Xu, et al. (2021). Since no complex surface reconstruction is needed, this approach is robust against noise and outliers within the point cloud. Furthermore, issues such as holes or uneven point density do not significantly impact the voxel representation, as each point is treated independently.

3.3 3D visibility analysis in voxel space

The central task of the implemented algorithm is to solve the 3D visibility problem. This is based on the previously performed discretization of the object under consideration into a voxel

grid. A ray casting method is used to check which voxels are visible from a given observer's perspective. It is assumed that the objects are completely opaque. The task is to identify those voxels that are not obscured by nearby voxels and can therefore be considered visible. Inspired by Hartmann, et al. (2023), spherical coordinates are used for this purpose. Due to the displacement of the voxel grid described in section 3.1.1, the observer's viewpoint is located at the origin (0,0,0). The spherical coordinates (ϕ, θ, r) therefore directly describe the geometric relationship between the viewpoint and the voxel. The radius r indicates the distance of the point from the viewpoint, while the azimuth ϕ and the horizontal angle θ define the orientation of the connecting line between the instrument and the point in space.
A voxel is classified as visible, if at least two of its eight corner points lie in the field of view, so that partial occlusion by other voxels is acceptable. For verification purposes, the representation of each voxel in spherical coordinates is first calculated from its Cartesian coordinates (x, y, z). The angle range covered by the voxel from the viewpoint is then determined. All objects within this angle range that have a smaller radial distance r obscure the voxel under consideration. Figure 6 illustrates this by showing the observer's viewpoint in magenta and the dashed area behind the grey voxel marking the obscured regions. The boundaries of this area are determined by the minimum and maximum azimuth and elevation angles of the spherical coordinates of the voxel vertices. Each vertex is checked to see if it lies within this angular range; if the radial coordinate is greater than that of the obscuring voxel, the point is considered invisible.

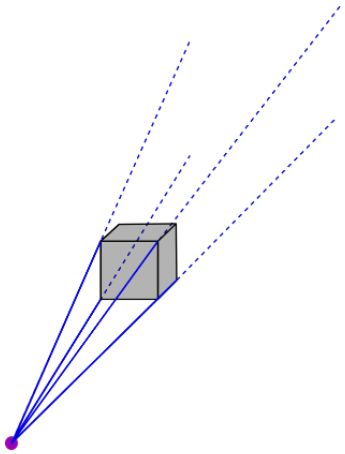


Figure 6: Coverage area of a voxel (dashed line)

When discretizing inclined faces, staircase structures occur due to the voxel approximation, so that neighbouring voxels of the same face can obscure each other, even though the entire face would be visible in reality. If a point on a plane is visible, the entire plane is visible from that viewpoint, provided it is not obscured by other objects.

In Figure 7 the blue sloped surface is discretely approximated by voxels. This results in the typical staircase structure. The front voxel of each step is clearly visible, as shown by the green connections to the magenta viewpoint. However, the orange sight lines of the voxels behind them are blocked by the front voxels. To avoid this false detection, the normal vectors of the

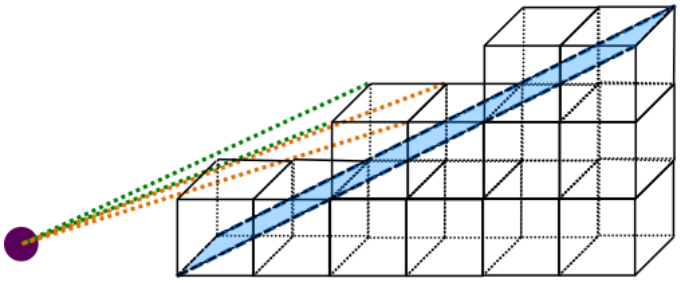


Figure 7: Discrete representation of the blue oblique surface using voxels. The line of sight (orange) from the viewpoint (magenta) to the rear voxel is blocked by the voxel in front of it.

faces are also included in the visibility check. For this purpose, each voxel is first assigned the normal vectors of the corresponding triangles of the underlying mesh. Afterwards, those normal vectors whose faces are not geometrically visible due to their orientation are removed. This is done by calculating the scalar product between the normal vector $\boldsymbol{fn}$ and the connection vector $\boldsymbol{LoS}$ from the voxel center to the observer's viewpoint. A scalar product less than or equal to zero means that the angle between the two vectors is at least 90°, and thus the edge or back side of the face cannot be visible, while a scalar product greater than zero means that the face is visible. The following applies:

$$\begin{aligned} \boldsymbol{fn} \cdot \boldsymbol{LoS} &\leq 0 \rightarrow \text{not visible}, \\ \boldsymbol{fn} \cdot \boldsymbol{LoS} &> 0 \rightarrow \text{visible}. \end{aligned} \tag{4}$$

This procedure corresponds to backface culling (Hughes, 2014), which is well known in computer graphics at the mesh level. In the following visibility check, occlusions are only taken into account if they are caused by voxels whose normal vectors differ from those of the target voxel.

In addition, an area directly below the terrestrial laser scanner is generally classified as non-visible. This ground gap results from the limited vertical scanning angle of the instrument and from the shadowing caused by the housing. As part of the analysis, an elevation angle range outside the measurement field is therefore defined, and all voxels within this range are classified as non-visible.

The result of the 3D visibility analysis is a matrix of all visible voxels. In addition, the associated possible normal vectors are stored so that the results can also be used for further geometric analyses.

4. EVALUATION OF THE ALGORTHM

A synthetic 3D model is used to evaluate an algorithm developed for 3D visibility analysis based on voxels. This model is available as a triangulated mesh. The object in Figure 8 has a tilted face and a protruding extension that can cause mutual occlusions depending on the perspective. Figure 9 shows the model completely voxelized.

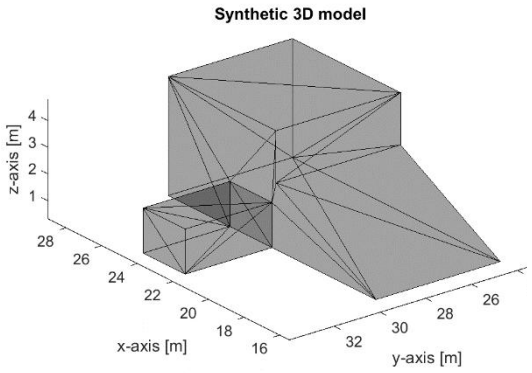


Figure 8: Mesh of the synthetic 3D model

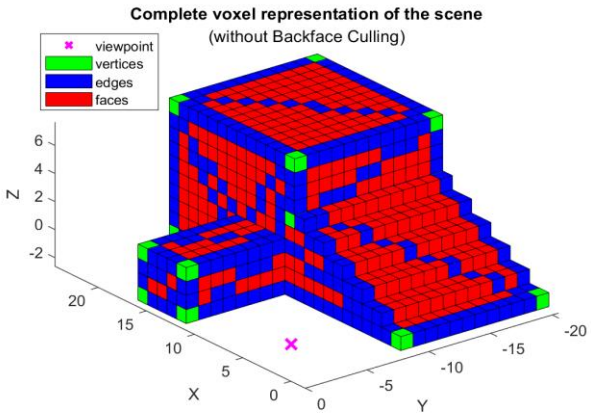


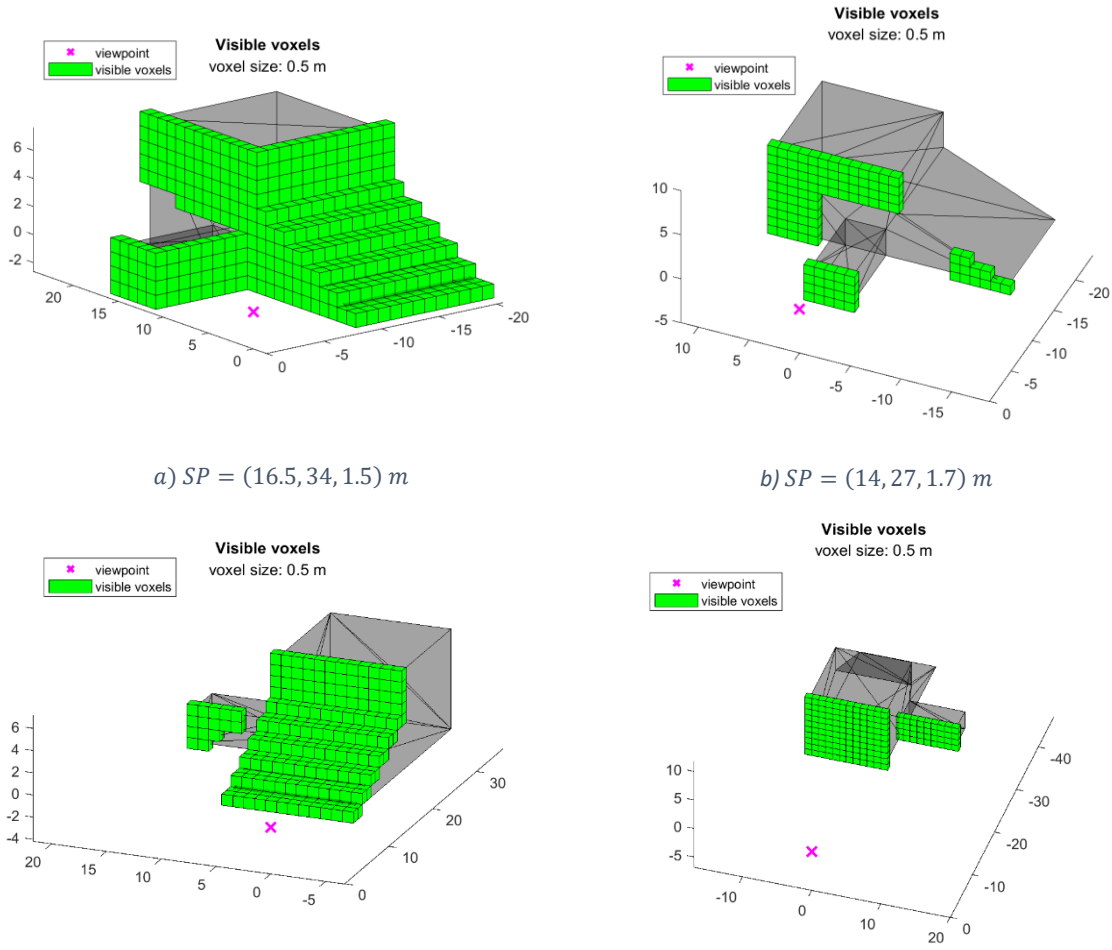
Figure 9: Complete voxel representation of the 3D object

4.1 Results of the visibility analysis of the synthetic 3D model

The results of the 3D visibility analysis algorithm are evaluated visual by various scenarios. As no reference data on current visibility is available, this is the only possibility.

Figure 10 presents the results of the visibility analysis for various freely chosen viewpoints at a voxel resolution of 0.5 m. The distribution of visible voxels corresponds well to the expected outcome: regions of the objects whose line of sight is obstructed by foreground structures are reliably identified as invisible. Likewise, all surfaces oriented away from the observer are consistently classified as non-visible. The results demonstrate the efficacy of the algorithm in ensuring a stable and accurate separation between visible and hidden areas from diverse viewing angles.

Figure 11 shows the results of the visibility analysis without including the surface normal vectors. In this case, gaps are visible, especially along the inclined surface. Individual voxels within an otherwise contiguous area are incorrectly classified as invisible, even though they are not actually obscured by other parts of the object. This effect results from discretization



a) $SP = (16.5, 34, 1.5)\ m$

b) $SP = (14, 27, 1.7)\ m$

c) $SP = (10, 25, 1.7)\ m$

d) $SP = (40, 28, 1.2)\ m$

Figure 10: Results of the 3D visibility analysis from different viewpoints

artefacts in which contiguous voxels of the same surface incorrectly obscure each other, even though the surface itself is actually completely visible.

Figure 12, on the other hand, shows the same scene after normal vectors have been included into the classification of voxels as visible or invisible. By integrating orientation information to the visibility check, false occlusions caused by voxels of the same surface are effectively

eliminated. This refinement ensures that contiguous surfaces, especially sloped or curved ones, are correctly rendered as visible. This improves the correctness of the visibility analysis.

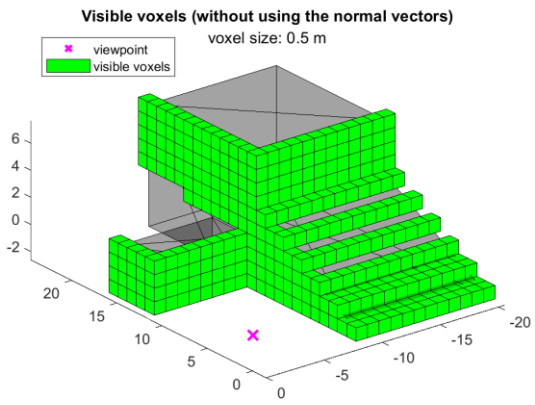


Figure 11: Result of the visibility analysis without including the normal vectors

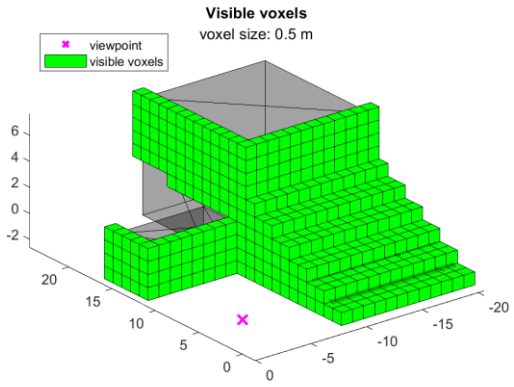


Figure 12: Result of the visibility analysis with normals

4.2 Results of the visibility analysis of a point cloud

In addition to triangulated meshes, the algorithm can also use real-world point clouds as input. For validation with real-world data, a point cloud of the front side of the 14-meter-high

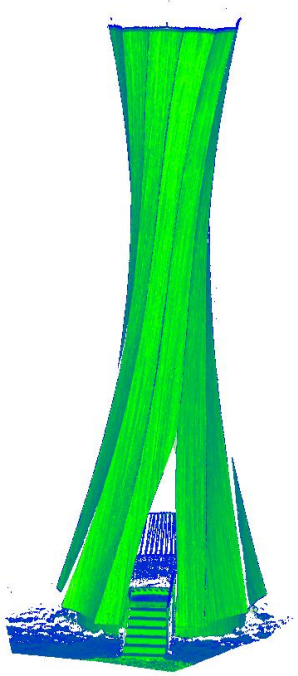


Figure 13: Point cloud of the Urbachtower

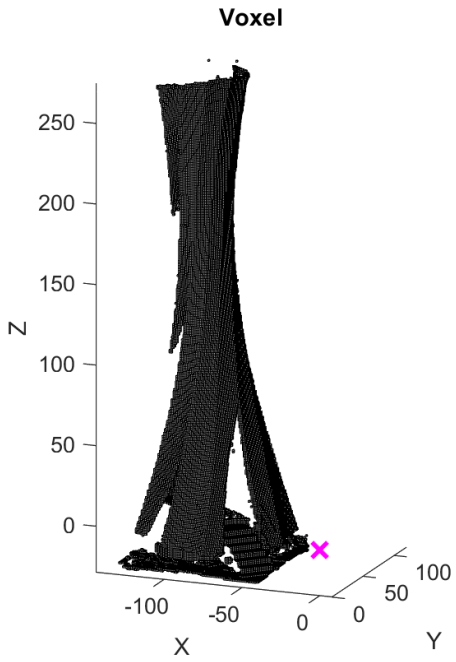


Figure 14: voxel representation of the Urbachtower (voxel size: 0.05 m)

Urbachtower, a wooden tower located near Stuttgart is used (see Figure 13). Figure 14 presents the resulting voxel representation, which accurately captures the towers doubly curved surface. In Figure 15, a sample viewpoint is selected, marked with a magenta-colored cross, for which the visibility analysis is performed. To better visualize the voxels, a zoomed-in section is shown in Figure 16. It is evident that the voxel representation effectively captures the shape of the tower.
So far, it has only been tested whether the algorithm can generally handle real-world point clouds. A comparison of the results from the visibility analysis with actual measurements has not yet been performed.

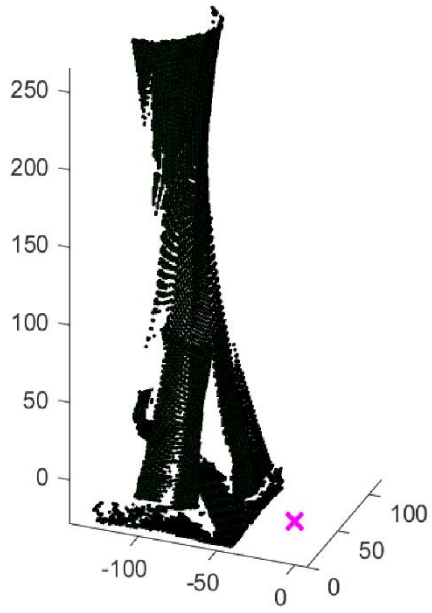


Figure 15: visible voxels from the magenta-coloured viewpoint (voxel size: 0.05 m)

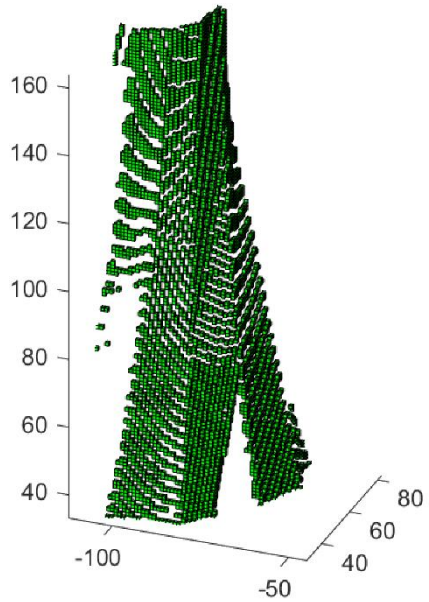


Figure 16: Close-up of the visible voxels

4.3 Impact of voxel size

The voxel size is a crucial parameter of the algorithm that influences both quality and computing time. It determines the resolution at which the object is approximated in the voxel grid, directly influencing the number of voxels required to model the scene. Depending on the complexity of the scene, a certain voxel size is necessary to reproduce all the necessary details. However, the available computing resources also limit the maximum number of voxels.
Table 1 shows the runtime of the algorithm and the number of voxels for different voxel sizes, based on a synthetic 3D model. These calculations were performed using a 12th Gen Intel® Core™ i7-1265U processor with 32 GB of RAM, with MATLAB R2023b running on Windows 11.

Table 1: Comparison of the influence of different voxel sizes

Voxel size	1 m	0,5 m	0,1 m	0,05 m
Processing time	1,114 s	1,272 s	281,130 s ≈ 4,69 min	1102,509 s ≈ 18,38 min
Required memory for voxel representation	0,0055 MB	0,0236 MB	0,6031 MB	2,4171 MB
Number of voxels before visibility check	241	1030	26352	105606
Number of visible voxels	53	237	5902	23083
Proportion of visible voxels in the overall model	78,01%	76,99%	77,60%	78,14%

As can be seen in Table 1, the number of voxels increases almost cubically as the voxel size decreases. Halving the voxel size theoretically increases the number of voxels by a factor of eight ($2^3 = 8$).
At the same time, both the processing time of the algorithm and the necessary memory requirements are directly related to the number of voxels.
To get a better sense of the algorithm's runtime, an estimation of the computation time for a construction site is carried out. The calculation time is significantly influenced by the scene size and the desired resolution, which determines the resulting number of voxels. For instance, processing 100000 voxels take approximately 18 minutes. With this voxel count, a construction site with a ground area of 50 m x 100 m and a building height of 15 m can be covered using a voxel size of 1 m. However, one advantage of using voxels is that most computations can be easily parallelized. This allows for at least a reduction in runtime. When comparing the proportion of visible voxels in the total object for a specific viewpoint, no influence of voxel size on the results of the 3D visibility analysis can be detected. It only affects the discretization error in the approximation of the object with voxels.
A balance must be struck between discretization quality and computing time, depending on the individual scene.

5. CONCLUSION AND OUTLOOK

This paper presents an algorithm for determining the visible surfaces of a three-dimensional object from a freely selectable instrument position. This method is based on voxelizing the object, resulting in a discrete representation that can be efficiently processed using computer vision techniques. Visibility analysis is performed according to the ray casting principle in spherical coordinates, whereby the geometric relationship between the viewpoint and the voxel can be clearly described. This allows us to check whether a voxel lies in the occlusion area of a closer voxel and is therefore classified as invisible. Due to the discrete representation of the

scene using voxels, the implemented algorithm is capable of handling both 3D models, such as those derived from planning, and real point clouds.

It is essential to include surface normal vectors in the visibility analysis. Without this, staircase artefacts caused by the discrete voxel structure led to misclassifications. This is especially important on inclined surfaces, where contiguous voxels of the same surface incorrectly obscure each other. Using normal vectors reliably solves this problem. Additionally, evaluating the normal vectors enables the detection of unfavourable intersection configurations during the visibility check and allows for further classifications. Furthermore, normal vectors can play a central role in optimising viewpoint planning for the sensitivity analyses.

To evaluate the correctness of the developed algorithm, a real-world validation is planned for the future. The results from visibility analysis will be verified on a real object, and a comparison between the expected and actual outcomes will be compared.

The evaluation also shows that the choice of voxel size has a significant impact on the efficiency of the algorithm, while the results of the visibility analysis remain largely robust when this parameter is varied. The ability to parallelize voxel processing is expected to save a significant amount of computing time. The algorithm will be further improved in this regard in the future. Nevertheless, selecting an appropriate voxel size remains a significant challenge, particularly in large-scale TLS projects, where it is of great importance for practical application.

Overall, the voxel-based 3D visibility analysis presented here is an important first step towards making TLS projects more efficient. It establishes a methodological foundation for the systematic optimisation of scanner positioning. This enhances the completeness of resulting point clouds and the quality of sensitivity analyses. In the future, this method will be incorporated into higher-level optimisation approaches to further automate and enhance the efficiency, accuracy, cost-effectiveness and reliability of terrestrial laser scanning.

REFERENCES

Aleksandrov, M. et al., 2019. Voxel-based visibility analysis for safety assessment of urban environments. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 23 09, Volume IV-4/W8, pp. 11-17.

Amanatides, J. & Woo, A., 1987. *A Fast Voxel Traversal Algorithm for Ray Tracing,* s.l.: Eurographics Association.

Bommes, D. et al., 2013. Quad-Mesh Generation and Processing: A Survey. *Computer Graphics Forum*, 9, 32(6), pp. 51-76.

Bresenham, J., 1965. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, Volume 4, pp. 25-30.

De Berg, M., Cheong, O., Van Kreveld, M. & Overmars, M., 2008. *Computational Geometry: Algorithms and Applications.* Berlin, Heidelberg: Springer.

Hartmann, J., Heiken, M., Alkhatib, H. & Neumann, I., 2023. Automatic quality assessment of terrestrial laser scans. *Journal of Applied Geodesy*, 26 10, 17(4), pp. 333-353.

Hearn, D. & Baker, P., 1997. *Computer Graphics, C Version.* 2 ed. s.l.:Pearson Education.

Hughes, J. F., 2014. *Computer graphics. Principles and practice.* 3 ed. Upper Saddle River, NJ: Addison-Wesley Educational Publishers Inc.

Jia, F. & Lichti, D. D., 2019. A Model-Based Design System for Terrestrial Laser Scanning Networks in Complex Sites. *Remote Sensing*, 25 07, 11(15), pp. 17-49.

Jin, B. et al., 2009. *Selective and adaptive supersampling for real-time ray tracing.* New Orleans Louisiana, ACM, pp. 117-125.

Liu, X.-W. & Cheng, K., 2002. Three-dimensional extension of Bresenham's algorithm and its application in straight-line interpolation. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 01 03, 216(3), pp. 459-463.

Morales, Y. et al., 2014. *Visibility analysis for autonomous vehicle comfortable navigation.* Hong Kong, China, IEEE, pp. 2197-2202.

Noichl, F., Lichti, D. D. & Borrmann, A., 2024. Automating adaptive scan planning for static laser scanning in complex 3D environments. *Automation in Construction*.

Ososinski, M. & Labrosse, F., 2014. *Multi-viewpoint Visibility Coverage Estimation for 3D Environment Perception - Volumetric Representation as a Gateway to High Resolution Data:.* Lisbon, Portugal, SCITEPRESS - Science and and Technology Publications, pp. 462-469.

Ray, H., Pfister, H., Silver, D. & Cook, T. A., 1999. Ray casting architectures for volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 5(3), pp. 210-223.

Soudarissanane, S., 2016. *The geometry of terrestrial laser scanning; identification of errors, modeling and mitigation of scanning geometry,* s.l.: s.n.

Wujanz, D. et al., 2016. Survey Configuration for. *Allgemeine Vermessungs-Nachrichten*.

Wujanz, D. & Neitzel, F., 2016. Model Based Viewpoint Planning for Terrestrial Laser. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 15 06, Volume XLI-B5, pp. 607-614.

Xu, Y., Tong, X. & Stilla, U., 2021. Voxel-based representation of 3D point clouds: Methods, applications, and its potential use in the construction industry. *Automation in Construction*, 06.

Yusheng Xu, X. T. U. S., 2021. Voxel-based representation of 3D point clouds: Methods, applications, and its potential use in the construction industry. *Automation in Construction*.

ACKNOWLEDGEMENTS
The research published in this article is supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under project number 536596365. The authors cordially thank the DFG.

BIOGRAPHICAL NOTES

M.Sc. **Pauline Speidel**

2018 – 2025	Studies of Geodesy and Geoinformation in Germany (University of Stuttgart)
since 2018	Research Associate at Institute of Engineering Geodesy, University of Stuttgart

Prof. Dr.-Ing. habil. Dr. h.c. **Volker Schwieger**

1983 – 1989	Studies of Geodesy in Hannover
1989	Dipl.-Ing. Geodesy (University of Hannover)
1998	Dr.-Ing. Geodesy (University of Hannover)
2004	Habilitation (University of Stuttgart)
since 2010	Professor and Head of Institute of Engineering Geodesy, University of Stuttgart
2015 – 2018	Chair of FIG Commission 5 ‘Positioning and Measurement’
2016 – 2021	Dean of Faculty of Aerospace Engineering and Geodesy, University of Stuttgart
2019	Dr. h.c. Technical University of Civil Engineering, Bucharest, Romania
2022	Honorary Member of FIG

CONTACTS

M.Sc. Pauline Speidel / Prof. Dr.-Ing. habil. Dr. h.c. Volker Schwieger
University of Stuttgart
Institute of Engineering Geodesy
Geschwister-Scholl-Str. 24 D
D-70174 Stuttgart
GERMANY
Tel. +49/711685-84061 | -84040
Email: pauline.speidel@iigs.uni-stuttgart.de | volker.schwieger@iigs.uni-stuttgart.de

Web site: https://www.iigs.uni-stuttgart.de/