ARCHITECTURE CIVIL ENGINEERING ENVIRONMENT
The Silesian University of Technology No. 3/2025

doi: 10.2478/ACEE-2025-0041

REAL-TIME UPDATE OF DIGITAL TERRAIN MODEL FROM LiDAR DATA

Daniel JANOS [a*], Łukasz ORTYL [b] and Przemysław KURAS [c]

[a] MSc; AGH University of Krakow, Faculty of Geo-Data Science, Geodesy, and Environmental Engineering; al. Mickiewicza 30, 30-059 Krakow, Poland
ORCID: 0000-0002-5394-7922
*Corresponding author. E-mail address: *janos@agh.edu.pl*

[b] DSc; AGH University of Krakow, Faculty of Geo-Data Science, Geodesy, and Environmental Engineering; al. Mickiewicza 30, 30-059 Krakow, Poland
ORCID: 0000-0002-9151-7931

[c] PhD; AGH University of Krakow, Faculty of Geo-Data Science, Geodesy, and Environmental Engineering; al. Mickiewicza 30, 30-059 Krakow, Poland
ORCID: 0000-0001-5066-6220

Received: 21.03.2025; Revised: 25.08.2025, Accepted: 01.09.2025

Abstract
Construction work using heavy machinery (e.g. bulldozers, graders and excavators) will in the future require real-time digital terrain model (DTM) updating. The original contribution of this paper is the innovative method and the algorithm to implement this requirement. A portable measurement platform has been built as a fusion of GNSS (global navigation satellite system), IMU (inertial measurement unit) and LiDAR (light detection and ranging) sensors. The algorithm for operating the platform was developed in the Python programming language. Analyses of the capabilities and limitations of both the platform and the algorithm were performed. A field test was conducted at a gravel pit where the surface of the excavated material was modified in the course of the experiment to verify the performance of the system. It is concluded that the prepared measurement platform and the original algorithm are highly efficient in performing the real-time update of DTM model from LiDAR data.

Keywords: **2.5D point cloud; Digital terrain model; GNSS; IMU; Lidar; Python programming language.**

1. INTRODUCTION

1.1. State of the art

The development of measurement systems enables monitoring of changes occurring in the space typical for civil engineering with any level of detail. A special case is monitoring of the construction site, particularly in real time [1]. Current control of the status of resources allows for their effective management.

Real-time updating of digital terrain model (DTM) is a valuable feature on a construction site. DTM is a digital, three-dimensional representation of terrain (ground) surface and objects for a given location and area. It can be used to represent the shape of terrain surface, calculate the height H of any point (on its surface) where XY coordinates are known, and determine values derived from the shape. This is because a DTM can also provide information about inclination and other topographic (characteristic) features of terrain, such as edges or faults. Moreover, the possibility of using DTM in software for designing in the BIM (building information modelling) technology seems particularly interesting [2].

The technologies used to acquire and update DTM include: InSAR (interferometric synthetic aperture radar), aerial photogrammetry, UAVs (unmanned aerial vehicles) and LiDAR (light detection and ranging) [3-5]. An efficient way of acquiring point clouds for

DTM creation is provided by measurement platforms known as mobile mapping systems (MMS). In order to acquire point clouds and georeference them, a mobile vehicle is equipped with a number of devices such as: laser scanners, cameras, GNSS (global navigation satellite system) receivers, IMU (inertial measurement unit) sensors, odometers, and radars [6-8]. Their effective operation requires a multi-sensor fusion system to determine the position and orientation of a moving platform, based on Kalman filtering, among other things [9]. However, for obtaining data for machine control systems, data integration based on timestamps may be a sufficient solution [10,11].

Changes in the terrain surface caused by various factors require updating of DTM. Gagula et al. [12] provide an example of DTM for flood modeling. To update information about the object, they use a UAV positioned by RTK (real time kinematic) GNSS, supported by static GNSS measurement. In this case, the DTM was updated by UAV, in SfM (structure from motion) technology, not in real-time, not by LiDAR, but by photogrammetry. A similar study is presented by Ćwiąkała et al. [13], but it covers the use of UAV to acquire terrain deformations due to underground mining. As a result, they determine the accuracy of displacements in relation to the ground sample distance (GSD). Another solution is presented by Skaloud et al. [14], who deal with moving platforms for point cloud generation in real-time from ALS (airborne laser scanning) missions. They state that sub-decimeter accuracy is achievable and sufficient for many applications. Going further, Ernst et al. [15] present the use of a laser scanner to build a DTM of the surrounding environment to support the navigation of a multi-sensor system implemented using GNSS and IMU. The research found that laser scanning observations improve the results obtained by IMU+GNSS positioning. On the other hand, Koch & Lacroix [16] address the topic of managing environmental models within teams or robots. These devices are equipped with LiDAR sensors, providing point clouds that are used to incrementally build DTMs. Another example is given by Wang et al. [17], who present a fusion of LiDAR and IMU to develop terrain mapping for emergency rescue vehicles. In their work, they analyzed measurement errors and their impact on the accuracy of the vehicle localization, which amounted to 4–8 cm.

Many of the solutions listed are designed for the extemporary needs, and data acquisition and processing are time-consuming tasks. However, a large part of the needs for providing geometric information about the surrounding terrain requires real-time operation. These include, in particular, mobile robots operating mainly on the basis of RGB-D cameras [18], off-road all-terrain vehicles [19], and wheeled planetary rovers [20], in which the model is updated thanks to the motion of the measuring platform. Another example of a real-time change detection system is presented by Ding et al. [21], who study the deformations of tunnel segments based on multi-sensor data fusion.

Providing real-time terrain data is also crucial for the operation of earthworks machines. Furthermore, systems providing real-time data must take into account the operating conditions of the system, especially in dusty environments [22]. Devices such as UAV and TLS can provide high-quality data on the current condition of the terrain. Robertson [23] notes, however, that the ability to generate data on a working machine means that it can respond to such deviations in real time. This translates into higher quality data, which in turn facilitates the management of mine resources and improves decision making.

Terrain modeling in the field of earthworks is a much broader topic, especially in the field of construction machinery automation. Multi-sensor systems are developed for mining shovels [24], bulldozers [25] and excavators [26]. Chae et al. [27] present the results of work on the construction of an autonomous earthwork system, in which laser scanning was used. The scanner collecting data was placed on a moving platform, and the registration of scans was carried out based on targets. The goal was to create a model of the actual earthwork environment. In turn, Guan et al. [28] develop a system for terrain mapping and navigation for excavator operations. The system acquires a 3D point cloud from the LiDAR, an RGB data from the camera, and the orientation of the excavator acquired from GNSS RTK. The created system allows for automatic navigation of the excavator in a real environment, in which there are stockpiles, hills and deep pits.

1.2. Review of machine control systems in terms of DTM updates

Most control systems are based on a static DTM provided by the surveyor on site, which, however, is not continuously updated. In currently used machine control/guidance systems, terrain measurement can only be performed manually by the operator, using the machine's bucket/blade. At present, this feature is used mainly to create work surfaces for the system in the field, and for rough as-built measurements. It is provided by the main manufacturers of machine control systems, including Leica Geosystems AG [29], Trimble CE&C Help: Machine Control [30], Unicontrol ApS

[31], Makin AS [32], Novatron Oy [33].

Some of these also have the so-called surface logging. If the function is enabled, places currently being cut by the blade are mapped. The result of the mapping can then be used by the operator or manager in the office (via file synchronization) to assess the progress of the work. Such possibilities are offered by software providers: AEC Software Technology [34], Trimble Construction Support [35], Leica Geosystems AG [36], Unicontrol ApS [37], Topcon Positioning Systems [38], Makin' 3D Machine Control [39].

Manual logging of points by the user greatly limits the efficiency of DTM creation. Automated logging of points from the machine blade, on the other hand, does not provide fully accurate DTM results. This applies especially to areas where material is added. It is also not possible to perform initial mapping of the terrain (a pass before starting excavation, e.g. in order to create a precise cut/fill analysis) without physically modifying it.

The systems mentioned, even the advanced ones, which allow for semantic segmentation of objects occurring on the working site, are not designed with the current update of DTM in mind. An attempt to create a solution that focuses on probabilistic height determination was undertaken by Bhandari et al. [40]. In their solution, which allows for terrain mapping for mining shovels, 3D lidar sensors allow adding new observations to the existing terrain height map, taking into account false observations coming from, for example, dust on the measurement site. Their main achievement is the quick and effective update of dynamic changes to provide real-time terrain map. A similar solution, but much more extensively presented, is proposed by D'Adamo [41]. It uses a LiDAR sensor mounted on the machine and a set of RTK GNSS and IMU to position the moving platform. Among the many challenges indicated, the key is to provide real-time terrain maps based on a set of measuring devices, ultimately for the construction of fully autonomous bulldozers.

1.3. Motivation

In this article, we propose a solution that provides the controlled machine with a DTM that is adequately prepared and updated in real time. An increasing number of studies in the field of real-time DTM updating for earthworks [40, 41] indicates a large potential for the development of the proposed system. Compared to the existing solutions, we propose several novelties:

- Placing the system at the back of the construction machine, which allows for collecting data on the terrain after the drive and obtaining the current state of the terrain. This is important because after the bulldozer pass, the terrain cannot be expected to be perfectly flat due to the continuous track used in the vehicle propulsion. In addition, the current DTM is important for the planned optimization of the machine's movement shaping the terrain in accordance with the design.
- The measurement results are immediately oriented in the global coordinate system, and the nodes in the grid can have any values. The original measurement accuracy is also preserved by saving the exact XYZ coordinates of the last measurement in a specific cell. In the Bhandari solution [40], the height ranges (0.25 m) for a specific cell were predefined, leading to a decreased accuracy of the generated model.
- No need for initial data to be available. Unlike in the D'Adamo proposal [41], which requires a reference surface (acquired e.g. by a UAV), no initial data is needed except the approximate size of the project, which can change freely in the course of the work anyway.
- Improved LiDAR sensor performance (15 Hz instead of 10 Hz) while using a computer with a similar setup.
- Discussion on the height grid. In particular, we indicate the dependence of the height grid on the driving speed, taking into account the hardware limitations of the measuring system.
- A ready-made solution for the real-time DTM updating in the commonly used Python language. The presented approach involves planning and execution of both software and a physical implementation (prototype) for the problem described above.

In addition, it should be noted that in the experiment performed so far, D'Adamo [41] ultimately did not finish testing the algorithm with LiDAR due to sensor failure and used terrain prediction based on a model of the blade and tracks of the dozer. This data may be of limited accuracy due to the large number of variables.

By presenting the results of our research, we make a significant contribution to faster development of technology in construction by increasing the popularity of algorithms which update DEM in real-time. This is a significant deficiency that occurs even in professional, commercial machine control systems.

1.4. Article structure

The remainder of this paper is arranged as follows. The equipment used for the study is presented in Section 2, along with the measurement setup. An explanation of the algorithm for data acquisition is presented in Section 3. The subject of Section 4 is the results of the field verification of system operation. They form the basis for the discussion presented in Section 5, and the final conclusions are given in Section 6. Additionally, the key functions used for the calculations are included in Appendix A in the form of Python code.

2. MEASURING SYSTEM PROTOTYPE

In order to answer the research question, a technical design of a prototype measuring platform for a bulldozer, and the principle for the sensors fusion on the platform were developed. The measuring platform was built to test the algorithm, which was developed to update DTM in real-time. Finally, an analysis of input parameters for the algorithm to operate was carried out.

The proposed measurement system is to be mounted on construction machines such as bulldozers, excavators and loaders. The proposed arrangement of the main system components (GNSS receivers, IMU and LiDAR sensor) is shown in Fig. 1. The figure also shows the area measured by the LiDAR sensor Velodyne VLP-16, which was used for the testing. When projected on a vertical plane, it logs 16 beams with a total angle of ±15° with respect to the horizontal plane of the sensor (Fig. 2a). For a standard size dozer such as the Caterpillar D6, the LiDAR would be mounted at a height of about 3 m. After setting the appropriate tilt of the sensor relative to the ground/dozer plane (about 45°) and narrowing down the data set in the sensor's horizontal plane to ±35°, and distance measurement to 6 m, the obtained coverage of the measurement area just behind the machine will be entirely sufficient. This is because the whole set generates an area on the ground shaped roughly like a ring segment, the span of which is about 4.4 m on the inner side (just behind the machine), up to 6.8 m on the outer side (Fig. 2b), whereas the width of a standard dozer blade is around 3 m to 4 m. This provides adequate overlap to allow for a change of machine direction or material spillage outside the plough/bucket. These parameters are the starting point requirements for the testing of the measurement platform.

To facilitate testing of the proposed algorithms, a portable measurement platform has been built. The heart of the prototype measurement system is the previously mentioned Velodyne VLP-16 rotating LiDAR sensor. The measurement platform is complemented by two GNSS RTK receivers, an IMU, Windows 11 tablet, frame made of aluminium profiles, 3D FDM printed mounts and adapters, as well as minor accessories and equipment for efficient power supply and data transmission. The whole set is shown in the photograph below (Fig. 3). The only modification compared to the previously mentioned setup of the measurement platform is raising the LiDAR sensor measurement angle in the horizontal plane from ±35° to ±90°, in order to streamline manual measurements in the field by increasing the measurement area and, additionally, burdening the algorithm with more incoming data. Visualizations of the portable platform with the discussed increased measurement angle are shown in Fig. 4.

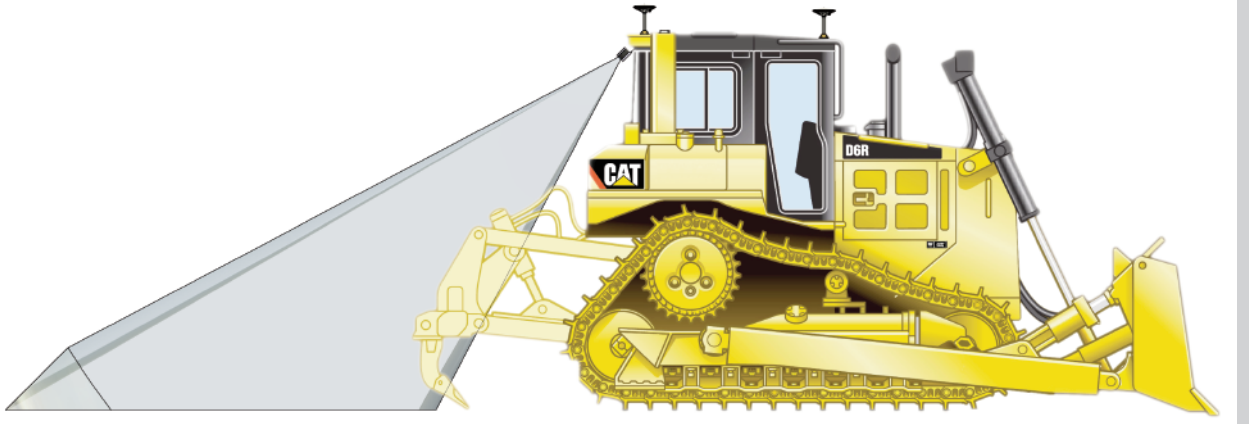


Figure 1.
Visualization of an illustrative CAT D6R dozer with a Velodyne VLP-16 LiDAR sensor and a measurement beam with the Hz angle of ±35°, V angle of ±15°, and measurement range of up to 6 m

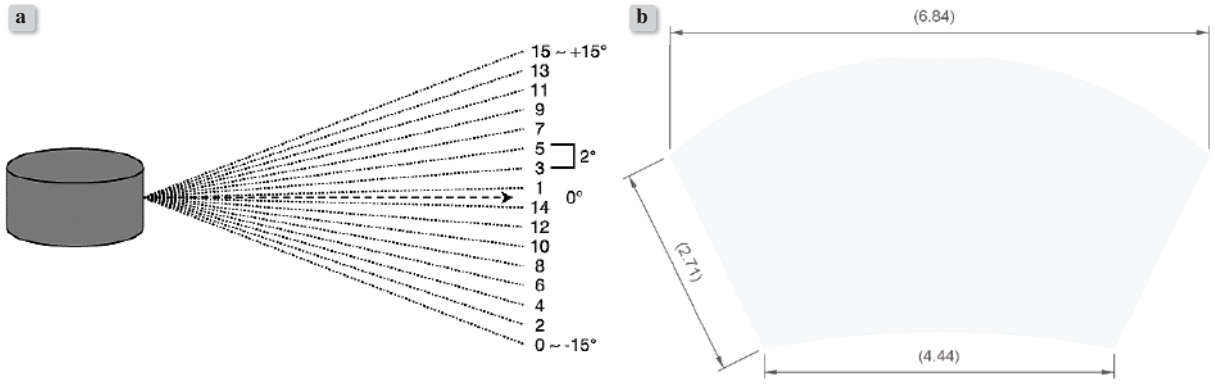


Figure 2.
Velodyne VLP-16 LiDAR parameters: a) measurement beams (channels) in the vertical plane [42], b) dimensions of the measurement area [m]

3. ALGORITHM DESCRIPTION

The following section presents in more detail the proposed algorithm for real-time DTM updating. It was used in the prototype of the measurement system described above (Fig. 3). The software of the measurement system prototype was designed with five key functionalities in mind:

A. definition of the work area and initialization of the measurement data collection space,

B. acquisition of measurement data to create a DTM using a laser scanner (LiDAR Velodyne VLP-16), two GNSS receivers and an IMU sensor,

C. filtration and georeferencing of measurement data in the global coordinate system,

D. collecting and updating the DTM with specified parameters in real time using a computing unit with average parameters (Microsoft Surface Pro 9 tablet with Windows 11),

E. performing additional calculations based on the DTM in parallel (e.g. volume of earth mass, data visualization, saving data copies to file) using the same computing unit.

The diagram of the program based on the developed algorithm is shown in Figure 5, where the functionalities described above are also indicated.

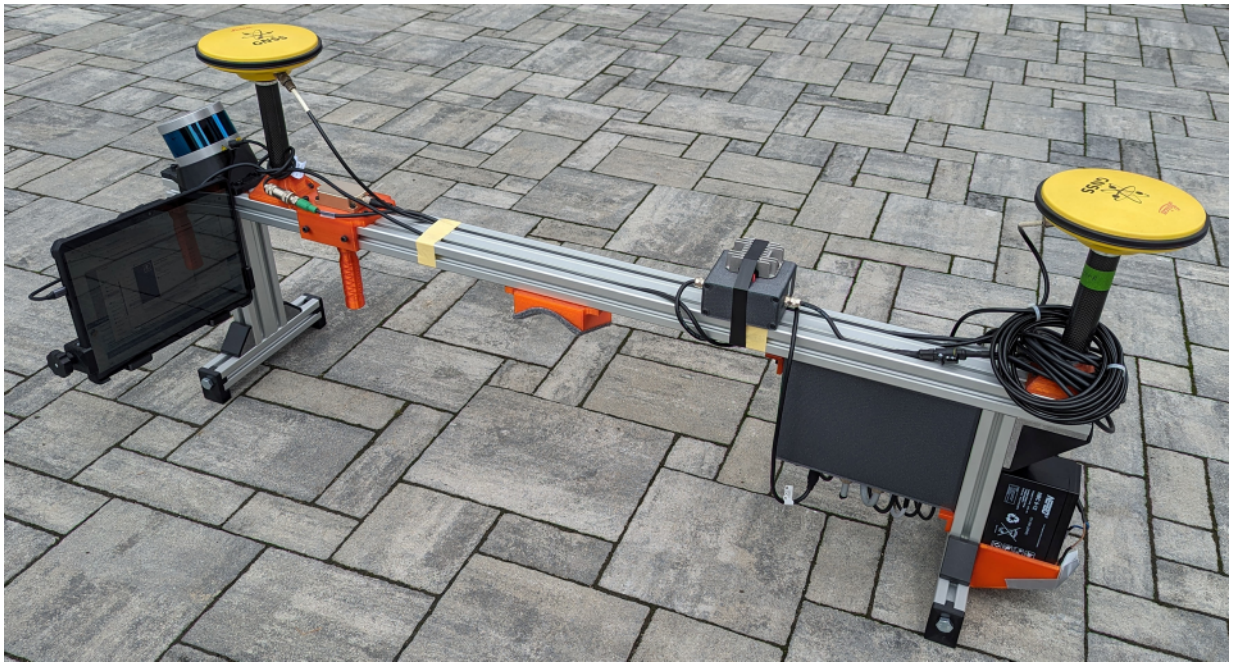


Figure 3.
Prototype of the measurement system

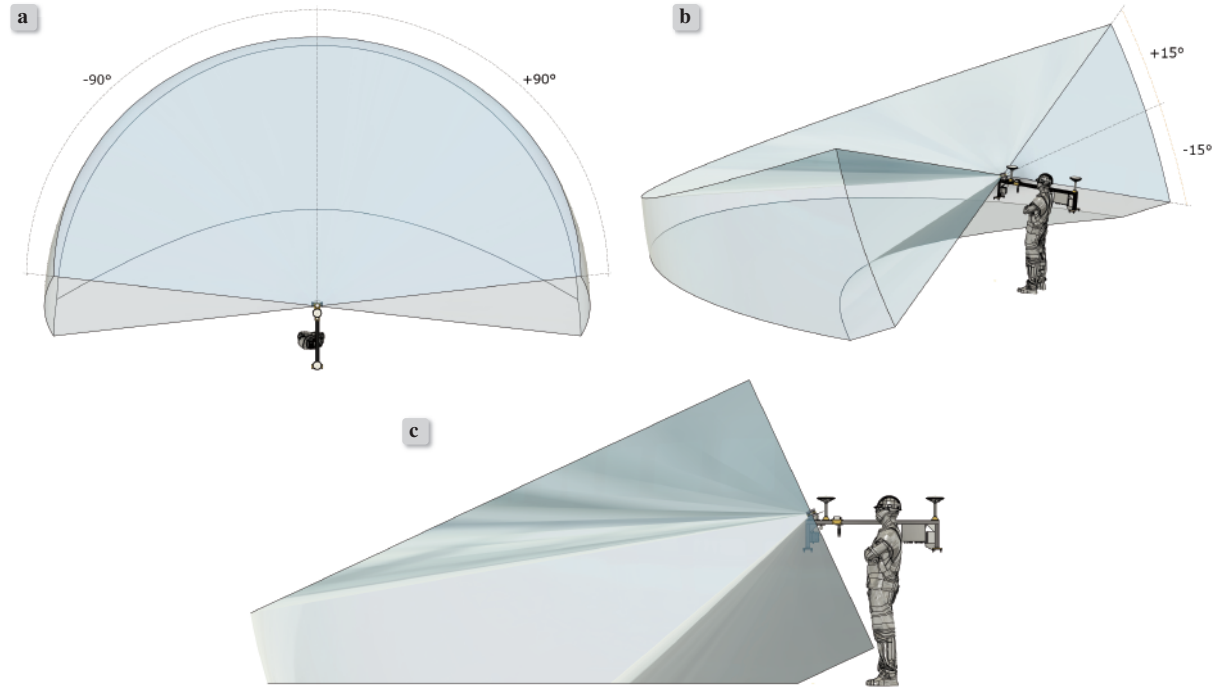


Fig. 4.
Visualization of the prototype with the LiDAR sensor measurement angle used in the tests: Hz ±90°, V ±15°, and measurement range up to 6 m: a) top view, b) spatial view, c) side view

Functionality D has become an important optimization issue from the perspective of the time efficiency. It should be noted that, due to the geometric arrangement, the acquired LiDAR data are characterized by a significant mutual irregularity of the measured points based on which the DTM is created. In a single LiDAR measurement, the distances between points are variable on both the horizontal and vertical plane. In addition, given the system (platform) movement factor, it is not possible to obtain mutually correlated data in terms of position for a quick and unambiguous update of the Z height value at the same points. Additionally, smooth operation of the entire system is crucial for the process of creating and updating a DTM in real time and performing derivative calculations, such as material volume (part E). Therefore, selection of a method of data representation has a very strong impact on the ultimate operation of the algorithm. According to the analysis performed by D'Adamo [41] in Chapter 2, reducing the collected LiDAR data to a regular grid is the optimal approach.

The work on our software has shown that pre-occupying space in the program's cache before starting the process significantly speeds up its subsequent iterations. Hence, before the start of LiDAR data recording and the DTM update process an empty data matrix is created (part A). It has a regular grid mesh size, and is expressed in the local coordinate system. Its length and width approximately cover the area for which the DTM will be updated (the software is designed to automatically increase the size of the data matrix in the event of exceeding the originally assumed measurement area).

The script performing the initialization of the data matrix is included in Appendix A as Code 1.

The elements of the data matrix A correspond to fragments of the terrain and are treated as "pixels" of a given size. The global coordinates *X, Y, Z* and other data describing a point physically measured in the terrain (LiDAR signal reflection intensity (*Intensity*), height difference to the design plane (H_{diff})) are recorded in subsequent "pixels". The size of rows, and cols of the data matrix is described by the following relationship:

$$rows = L * \frac{100}{px}, cols = W * \frac{100}{px}, \quad (1)$$

where:

rows – the number of grid nodes ('pixels') on the X axis (rows), rows $\in \mathbb{N}^+$,

cols – the number of grid nodes ('pixels') on the Y axis (rows), cols $\in \mathbb{N}^+$,

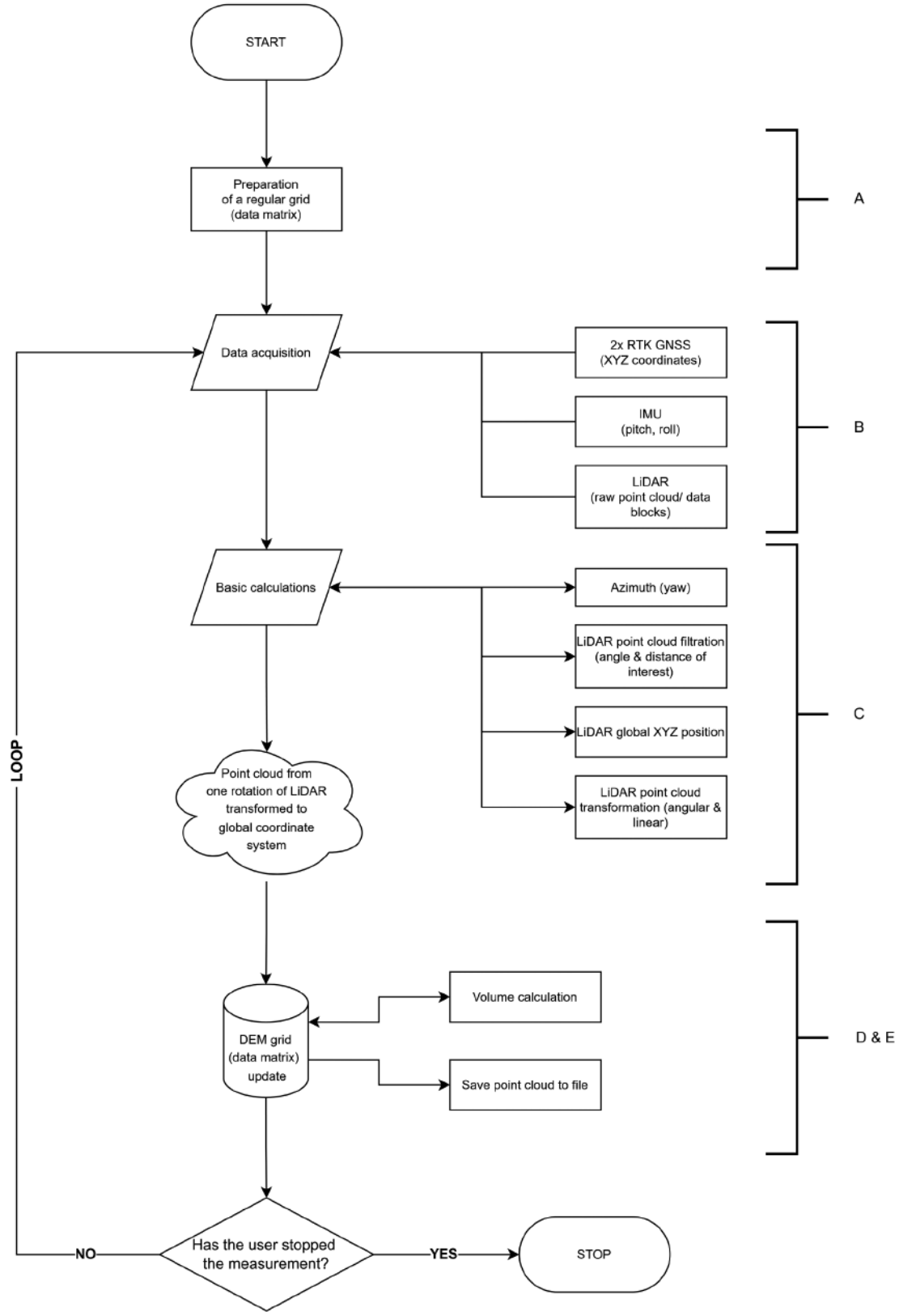


Figure 5.
Block diagram of the program based on the developed algorithm

L – the length of the DTM acquisition area on the X axis [m], $L \in \mathbb{N}^+$,

W – the length of the DTM acquisition area on the Y axis [m], $W \in \mathbb{N}^+$,

px – the size of a grid node ("pixel") [cm], a factor of 100, px {1,2,4,5,10,20,25,50,100}.

When data matrix $A_{i,j}$ ($i \in \{0 .. rows\text{-}1\}$, $j \in \{0 .. cols\text{-}1\}$) is initialized, all vector items located at the grid nodes ("pixels") are filled with zero values.

After the data storage space is prepared, the data acquisition and processing stage (part B) begins. The starting point for the measurement and computational part of the algorithm's work is the point cloud from a single rotation of the LiDAR sensor, limited to an angle in the horizontal plane indicated by the user. For our platform, it was the angle of ±90° from the azimuth axis (Hz) of the device frame, as mentioned in the previous section. The key element is the orientation of the acquired clouds from subsequent rotations of the LIDAR sensor in the global coordinate system. This is achieved by determining the current position of the measurement system from GNSS and IMU observations. Part B was executed, among other things, through the application of serial communication (GNSS and IMU sensors) and Ethernet (LiDAR sensor), as well as the "pyubx2", "pySerial" and "socket" libraries. Basic calculations (part C) were based on the "SciPy" library which calculates rotation using Euler angles [43].

In order to link the data matrix to global coordinates acquired through LiDAR (GNSS), translation parameters along the X and Y axes must be determined in order to obtain the minimum global coordinates X_{min}, Y_{min} of the DTM acquisition area (part C). This is done by recording the global coordinates of the LiDAR sensor at the time of starting data acquisition (initial position X_{init}, Y_{init}) and accounting for the size of the DTM acquisition area. The shift of coordinates measured by the LiDAR sensor relative to the data matrix is described by the following equation:

$$X_{min} = \left\lfloor X_{init} - \frac{L}{2} \right\rfloor, Y_{min} = \left\lfloor Y_{init} - \frac{W}{2} \right\rfloor, \quad (2)$$

where:

$\lfloor \cdot \rfloor$ – means the floor function.

The main loop of the developed program covers continuous collection and integration of data from all sensors. The result of its single iteration is a point cloud from one rotation of the LiDAR sensor, limited to the selected angle. The point cloud is unorganized and randomly distributed geometrically, but oriented in the global coordinate system. This means that one area can have an overrepresentation and another a deficiency of data relative to a grid mesh of the DTM created and updated in real time. For this reason, the application of the integral part of the real number (floor function) in equation (3) is a very desirable procedure and an original optimization approach in the process of determining the position of a data matrix node. Only one point of the cloud can be recorded in that node.

The node index [row_n, col_n] of the data matrix at which the values will be replaced with ones coming from the nth point of the cloud from the measurement cycle is determined based on the following relationship:

$$row_n = \lfloor (X_n - X_{min}) \rfloor * \frac{100}{px}, col_n = \\ = \lfloor (Y_n - Y_{min}) \rfloor * \frac{100}{px}, \quad (3)$$

where:

n – point number in the point cloud collected during the measurement cycle,

X_n, Y_n, Z_n – coordinates of the nth point of an acquired point cloud.

In a specific node, all subsequent points belonging to that node are continuously replaced from the point cloud collected in a measurement cycle, which means that the finally saved change is the last one. This method of operation is justified by the fact that the algorithm operates independently of the measurement parameters – a change of the mesh size does not require a change of the algorithm.

Since it takes the same time to check whether a mesh node has data assigned to it from a previous measurement as to overwrite that node with new data, multiple data updates for mesh nodes do not delay the program's operation. In addition, this approach always provides the most up-to-date data, regardless of whether it was obtained from subsequent LiDAR rotations at insignificant fraction-of-a-second intervals or after a significant change in the shape of the terrain due to earthworks.

The script executing the update of the main data matrix is presented in Appendix A as Code 2. In addition to exact coordinates of the measured point *X, Y, Z*, also recorded are LiDAR beam reflection Intensity and the height difference between the height of the point and the design plane H_{diff}, determined by the following formula:

$$H_{diff} = Z_{PC} - H_{proj}, \quad (4)$$

where:

H_{proj} – the height of the design plane (given by the user) at point X,Y.

A visualization of data matrix filling is shown in Figure 6.

If the calculated index of the data matrix node row_n or col_n exceeds the maximum size of *rows*, or *cols*, a new row or column is added, depending on in which direction there is no space. Additionally, where the data matrix needs to be shifted in the negative direction, such shift should be taken into account by also recalculating X_{min} and Y_{min} accordingly.

Having an always-up-to-date point cloud (in a DTM grid model), the volume of material (cut/fill parameters), among other things, can be continuously controlled. Height measurements can be referenced to a "0" level, a fixed horizontal plane, an inclined plane or a 3D model:

- In the first case, volume calculation comes down to summing up the height column and multiplying it by the grid pixel area:

$$V = \sum_{i=0}^{rows-1} \sum_{j=0}^{cols-1} Z_{i,j} * \frac{px^2}{10000}, \qquad (5)$$

where:

$Z_{i,j}$ – point height at node *i, j* of data matrix A.

- In the second case (implemented in the presented prototype system), an extra column has been added to the data matrix, containing the differences between the measured height and the constant height of the design plane (H_{diff}), calculated at the time of swapping the X, Y, Z data for a given grid cell. Then, to obtain the volume, the column with these differences should be summed up and multiplied by the grid mesh size:

$$V = \sum_{i=0}^{rows-1} \sum_{j=0}^{cols-1} H_{diff\ i,j} * \frac{px^2}{10000}, \qquad (6)$$

where:

$H_{diff_{i,j}}$ – height difference at node *i, j* of data matrix A.

The script executing the volume calculation for this case is shown in Appendix A as Code 3.

- The most advanced case (inclined plane or 3D model) involves adding 2 columns to the data matrix. First (auxiliary) column contains target heights for each grid node, calculated before starting the measurements. Second column contains differences between the measured and target values. Then, as in the previous case (Eq. 5), these differences should be simply summed up and multiplied by the grid mesh size to obtain the volume.

Creating an initial empty matrix for measurement data has its advantages, but to save the current DTM to a file, it is necessary to process it first – by removing grid cells with zero values (where the terrain was not measured). The script that executes this task is included in Appendix A as Code 4.

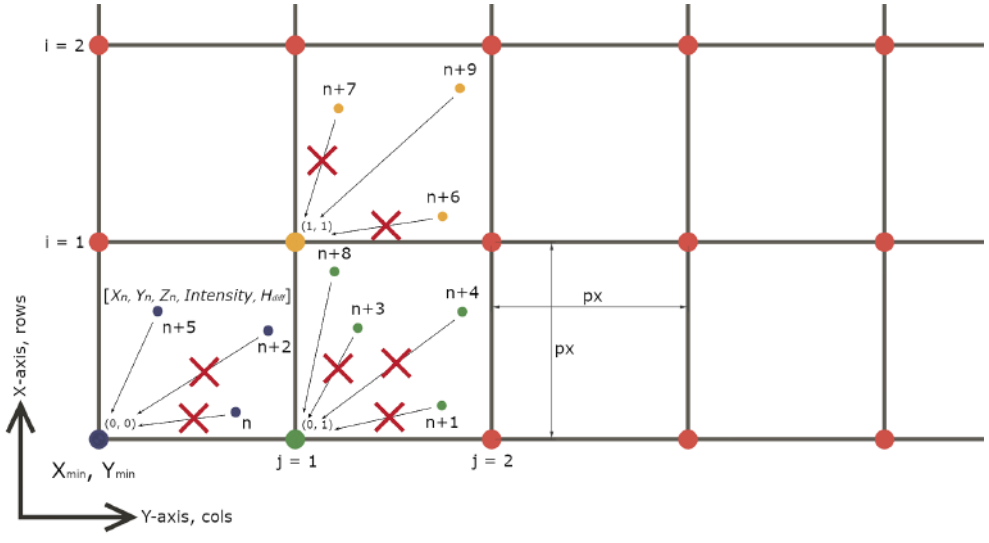


Figure 6.
Visualization of the data matrix – a grid with a single pixel size of px. Each pixel, starting in the lower left corner, can only contain one point with data [X,Y,Z, Intensity, H_{diff}]. The colors blue, green and yellow indicate the corresponding grid nodes (pixels) and measured points

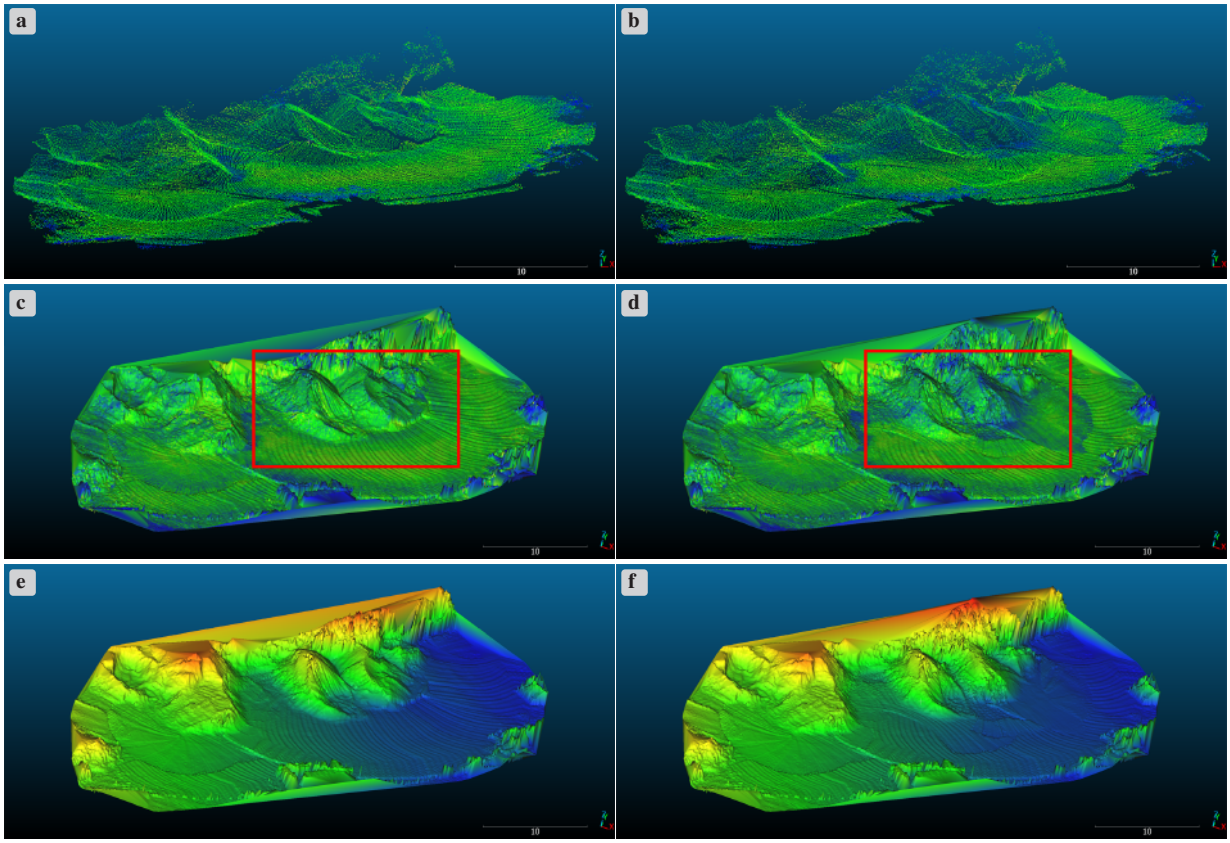


Figure 7.
Point cloud (a), b)) and mesh model (c), d), e), f)) from the conducted test measurement: before (a), c), e)) and after (b), d) f)) the pile was modified with an excavator. The colours for graphics a), b), c) and d) are the LiDAR laser beam intensity, while for graphics e) and f) it is the height map. The area of changes has been marked with a red rectangle

The described approach to the issue of real-time DTM update is crucial because it reduces operations performed by the computational unit. Coordinate values are not compared iteratively, but are immediately and efficiently assigned to a given mesh. It is an important contribution to the problem of real-time update of DTM in terms of a practical application of the prototype in the future, e.g. as a fully functional component of machine control systems (e.g. for motion optimization).

4. RESULTS – FIELD VERIFICATION OF SYSTEM OPERATION

In order to verify the operation of the algorithm (continuous measuring and updating of the grid DTM model), field measurements have been performed. They included measuring a section of a small gravel mine – a maneuvering area and small gravel piles, pushing the material onto the piles with an excavator and repeating the measurement, but only along the modified piles. The measurement conditions were good – a sunny day, free of dust. Measurement object – mainly gravel, sand and dry soil, slightly moist in the place of modification. The screenshots (Fig. 7) show the results (point cloud straight from the measurement prototype, not modified or filtered in any way, and a mesh model) before and after pushing the material with the excavator. While the material was being pushed in with the excavator, program operation was not interrupted, and the clouds shown here were saved to a file while the program was running.

In order to lend credibility to the tests and the performance of the presented algorithm, screenshots from the program application are shown in Fig. 8, providing the following information:

- the number of LiDAR data packets before pile modification and just before the measurements were completely finished,
- the time of taking measurements/screenshots (both from GPS and Windows),

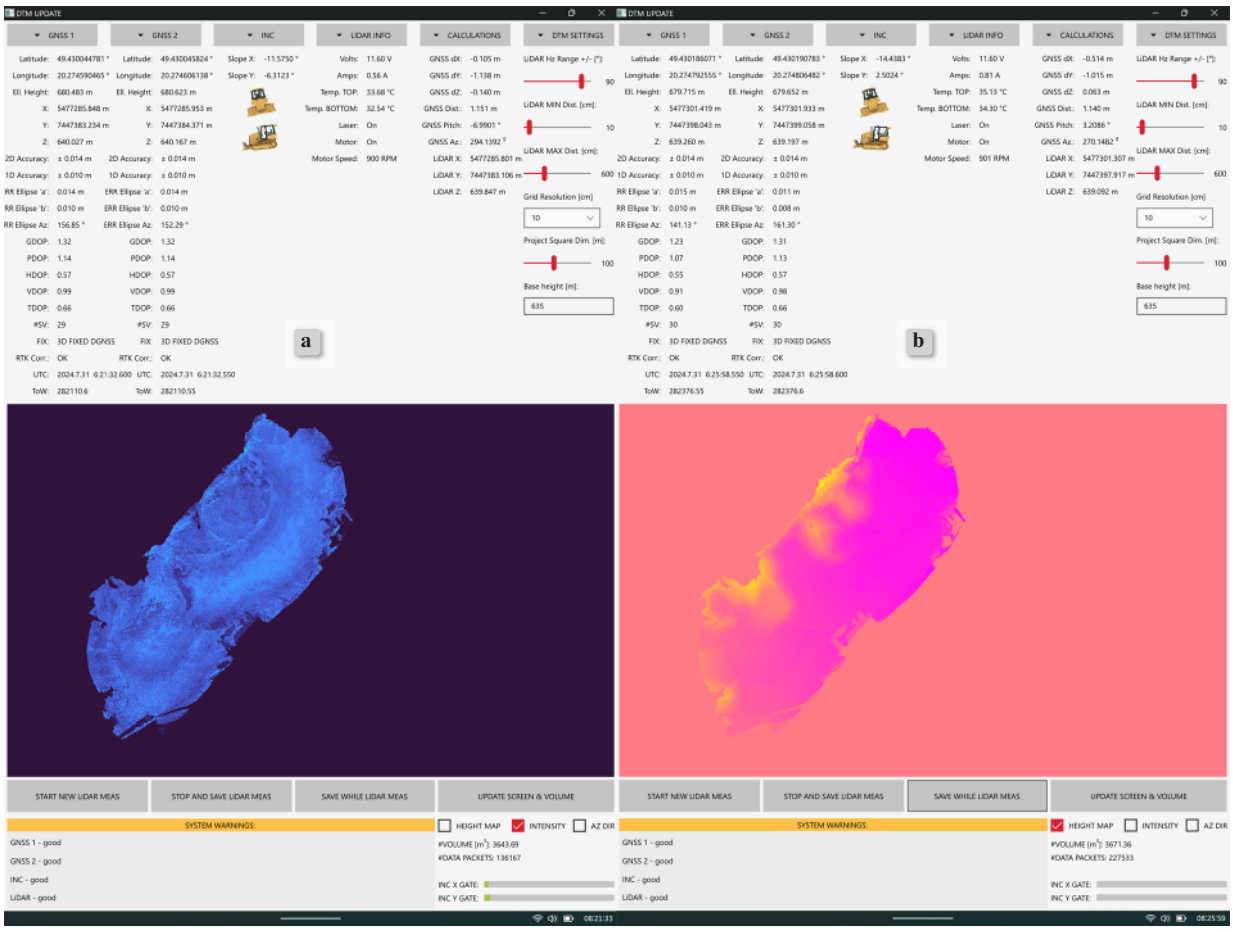


Figure 8.
Screenshots from the program's field application. The image a) shows a view of the cloud with LiDAR beam reflectance intensity colors, before the pile was modified. The image b) shows a view of the cloud where the colors of the points are determined by their height, after the pile was modified and the changes captured, just before the program stopped running

- the volume calculated relative to the base plane. Any differences may be due to difficulties in measuring the top of the pile and measurement inaccuracies of the system (the prototype is in the initial stage of development).

This is complemented by a video showing the measurement process, provided in Appendix B. A photos taken before and after the measurements were completed is shown in Fig. 9.

5. DISCUSSION

The test was conducted to verify the performance of the proposed real-time DTM update algorithm. The results submitted (Figs. 7-8) suggest that it operates correctly, as points at newly measured locations are continuously replaced with new, up-to-date ones. The successively saved point clouds show the modifications made to the pile and traces on the ground around it. As the measurements were conducted, with the specified measurement parameters (resolution 10 cm, project size 100 x 100 m), no issues with insufficient computing power of the tablet were felt – data packets were received and processed smoothly.

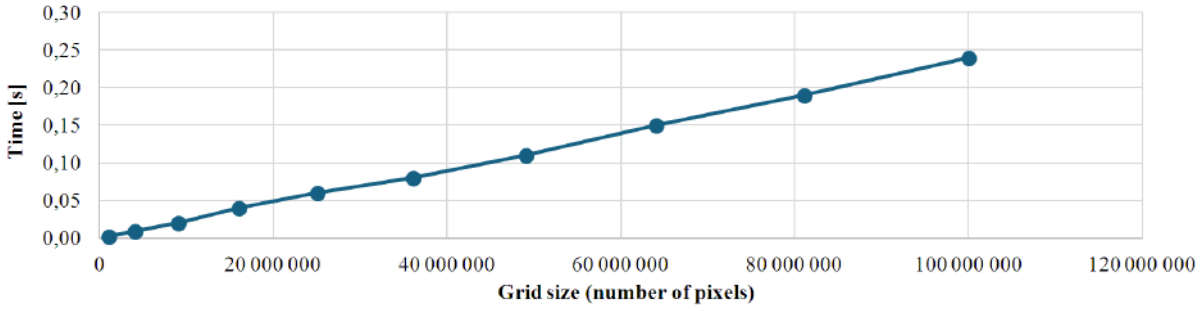
For the purpose of testing the performance of the algorithm, a Microsoft Surface Pro 9 tablet with an Intel i5-1235U processor clocked at 2.50 GHz was used for field work. For the abovementioned measurement parameters, a single iteration of the main loop, corresponding to one rotation of the LiDAR sensor and the update of main data matrix, takes less than 0.001 seconds (Appendix A, Code 2). The computational equipment used is fast enough for the presented prototype to operate and be developed. The time for volume calculation (Appendix A, Code 3) for the mentioned measurement parameters is about 0.002 seconds. The developed algorithm is not without its limitations, however. The first one of them is

Figure 9.
Photos before (a) and after (b) the pile modification. Video from field tests is included in Appendix B

Table 1.
Maximum project dimensions depending on the selected resolution of DTM grid

Grid resolution (pixel size) [m]	Max project area [m^2]	Example dimensions [m]
0.01	90,000	300 x 300
0.02	360,000	600 x 600
0.04	1,440,000	1200 x 1200
0.05	2,250,000	1500 x 1500
0.10	9,000,000	3000 x 3000
0.20	36,000,000	6000 x 3000
0.25	56,250,000	7500 x 3000
0.50	225,000,000	15000 x 3000
1	900,000,000	30000 x 3000

Table 2.
Relationship between the grid DTM volume calculation time and the data matrix size



the size of the NumPy matrix storing the main data matrix. Its size depends mainly on the computer parameters, but also the type of data that the matrix collects. For the presented Microsoft Surface Pro 9 tablet with 16 GB RAM, the maximum size of the data matrix (storing the X, Y, Z coordinates, the intensity of the LiDAR beam reflection and the height difference relative to the base plane for each grid point) must be below 900,000,000 "pixels". Their number is affected by the preset grid resolution and the size of the project area. Table 1 shows the relationship between grid model resolution and the project size, giving the real picture of the program's limitations. This limitation can be minimized by using a different computer (one with more RAM), data type (e.g. float32), or programming language (compiled, e.g. C++).

The time needed for volume calculations is another limitation. Calculation time gets higher in proportion to the size of the data matrix, because of the summing up of values in the height difference column (Table 2.). The burden of long calculations for a large data matrix can be reduced by only updating the volume at certain intervals, e.g. every second or every few seconds. It is also possible to change the approach to these calculations by updating the volume at the moment of replacing a given point in the data matrix, without summing up the entire height

difference column. This solution will be considered in the future.

In addition to restricting the maximum size of the project, use of the finest grid resolutions (higher than 5 cm) may also create issues with continuous filling up of all grid nodes. Emerging gaps will result in incomplete data matrix updates due to the lack of currently measured points from the LiDAR sensor at a given location. This is because the sensor can collect a limited amount of data in a given unit of time (~300,000 pts/s for the full 360° horizontal angle and vertical angle of ±15°, in single return mode [44]). The problem can be solved e.g. by using linear interpolation which, however, is not yet being implemented (mainly because a higher resolution is not needed for earthworks). Therefore, in order to make sure the grid cells are filled in correctly, the speed of machine movement must be adjusted (limited) appropriately. Accordingly, analysis has been carried out to show how the LiDAR rotation speed and the movement speed affect the resolution of measurements that can ultimately be obtained:

- Table 3 shows the relationship of the distance between successive points of the same channel/beam (azimuthal resolution) in the horizontal plane at the tested distance of 6 m and the LiDAR rotational speed – revolutions per minute (RPM). As can be seen, the resolution is very high, considering it is to be applied in construction works. It is therefore possible to select even the highest rotational speed, as ultimately that would not have a very negative impact on the azimuthal resolution, while it might have a positive impact on the radial resolution. Consequently, it has been decided to increase the standard rotation speed of the LiDAR from 600 RPM to 900 RPM. A decision was taken not to use the maximum rotational speed of 1200 RPM, so as to avoid overloading sensor bearings, and thus raising the temperature of the sensor.
- Figure 10 shows distances on the ground between points from all the 16 LiDAR channels, with the sensor mounted as specified. As can be seen, these distances are significant. As the platform moves, the points from successive measurements will

Table 3.
Dependence of the azimuthal resolution at a distance of 6 m on the rotational LiDAR speed

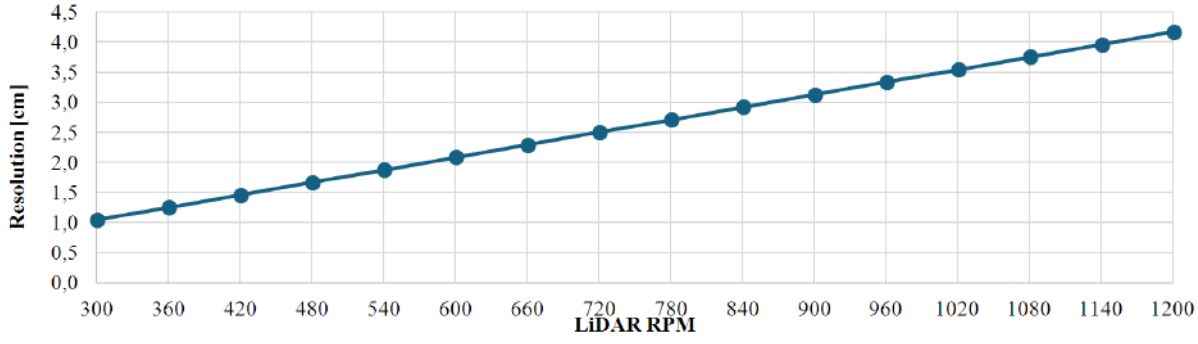
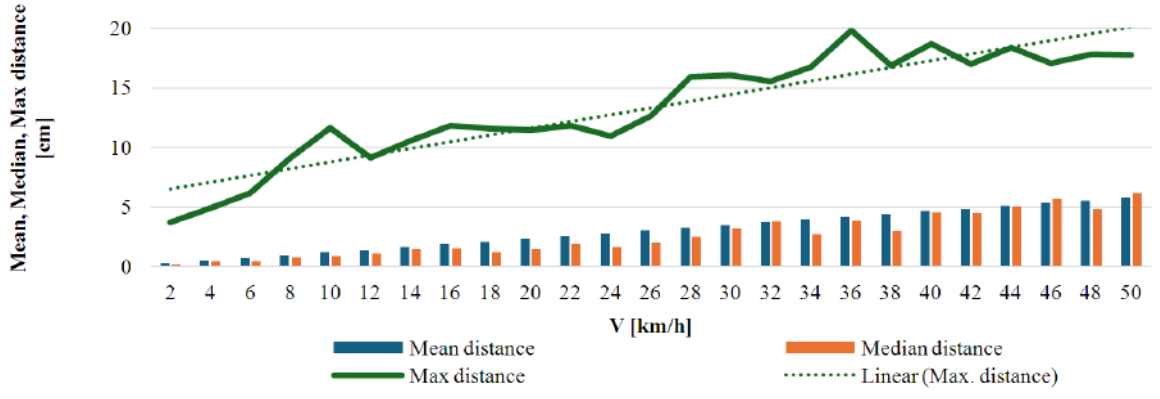


Table 4.
Dependence of the moving speed (assuming the rotational speed of the LiDAR sensor is 900 RPM) on terrain coverage with measurements



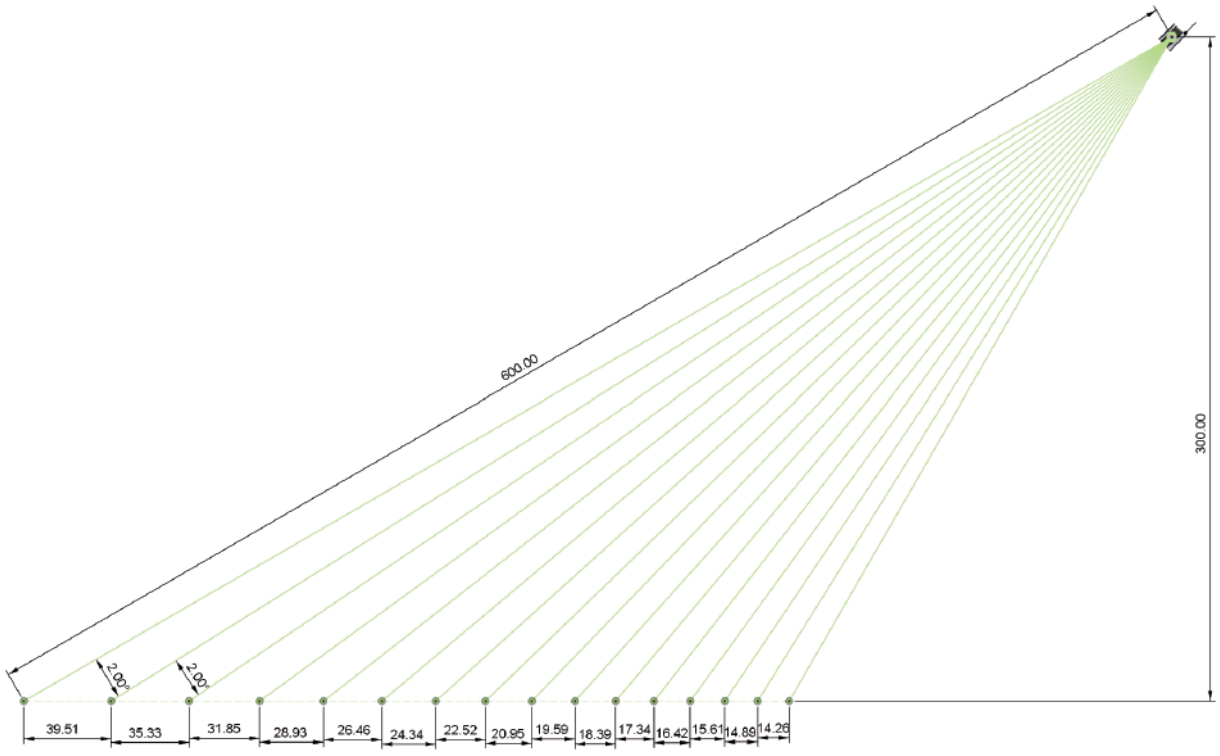


Figure 10.
Distances [cm] on the ground between points from all the 16 LiDAR channels, with the sensor mounted as specified: 3 m above the ground, at an angle of 45°, maximum distance 6 m

begin to overlap. Therefore, the moving speed determines how the measurement area will be covered with successive points.

- Table 4 shows the dependence of the moving speed (assuming the rotational speed of the LiDAR sensor is 900 RPM) on terrain coverage with measurements. The first thing that can be noticed is the irregularity of the maximum observed gap that can occur in the grid that is being logged. It is mostly determined by the random factor, because in field conditions it is very difficult to achieve a perfectly constant moving speed and a smooth terrain surface. It can therefore be assumed that the fitted trend line correctly answers the question of what moving speed corresponds to the maximum possible grid resolution. At low speeds, many points practically overlap with each other, hence the fitted line is not significantly inclined and the resolution does not decrease proportionally to the increase in moving speed.

When it comes to the design of the platform itself and the sensors within it, certain limitations may arise. Primarily, GNSS signal reception can be problematic when operating in dense urban areas, among trees, or under structures. Mounting the platform on the machine's roof and operating outdoors can virtually eliminate this problem. In the case of a LiDAR sensor, reflections from mirrored and semi-transparent surfaces can be problematic. However, construction sites primarily consist of soil and various types of bulk materials, which should not pose a challenge for this type of sensor. The only real obstacle may be dust floating behind the machine during operation. This problem can be partially eliminated by using appropriate filtration, for example, based on the intensity of the laser beam's reflection and proximity to other nearby measured points, as described in [45]. At the time of publication, the measurement platform is not designed for operation in harsh field conditions, except for data recording interruption in the absence of a FIXED position solution for both GNSS receivers. More advanced solutions are planned for future implementation.

6. CONCLUSIONS

The work carried out and the analysis of the collected test data allowed for the evaluation of the performance of the proposed measurement platform and processing algorithm. For the purposes specified at the beginning (measurements on construction sites and machinery control systems), the use of resolutions ranging from 10 to 25 cm seems most reasonable for several reasons:

- Work can be performed on a larger project area.
- No issues with filling in the DTM grid meshes, even at high speeds of the measuring platform.
- In any case, the specified resolution is several times higher than one a surveyor would ever use to measure the terrain using a total station or GNSS receiver and, as verified by the studies [46], increasing the resolution does not significantly affect the result of volume calculations.
- Ongoing volume calculations would not pose any problems due to the large size of data matrix.
- Working with a bulldozer, grader or excavator involves producing flat areas with large surfaces, so resolution based on single centimetres is normally not needed.

For the purpose of acquiring data from the level of earthworks machines, modern low-cost LiDAR scanners provide sufficient speed and resolution. Despite the small number of channels/scanning beams, these sensors can be used successfully, because the appropriate resolution can be obtained thanks to the movement of the machine and the overlap of subsequent beams on the mapped area. However, the measurement system settings should take into account the machine's speed, especially when high resolution of the acquired point cloud is required. In addition, the sensor's rotational speed should be taken into account to avoid hardware overload.

The algorithm was shown to be correct and effective using a mid-range processor to process real-time measurement data. The presented system and algorithm are efficient enough for typical DTM update needs for earthworks, despite the fact that a low-level programming language was not used.

Appendix A. Algorithm code in Python language

Code 1

The following code creates the empty matrix (containing only zeros) with parameters specified by the user. Additional clarifications on what is executed by a given line are provided as the comment.

```
import numpy as np

# USER SETTINGS:
project_dimensions = (100, 100)  # [m] dimensions of whole project grid data matrix
pixel_size = 10  # [cm] dimension of the smallest grid pixel; may be any divisor of 100 (1,2,4,5,10,20,25,50,100)
project_height = 635  # [m] the plane height to which volume measurements will be referred

# the DTM grid is treated like an RGBA image
pixel_area = (pixel_size / 100) ** 2  # [m2] grid smallest pixel area; needed for volume calculations
pixels_per_1m = int(100 / pixel_size)  # [px / m] number of pixels in 1m row (column)
print(f'grid pixels per 1 m2: {pixels_per_1m}x{pixels_per_1m} = {pixels_per_1m ** 2}')

matrix_height = project_dimensions[0] * pixels_per_1m  # [px]
matrix_width = project_dimensions[1] * pixels_per_1m  # [px]
matrix_size = (matrix_height, matrix_width)  # [px] x [px]
print(f'grid image resolution: {matrix_height}x{matrix_width} = {matrix_height * matrix_width}')

grid_data_matrix = np.zeros((*matrix_size, 5), dtype=np.float64)  # numpy 3D matrix; grid with data: [X, Y, Z,
Intensity, Height Difference]
grid_data_matrix.fill(0)  # fill the matrix actually with zeros in memory

# lidar_x, lidar_y -> global LiDAR coordinates at the time of starting the measurement (grid center coordinates)
xmin = int(lidar_x - project_dimensions[0] / 2)  # minimum global coordinates of the grid
ymin = int(lidar_y - project_dimensions[1] / 2)
```

Code 2

The source code that executes the update (replacement) of data in the main data matrix is shown below. To achieve adequate computational time in the Python language, the Numba JIT compiler was used. Once the terrain point data matrix is updated, the program's main measurement loop starts over.

```
import numba

# DTM UPDATE FUNCTION
@numba.njit("float64[:, :, :](float64[:, :, :], int64, int64, float64[:, :], int64, float64)")  # output and input
data types for Numba JIT decorator
def grid_update(grid_data_matrix, xmin, ymin, one_rotation_point_cloud, pixels_per_1m, project_height):
    for xyz_i in one_rotation_point_cloud:  # loops through all points data from LiDAR
        x_img = int((xyz_i[0] - xmin) * pixels_per_1m)  # calculates X and Y image coordinates
        y_img = int((xyz_i[1] - ymin) * pixels_per_1m)
        hdiff = xyz_i[2] - project_height  # calculates difference to project height
        xyz_i_dh = np.append(xyz_i, hdiff)  # joins LiDAR point data with calculated height diff
        grid_data_matrix[x_img, y_img] = xyz_i_dh  # updates data in the given grid point
    return grid_data_matrix  # returns updated grid_data_matrix

# one_rotation_point_cloud -> data from one rotation of LiDAR in the form of matrix of [X, Y, Z, Intensity] data

grid_data_matrix = grid_update(grid_data_matrix, xmin, ymin, one_rotation_point_cloud, pixels_per_1m, project_height)
# DTM UPDATE
```

Code 3

A simple function that sums up the height differences in the data matrix (H_{diff}; index 4 denotes column no. 5) and multiplies that sum by the area of one grid cell is presented below. It returns the volume of the terrain that was measured (for which the grid pixels have non-zero values).

```
volume = np.sum(grid_data_matrix[:, :, 4]) * pixel_area  # measured DTM grid volume
```

Code 4

The source code shown below is used for saving the final DTM. The "get_points" function iterates through all the data matrix elements and returns only those with a value. Finally, a file with XYZ coordinates, Intensity and H_{diff} is saved to a disk.

```
import os
import time

# POINTS EXTRACTION FUNCTION
@numba.njit("float64[:, :](float64[:, :, ::1])")  # output and input data types for Numba JIT decorator
def get_points(grid_data_matrix):
    i, j, k = grid_data_matrix.shape  # number of rows, columns and point data parameters in grid data_matrix
    mask = np.zeros(i * j, dtype=np.bool_)  # creates a zero array with the given dimensions to store
    the numpy mask
    point_cloud = np.reshape(grid_data_matrix, (i * j, k))  # 3D array -> 2D array
    for ij in numba.prange(i * j):  # loop through all grid points
        mask[ij] = ((point_cloud[ij, 0] != 0) | (point_cloud[ij, 1] != 0) | (point_cloud[ij, 2] != 0) |
                    (point_cloud[ij, 3] != 0))  # mask[ij] is True in non-zero points
    point_cloud = point_cloud[mask]  # apply mask (take only non-zero points)
    return point_cloud  # return only real, measured points

directory = os.path.dirname(os.path.abspath(__file__))  # get program run file directory
timestr = time.strftime("%Y%m%d-%H%M%S")  # get current time
filename = directory + f"/saved data/LIDAR/{timestr}.txt"  # new file path
xyz = get_points(grid_data_matrix)  # get only real, measured points
np.savetxt(filename, xyz, delimiter=',', fmt='%.3f, %.3f, %.3f, %i, %.3f')  # save points (point cloud) to file
```

Appendix B. Field test video

A video from the field tests of the described measurement platform:
https://youtu.be/GSM3mxwRbJU

ACKNOWLEDGEMENTS

Financial support: AGH University of Krakow – subsidy no. 16.16.150.545, IDUB for the University of Krakow – research project partly supported by program "Excellence initiative – research university" for the AGH University of Krakow.

REFERENCES

[1] Rao AS, Radanovic M, Liu Y, Hu S, Fang Y, Khoshelham K, Palaniswami M, Ngo T. (2022). Real-time monitoring of construction sites: sensors, methods, and applications. *Autom. Constr*., 136, 104099. doi:10.1016/j.autcon.2021.104099.

[2] Clemen C, Schröder M, Kaiser T, Romanschek E. (2021). IFCTerrain – a free and open source tool to convert digital terrain models (DTM) to OpenBIM Industry Foundation Classes (IFC). ISPRS Annals, VIII-4/W2, 145–51. doi:10.5194/isprs-annals-VIII-4-W2-2021-145-2021.

[3] Marotta F, Teruggi S, Achille C, Vassena GPM, Fassi F. (2021). Integrated laser scanner techniques to produce high-resolution DTM of vegetated territory. *Remote Sens., 13*(13), 2504. doi:10.3390/rs13132504.

[4] Gbopa AO, Ayodele EG, Okolie CJ, Ajayi AO, Iheaturu CJ. (2021). Unmanned aerial vehicles for three-dimensional mapping and change detection analysis. *GaEE, 15*(1), 41–61. doi:10.7494/geom.2021.15.1.41.

[5] Matrone F, Gallitto F, Lingua AM, Maschio PF. (2024). Exploitation of the number of return echoes for DTM extraction from point clouds acquired by LiDAR UAS DJI Zenmuse L1. ISPRS Archives, 48, 233–40. doi:10.5194/isprs-archives-XLVIII-2-2024-233-2024.

[6] Bao Z, Hossain S, Lang H, Lin X. (2023). A review of high-definition map creation methods for autonomous driving. E*ng. Appl. Artif. Intell.*, 122, 106125. doi:10.1016/j.engappai.2023.106125.

[7] Elhashash M, Albanwan H, Qin R. (2022). A review of mobile mapping systems: from sensors to applications. *Sensors, 22*(11), 4262. doi:10.3390/s22114262.

8] Kaczmarek A, Rohm W, Klingbeil L, Tchórzewski J. (2022). Experimental two-dimensional extended Kalman filter sensor fusion for low-cost GNSS/IMU/odometers precise positioning system. *Meas*., 193, 110963. doi:10.1016/j.measurement.2022.110963.

[9] Pereira T, Gameiro T, Viegas C, Santos V, Ferreira N. (2023). Sensor integration in a forestry machine. *Sensors, 23*(24), 9853. doi:10.3390/s23249853.

[10] Dahal P, Arrigoni S, Bijelic M, Braghin F. (2024). MLIO: multiple LiDARs and inertial odometry. In: Proceedings of the Advanced Vehicle Control Symposium, Cham (Switzerland), *Springer Nature Switzerland*, 833–42. doi:10.1007/978-3-031-70392-8_118.

[11] Immonen M, Niskanen I, Hallman L, Keränen P, Hiltunen M, Kostamovaara J, Heikkilä R. (2021). Fusion of 4D point clouds from a 2D profilometer and a 3D LiDAR on an excavator. *IEEE Sens. J., 21*(15), 17200–6. doi:10.1109/JSEN.2021.3078301.

[12] Gagula AC, Bolanio KP, Bermoy MM, Salupado CA, Arayan MG. (2023). Integrating geospatial techniques and UAS technology to update LiDAR DTM for flood modeling in Las Nieves, Agusan del Norte, Philippines. ISPRS Archives, 48, 109–16. doi:10.5194/isprs-archives-XLVIII-4-W6-2022-109-2023.

[13] Ćwiąkała P, Gruszczyński W, Stoch T, Puniach E, Mrocheń D, Matwij W, Matwij K, Nędzka M, Sopata P, Wójcik A. (2020). UAV applications for determination of land deformations caused by underground mining. *Remote Sens., 12*(11), 1733. doi:10.3390/rs12111733.

[14] Skaloud J, Schaer P, Stebler Y, Tomé P. (2010). Real-time registration of airborne laser data with sub-decimeter accuracy. ISPRS J. Photogramm. *Remote Sens., 65*(2), 208–17. doi:10.1016/j.isprsjprs.2009.12.003.

[15] Ernst D, Jüngerink J, Kindervater L, Moftizadeh R, Alkhatib H, Vogel S. (2021). Data fusion for georeferencing a laser scanner-based multi-sensor system in a city environment. In: Proceedings of the 24th International Conference on Information Fusion, Sun City (South Africa), *IEEE*, 1–8. doi:10.23919/FUSION49465.2021.9627026.

[16] Koch P, Lacroix S. (2016). Managing environment models in multi-robot teams. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Daejeon (Korea), *IEEE*, 5722–8. doi:10.1109/IROS.2016.7759842.

[17] Wang L, Li S, Du M, Ji G, Li K, Liu D. (2025). Terrain preview detection system based on loosely coupled and tightly coupled fusion with LiDAR and IMU. *Meas*., 242, 115924. doi:10.1016/j.measurement.2024.115924.

[18] Fankhauser P, Bloesch M, Hutter M. (2018). Probabilistic terrain mapping for mobile robots with uncertain localization. *IEEE Robot. Autom. Lett., 3*(4), 3019–26. doi:10.1109/LRA.2018.2849506.

[19] Maturana D, Chou PW, Uenoyama M, Scherer S. (2018). Real-time semantic mapping for autonomous off-road navigation. In: Field and Service Robotics: Results of the 11th International Conference, Cham (Switzerland), Springer International Publishing, 335–50. doi:10.1007/978-3-319-67361-5_22.

[20] Li Y, Ding L, Zheng Z, Yang Q, Zhao X, Liu G. (2018). A multi-mode real-time terrain parameter estimation method for wheeled motion control of mobile robots. *Mech. Syst. Signal Process*., 104, 758–75. doi:10.1016/j.ymssp.2017.11.038.

[21] Ding N, Zhou Y, Li D, Zeng K. (2024). Real-time deformation monitoring of large diameter shield tunnel based on multi-sensor data fusion technique. *Meas*., 225, 114061. doi:10.1016/j.measurement.2023.114061.

[22] Phillips TG, Guenther N, McAree PR. (2017). When the dust settles: the four behaviors of LiDAR in the presence of fine airborne particulates. *J. Field Robot., 34*(5), 985–1009. doi:10.1002/rob.21701.

[23] Robertson G. (2016). Mining's imperative for enhanced accuracy of real-time spatial data and reporting. In: Proceedings of the 16th International Congress for Mine Surveying, Brisbane (Australia), [publisher unknown].

[24] Li X, Liu C, Li J, Baghdadi M, Liu Y. (2021). A multi-sensor environmental perception system for an automatic electric shovel platform. *Sensors, 21*(13), 4355. doi:10.3390/s21134355.

[25] Hirayama M, Guivant J, Katupitiya J, Whitty M. (2019). Path planning for autonomous bulldozers. *Mechatronics*, 58, 20–38. doi:10.1016/j.mechatronics..01.001.

[26] Zhang L, Zhao J, Long P, Wang L, Qian L, Lu F, Song X, Manocha D. (2021). An autonomous excavator system for material loading tasks. *Sci. Robot., 6*(55), eabc3164. doi:10.1126/scirobotics.abc3164.

[27] Chae MJ, Lee GW, Kim JY, Park JW, Cho MY. (2011). A 3D surface modeling system for intelligent excavation system. *Autom. Constr., 20*(7), 808–17. doi:10.1016/j.autcon.2011.02.003.

[28] Guan T, He Z, Song R, Zhang L, Manocha D. (2023). TNES: terrain traversability mapping, navigation and excavation system for autonomous excavators on worksite. *Auton. Robots., 47*(6), 695–714. doi:10.1007/s10514-023-10113-9.

[29] Leica Geosystems AG. (2022 May 16). 4 Create a surface model [video on the Internet]. YouTube [cited 2025 Sep 21]. Available from: https://www.youtube.com/watch?v=jiZy5DLMT6o.

[30] Trimble CE&C Help. (2018 Nov 28). Trimble Earthworks tutorial – record a point [video on the Internet]. YouTube [cited 2025 Sep 21]. Available from: https://www.youtube.com/watch?v=q3tS63oO3ls.

[31] Unicontrol ApS. [date unknown]. Learn how to utilize 3D machine control: create surface from points [Internet]. Unicontrol ApS [cited 2024 Nov 18]. Available from: https://www.unicontrol.com/learn-machine-control.

[32] Makin AS. (2023). Makin' 3D: user guide 2.20 [Internet]. Makin AS [cited 2025 Sep 21]. Available from: https://maskinsystem.com/wp-content/uploads/makin_excavator3d_usermanual-3.pdf.

[33] Novatron Oy. (2021). Xsite Pro LandNova X: brukerhåndbok for gravemaskin version 1.14.3.80 [Internet]. Novatron Oy [cited 2025 Sep 21]. Available from: https://hella.no/wp-content/uploads/2021/12/Xsite_Pro_user_manual_LandNova_X_1_6_2021_NO_8.11.2021.pdf.

[34] AEC Software Technology. (2018 Oct 8). Asbuilt mapping with machine control [video on the Internet]. YouTube [cited 2025 Sep 21]. Available from: https://www.youtube.com/watch?v=UluWSMWvpGw.

[35] Trimble Construction Support. (2021 Jul 27). Trimble Earthworks – how to: track mapping in Earthworks 2.5 [video on the Internet]. YouTube [cited 2025 Sep 21]. Available from: https://www.youtube.com/watch?v=hoPJnnoSWdQ.

[36] Leica Geosystems AG. (2023 Jan 23). How to use surface logging in MC1 [video on the Internet]. YouTube [cited 2025 Sep 21]. Available from: https://www.youtube.com/watch?v=J63XPDNNods.

[37] Unicontrol ApS. (2024 May 23). Dredging operations in Norway: Hylland's success story revealed [Internet]. LinkedIn [cited 2025 Sep 21]. Available from: https://www.linkedin.com/pulse/dredging-operations-norway-hyllands-success-story-revealed-zgbhf/.

[38] Topcon Positioning Systems. (2017). 3D-MC software: user's manual [Internet]. Topcon Positioning Systems [cited 2025 Sep 21]. Available from: https://myconstructiontechnology.com/wp-content/uploads/2021/04/3dmc_um.pdf.

[39] Makin' 3D Machine Control. (2022 Oct 20). Makin' surfaces – productivity and efficiency at a glance [video on the Internet]. YouTube [cited 2025 Sep 21]. Available from: https://www.youtube.com/watch?v=bXwhVzUbOyw.

[40] Bhandari V, James J, Phillips T, McAree PR. (2024). Probabilistic height grid terrain mapping for mining shovels using LiDAR. *IFAC-PapersOnLine, 58*(22), 54–9. doi:10.1016/j.ifacol.2024.09.290.

[41] D'Adamo T. (2023). Building and maintaining real-time terrain maps for autonomous bulldozer mining [dissertation]. Brisbane (Australia), The University of Queensland. doi:10.14264/212c46e.

[42] Okunsky MV, Nesterova NV. (2019). Velodyne LIDAR method for sensor data decoding. IOP Conf. *Ser. Mater. Sci. Eng., 516*(1), 012018. doi:10.1088/1757-899X/516/1/012018.

[43] SciPy Community. [date unknown]. SciPy documentation: scipy.spatial.transform.Rotation [Internet]. SciPy [cited 2025 Feb 19]. Available from: https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.transform.Rotation.html.

[44] Velodyne Lidar Inc. (2019). Velodyne Lidar Puck VLP-16: datasheet 63-9229 Rev-K [Internet]. Velodyne Lidar Inc [cited 2025 Sep 21]. Available from: https://ouster.com/downloads/velodyne-downloads.

[45] Afzalaghaeinaeini A, Seo J, Lee D, Lee H. (2022). Design of dust-filtering algorithms for LiDAR sensors using intensity and range information in off-road vehicles. *Sensors, 22*(11), 4051. doi:10.3390/s22114051.

[46] Zhao S, Lu TF, Koch B, Hurdsman A. (2015). 3D stockpile modelling and quality calculation for continuous stockpile management. *Int. J. Miner Process*., 140, 32–42. doi:10.1016/j.minpro.2015.04.012.