TECHNICAL TRANSACTIONS 12/2017
CZASOPISMO TECHNICZNE 12/2017
COMPUTER SCIENCES
DOI: 10.4467/2353737XCT.17.215.7758

Jerzy Orlof (jerzy.orlof@pk.edu.pl)
Faculty of Physics, Mathematics and Computer Science, Cracow University of Technology

Point cloud based viewshed generation in AutoCAD Civil 3D

Wyznaczanie wykresu widoczności na podstawie chmury punktów w programie AutoCAD Civil 3D

Abstract

We propose a new method for a point cloud based viewshed generation. We start with a brief summary of existing methods for viewshed generation and then present ours based on point clouds. The method is implemented in AutoCAD Civil 3D. Moreover, we present an idea of how to deal with huge point clouds using AutoCAD Civil 3D and how to automatize this process.

Keywords: viewshed; point cloud, digital modelling; ray tracing; LiDAR

Streszczenie

W artykule została przedstawiona metoda tworzenia wykresu widoczności na podstawie chmury punktów. Zaprezentowano zarówno znane dotychczas metody wyznaczania wykresów widoczności jak i metodę bazującą na chmurze punktów zaimplementowaną w oprogramowaniu AutoCAD Civil 3D. Omówiono również nowe rozwiązanie problemu przetwarzania dużej chmury punktów za pomocą programu AutoCAD Civil 3D oraz propozycję automatyzacji tego procesu.

Słowa kluczowe: wykres widoczności, chmura punktów, model cyfrowy, śledzenie promienia, LiDAR

1. Introduction

Together with the development of technologies, it is important to consider evolving data types, such as point clouds [1]. The particular structure of point clouds provides new opportunities for 3D modelling and visualization. Nowadays, any professional project, for example of a building, should be tested to fit urban design and landscape planning. Thanks to point clouds such tests are easier to conduct. Both during design and construction, point clouds can be used for visibility purposes, from public safety to environmental protection, traffic or research. To check the visibility using a point cloud, a viewshed must be generated. A viewshed [2] is a geographical map generated for a fixed position. This map contains and differs between two types of zones: (1) in line-of-sight (visible) (2) and not visible from a selected position. Moreover, the grade of visibility can be shown.

Generation of a viewshed from a point cloud is not an easy task: point clouds can be huge and difficult to handle, and the viewshed calculation can be time-consuming and generate errors. In this article, we present a solution to this problem introducing a new method for point cloud based viewshed generation.

Viewshed applications depend on the area that they cover. A viewshed can contain areas such as a building with its terrain, a village or even a whole city. In particular, an example of a viewshed application for a car driver shows the part of the road that he/she can see from his/her point of view. Other examples include: change of the visibility due to the insertion of a new building, best placement of security cameras (to limit blind spots), both indoors as well as outdoors for city monitoring, where at least one camera should be able to follow a moving object. In marketing, viewsheds are used to decide the position of a banner advertisement so that it is visible from as many roads as possible. A combination of viewsheds (shadow analysis) with solar radiation and 3-D city models allows the sustainability of the rooftops for solar panels hosting to be evaluated. Finally, viewshed analysis is used in the visual impact assessment and landscape planning to find the most sustainable scenarios for the future

2. General concept

Historically, the visibility was checked through empirical methods, such as an architectural model with a light source.

Nowadays, the visualization of tridimensional data is done using numerical methods, based on GIS (Geographical Information System). These systems use algorithms to verify the visibility from a specific position and create a viewshed. In particular, GIS can visualize DEM (Digital Elevation Model) as heightmaps or TIN (Triangular Irregular Network) surfaces. A heightmap is a raster image made of a matrix of values (i.e. RGB) in a grid while a TIN is a vectorial surface created from triangles. If the resolution of the model (raster or vectorial) is insufficient in relation to the dimension of the target object, the resulting model can be irregular and has to be complemented by additional measurements points to increase the sampling resolution.

Following the development of digital techniques, visibility check algorithms based on rendering (computer graphics) were implemented in GIS programs. One of such algorithms is Ray tracing [3] created by A. Appel in 1968. This algorithm is used to simulate the propagation of the light waves and designate the shadows of objects. It generates a viewshed as an image with brightened/shadowed objects visible/invisible from the light source point.

The Ray tracing method works as follows. First, a ray is sent from a light source. The ray is reflected by any 3D object encountered along its path and travels back to the camera mounted above the light source. The ray can't penetrate objects, so only those visible from the light source are illuminated.

Recently, research has been directed towards the development of a new data structure: the point cloud. Point clouds contain information about the terrain that includes both natural elements and those created by people. A point cloud is a set of points with a structure assigned to each of them. This structure always stores information about the coordinates (x,y,z), and optionally also colour (RGB), intensity, return time or echo number related to the landform. It is not possible to create a viewshed directly from a point cloud because the points are not connected in any way and so they do not describe a surface. To generate a viewshed, first one has to create a surface from the point cloud, usually a TIN [4]. The next step is to set up a light source. Finally, the viewshed can be generated from the Ray tracing method.

2.1. Point cloud generation methods

The performance of the methods for the generation of a point cloud depends on the size of the area that the point cloud should cover.

Point clouds can be generated using different types of scanners. Widely used are 3D scanners, which can be divided into two groups: contact and non-contact scanners.

Contact scanners require the physical touch of the rigid or articulated arm of the scanning machine with the object. That is why this method may result in damage to the target body and so can be too invasive for some measurements. The process itself is done through the gliding of the rigid arm scanner along the measurement line or through the rotation of the articulated arm. In both cases, carefully mounted sensors on the arms collect data points according to the shape of the scanned bodies.

Non-contact scanners are based on a completely different approach. These scanners emit radiation (light, ultrasound or x-ray) and collect it back after it's reflected by the target object. The time between the emission and collection of the radiation is directly related to the distance to the target. As the name suggests, a rangefinder is an example of such a scanner.

An example of non-contact scanning technique is LIDAR [5] (LIght Detection And Ranging, Fig.1). This scanner, installed on an airborne vector, is used to survey large areas. It is based on laser scanning from an aircraft, helicopter, drone or any other machine that is able to fly. The scanning gives the distance between the machine and a point on the ground. To get optimal results using LiDAR a combination of the scanner, GPS and an Internal Navigation System (INS) must be used to obtain a very precise point cloud.

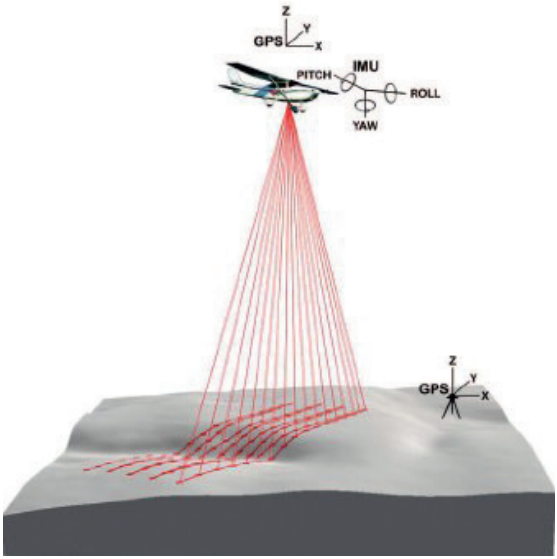

Fig. 1. Graphical visualization of the LiDAR airborne scanning method [6]

Another method for point cloud generation is photogrammetry (i.e. from the ReCap program) and requires very detailed georeferenced photographs of the terrain as an input. The photographs can be obtained by a photographer on the ground or, as in the laser scanning, by an airborne sensor platform. For this method to function correctly a specified overlapping among the photographs is needed.

Photogrammetry doesn't have to be used for terrain surveys. For example, in medicine, detailed and processed photographs of a body part can be used for the 3D printing of prosthesis.

Yet another method to be mentioned is based on structured light. Here a pattern of white lines of define sizes is projected on the scanned object (Fig. 2). From the computer analysis of the distortion of these lines a point cloud – and so the shape of the object – can be recovered.

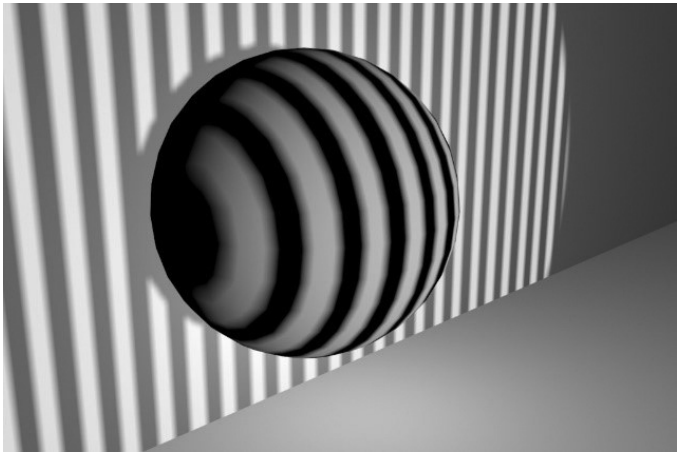

Fig. 2. Pattern of white lines projected on the scanned object

To get the whole 3D image of the object, it is enough to put it on the rotating plate and scan it from all sides.

The last method that we mention here is terrestrial 3D scanning [7].

This method works well for small areas like a room and it is not considered in the next subsection.

2.2. Viewshed generation from a big data set

As mentioned before, a viewshed cannot be generated directly from a point cloud. The intermediate step is to create a surface from a point cloud. This may be a difficult task if the number of the measured points is huge and the required computational power is not feasible. In the next part of the article, we give a solution to this problem and propose a method based on the generation of partial viewsheds and the way of connecting them.

In this article we present an example of a huge point cloud from which the generation of a viewshed was not possible. This example was a trigger to come up with a feasible solution for the surface and viewshed generation from huge point clouds. A viewshed from this huge point cloud was needed to another research project.

3. Input data

As input data we consider a point cloud of Krakow from 2012, within the project ISOK, Standard II – 12 points / m^2.

We present a set of assumptions, which were crucial for the choice of divisions of the example point cloud:

- The point cloud is in Standard II – 12 points / m^2
- The structure containing information about a single point occupies 48 bytes
- The whole point cloud file occupies 12 GB, giving around 270833333 points
- The available computers cannot cope with such a big point cloud

The choice of the point cloud was made according to the data size and the reality of the location, otherwise was random.

3.1. Division of the point cloud

Once a central point (a light source) in the point cloud has been decided, it is natural to start the division at that point. That is why we propose a division into radial slices with a centre in the light source. All of the slices have the same, fixed angle and only 'x' and 'y' coordinates are relevant for the assignment a point into a pie slice. Note that the central point is not necessarily in the centre of the point cloud. The position of the central point in this example was not important for us, it was requested in another research project. Differently, the choice of the division method was crucial due to the necessity of the light ray continuation. That is why the methods such as

division into square grid would result in errors and were rejected. This is because viewsheds for every single square would be generated separately and the big objects, such as buildings in the squares closest to the light source, would not shed the smaller objects in the farther squares. Now, instead of our division into pie slices according to the cylindrical coordinates, one could consider pieces obtained from the division of a sphere according to the spherical coordinates. This division would be similarly good, however more difficult to handle due to the varied number of points in the pieces and more complicated overlapping zones. Finally, one could shrink the number of points by consideration of mean values. As we were interested in a very detailed viewshed, the idea of mean values was rejected due to the approximation errors.

In our pie division method, the number of slices can be chosen arbitrary and depends on the size of the point cloud and the computational power available. We conducted many iterations in the search for the minimal number of slices.

Let's consider an example of division of the point cloud into pieces

- Division of the point cloud into 10 pieces (Fig. 3)
- Whole pie = 360°
- A piece of a pie = 360° / 10 = 36°

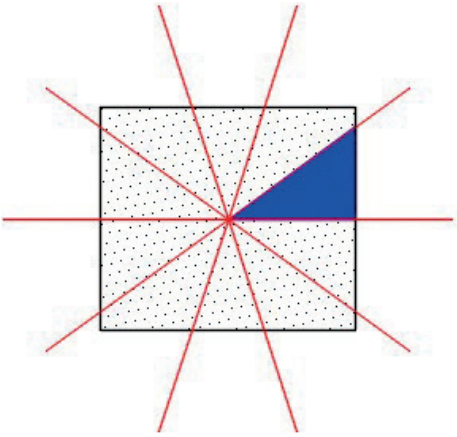

Fig. 3. Schematic representation of the division method. The square contains all the points of the considered point cloud. The division into pieces of pie (an example one is indicated in blue) is done from the central point

The resulting pieces contain a different number of points because the points in the point cloud are not uniformly distributed, the central point can be chosen arbitrary and the slices may have different areas. This has to be taken into account and accordingly the division would be called a success provided the generation of a surface from the biggest slice point cloud was possible.

In this sense neither the division into 10 pieces, nor in 30 led to success. Increasing the number of pieces, we found the first working division for 180 pieces, with a single slice of 2°. The surface generation time was related to the number of points contained in a single piece, where for bigger pieces it took way more time than for the smaller ones.

3.1.1. Point cloud division principle

A point cloud stores information about coordinates in Cartesian form (x,y,z). To divide a point cloud into slices of a pie (as described in the previous section), it is necessary to change the coordinates (x,y), into the polar form. As mentioned before the 'z' coordinate is irrelevant for the division. The change is performed according to the following formulas

$x=r\cdot\cos(\alpha)$

$y=r\cdot\sin(\alpha)$

$r=\sqrt{x^2+y^2}$

For $x>0$ $\alpha=\dfrac{arctg\left(\dfrac{\$y}{\$x}\right)\cdot 360^\circ}{(2\cdot\pi)}$

For $x<0$ and $y\geq 0$ $\alpha=\left(arctg\left(\dfrac{\$y}{\$x}\right)+\pi\right)\cdot\dfrac{360^\circ}{2\cdot\pi}$

For $x<0$ and $y<0$ $\alpha=\left(arctg\left(\dfrac{\$y}{\$x}\right)-\pi\right)\cdot\dfrac{360^\circ}{2\cdot\pi}$

For $x=0$ and $y>0$ $\alpha=\dfrac{\pi}{2}\cdot\dfrac{360^\circ}{2\cdot\pi}$

For $x=0$ and $y<0$ $\alpha=-\dfrac{\pi}{2}\cdot\dfrac{360^\circ}{2\cdot\pi}$

For $x=0$ and $y=0$ $\alpha=0$

where the arctangent is calculated in radians.

The subdivision of the points into slices of a pie was done according to the angle.

3.1.2. Implementation of the viewshed generation method into AutoCAD Civil 3D

After we have found the right division, we can focus on the viewshed generation from a smaller point cloud, namely our piece of a pie. This can be done in AutoCAD Civil 3D.

A viewshed generation can be performed in 7 steps (AutoCAD Civil 3D nomenclature is used here):

1) Set up a light source in a fixed position
2) Create a new empty TIN surface (the surface style must be renderable)
3) Import points from the point cloud to build the surface
4) Set up the orthogonal view from the top
5) Perform rendering, it will show the difference in illumination for visible and invisible zones
6) Save the file

The manual call of those functions for every single pie slice would take a lot of time. The solution is automatization of this process so that the program with steps 1-6 is applied in batch to all the slices of the pie. We perform the automatization writing a template in AutoCAD Civil 3D with a light source, correct TIN surface style and the orthogonal view.

The first step has to be performed manually by the user, namely one has to start program AutoCAD Civil 3D and open the template. Next, the steps 1:6 are performed automatically for all the pie pieces.

4. Results

Thanks to the division into 180 pieces, the generation of a viewshed for every single piece was possible. The next step was to connect the pieces and obtain the whole point cloud viewshed, which was done in Matlab. In Fig. 4, 5 and 6, we show one slice of the pie, a few connected pieces and the whole viewshed, respectively. The image shows a bridge over the Vistula River. The light source was placed in the centre of the image, around 1.5 m above the bridge so that it mimics the position of a human eye. The viewsheds are in grey scale, where the illuminated zones are brightened and the invisible ones are darkened. As our goal is to differ between visible and invisible zones, we transferred Fig. 6 into a binary scale, presented in Fig.7 with visible elements in white and invisible in black.

We encountered an issue while connecting the partial viewsheds to obtain the whole image. Every single viewshed was based on a triangular surface (TIN). Because of that there was always a piece of surface missing between two adjacent slices (Fig. 8). The solution was to increase the angle that a single viewshed covered adding 0.1° on both sides of the 2° slice (Fig. 9), to provide a partial overlapping between adjacent slices. The additional angle (0.1°) was chosen to be sufficiently big so that possible glitches on the overlapping surfaces were negligible. Note that, for the sake of clarity in the presentation, we show smaller number of points and division slices in Fig.8-10, than in our actual project presented in the paper.

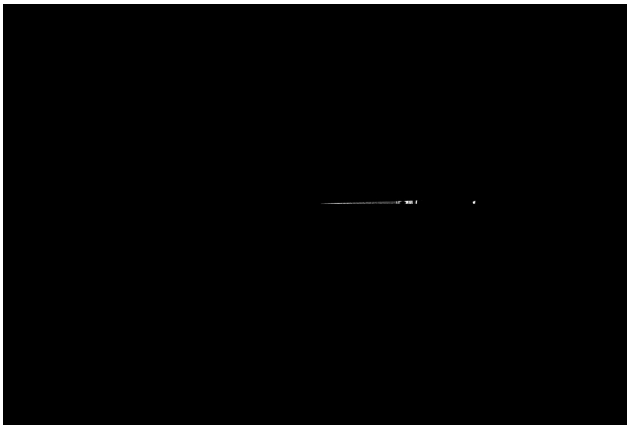

Fig. 4. The viewshed of a pie piece (2°) after rendering

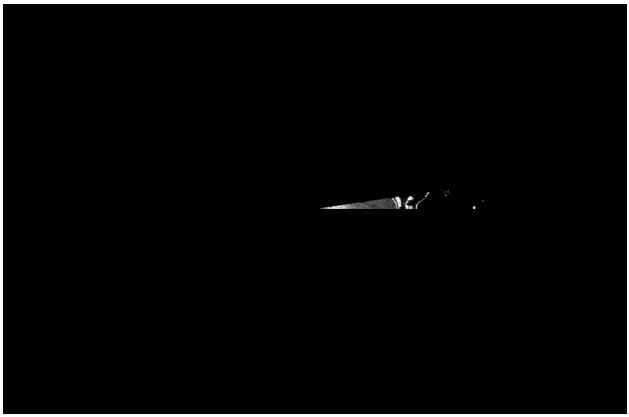

Fig. 5. he viewshed from a few connected rendered pieces

Fig. 6. The viewshed of the whole area, terrestrial view

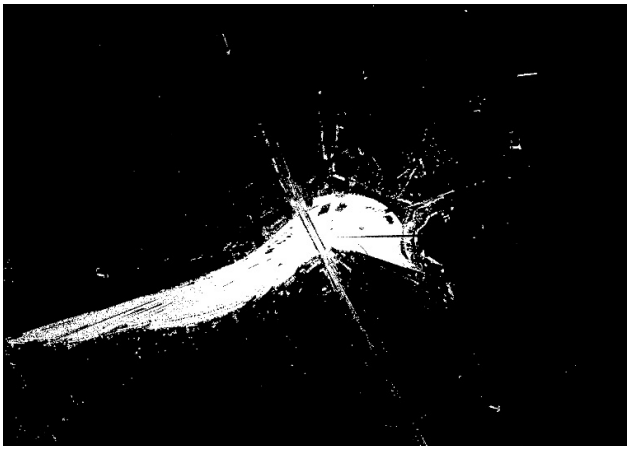

Fig. 7. The final binary viewshed

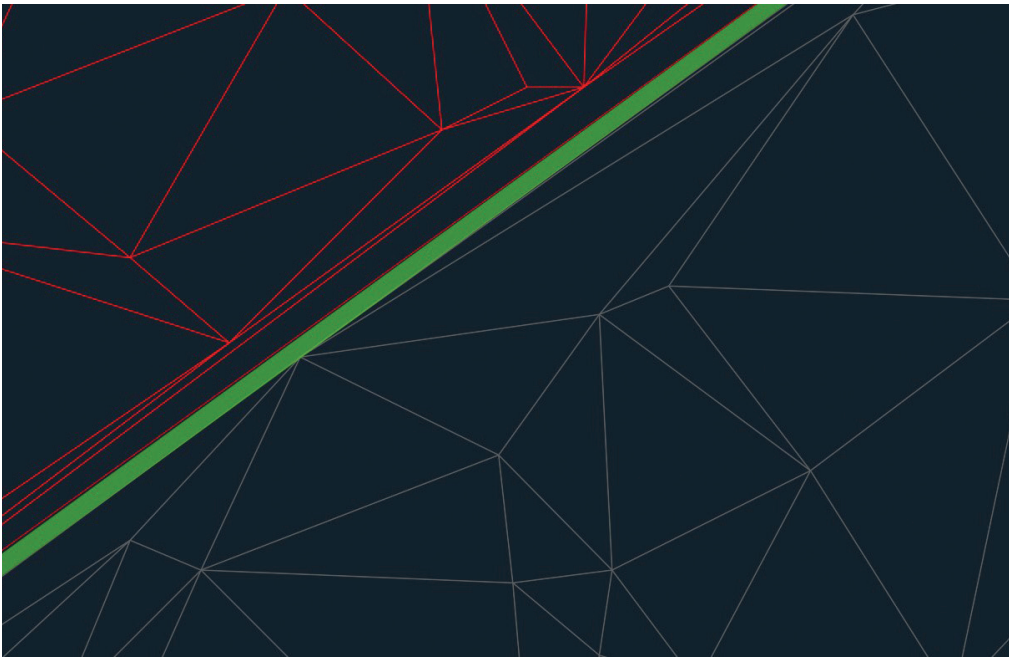

Fig. 8. The connection of two surfaces (in red and grey) with an empty space in the middle (in green)

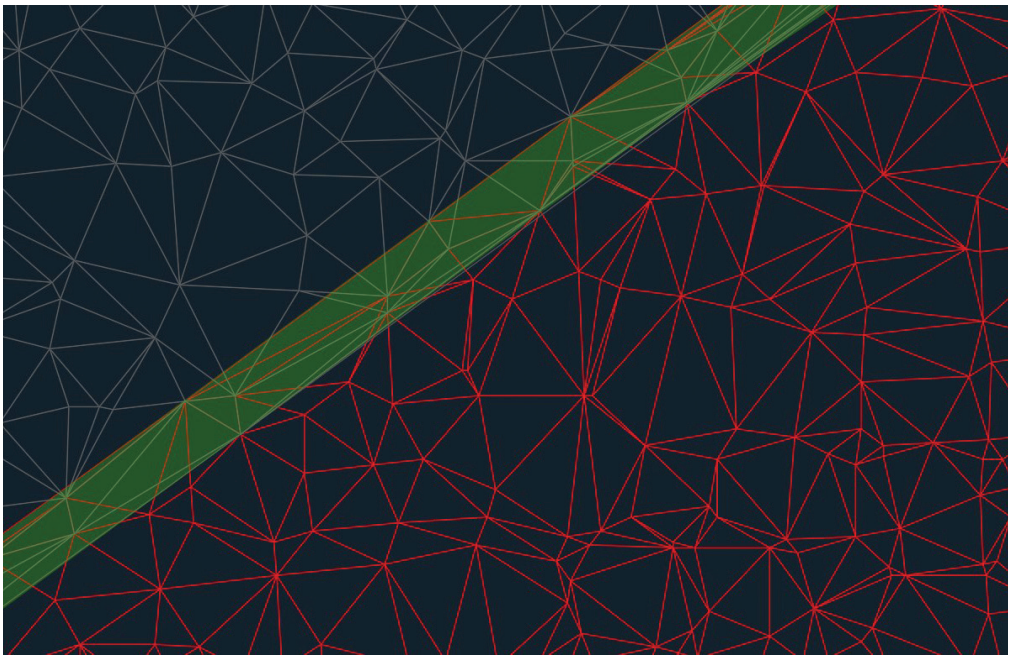

Fig. 9. Connection of two surfaces (in red and grey) The surfaces have overlapping piece of 0.2° (in green)

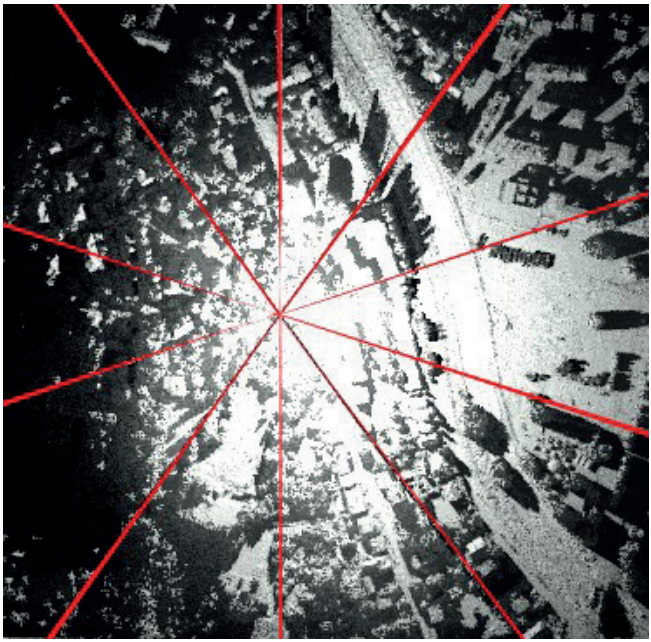

Fig. 10. The final viewshed with all the pie pieces connected. In red we indicate the overlapping part between the consecutive viewsheds

Another problem was a surface in the central point of the viewshed, where all the compounding surfaces should connect (Fig. 11). To deal with this problem, we generated an additional square surface covering the centre (Fig. 12–13).

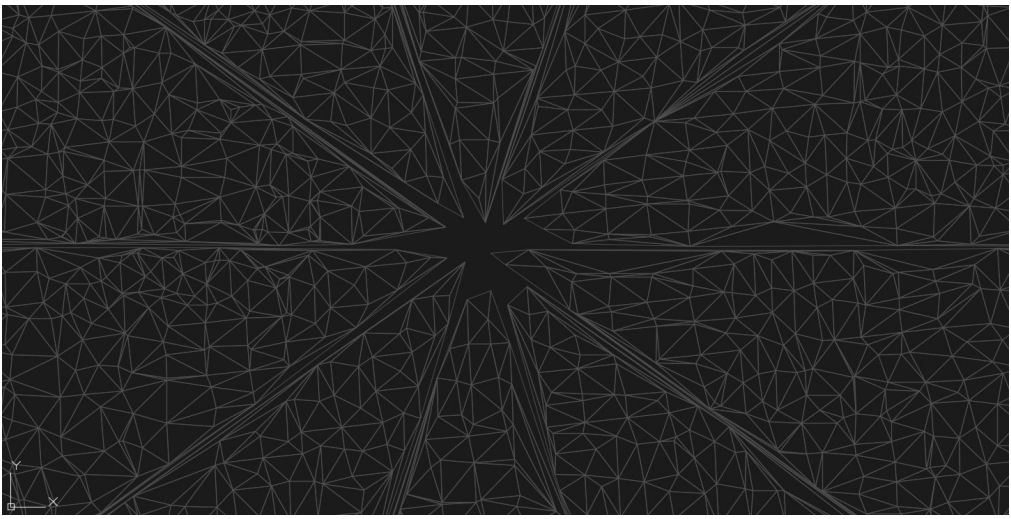

Fig. 11. Connection of all the compounding surfaces. Note the empty space in the centre and in between every two consecutive pieces of pie

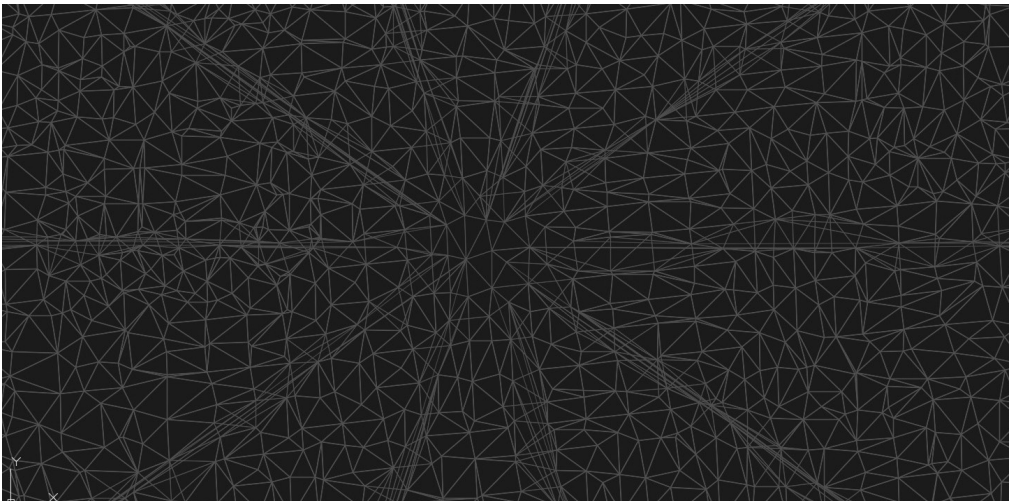

Fig. 12. Connection of all the compounding surfaces with the overlap). Note the overlapping part of the surface between the consecutive slices and additional surface in the middle

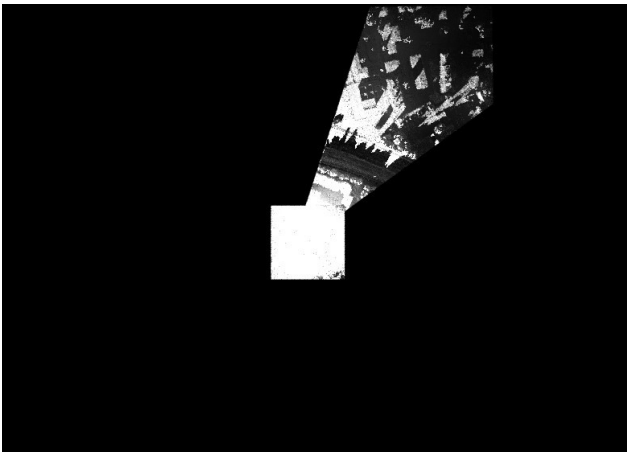

Fig. 13. Connection of between a single slice of a pie viewshed with the central square viewshed

5. Parallelization

In the future we plan to parallelize the programs for the division of the point cloud and for the generation of viewsheds in Autocad Civil 3D. It is possible to apply parallelization to divide a big point cloud into smaller ones because all the points are independent of each other. This independence of data is a necessary condition for parallelization. Thanks to the parallelization, a few points can be distributed into smaller point clouds at the same time. Similarly, the generation of viewsheds in Autocad Civil 3D can be performed with parallelization. The output viewshed from the first pie slice is not necessary to create the next ones. The viewsheds from different pieces of pie can be done simultaneously on different computer cores.

6. Summary

In this article, we presented a method for viewshed generation from a point cloud. For big point clouds, we proposed a method based on their subdivision according to pie slices starting from a central point. To apply this method, we considered an example of a point cloud, where we generated separate viewsheds for every single slice. Finally, while connecting the slices we encountered problems with empty space on the slice borders and near the central point, that we fixed using partial overlapping between slices. The generation of viewsheds was performed and automatized in AutoCAD Civil 3D. The next step is the parallelization of the division and viewsheds generation process.

References

[1] Warchoł A., *Gęstość chmury punktów pochodzącej z mobilnego skanowania laserowego /Density of point clouds in mobile laser scanning*, Archives of Photogrammetry, Cartography & Remote Sensing. 27, 149–161, Jan. 2015.

[2] Kim Y., Rana S., Wise S., *Exploring Multiple Viewshed Analysis Using Terrain Features and Optimisation Techniques*, Computers and Geosciences 2004, 30(9).

[3] Kajia K.T., *The rendering equation*, ACAM SIGGRAPH Computer Graphic Hompage Vol. 20, Issue4, 1986.

[4] Quan Y., Song J., Guo X., Miao Q., Yang., *Filtering LiDAR data based on adjacent triangle of triangulated irregular network*, Multimedia Tools & Applications, 8 May 2017.

[5] Ptak A., *LiDAR w wielkim mieście*, Geodeta: Magazyn Geoinformacyjny, May 8, 2017.

[6] Researchgate, https://www.researchgate.net/profile/Robert_Mcgaughey3/ publication/233622295/ figure/fig1/AS:299984089042947@1448533188764/ Figure-1-Schematic-showing-LIDAR-data-collection-over-bare-ground.png (access:

[7] 03.07.2015).

[8] Przywara J., *Skaner, czyli myśleć w 3D*, Geodeta: Magazyn Geoinformacyjny, May 8, 2017.