**Control and Cybernetics**
vol. **54** (2025) No. 3
pages: 341-358
DOI: 10.2478/candc-2025-0012

# The use of Octree in point cloud analysis with application to cultural heritage*

by

**Rafał Bieńkowski**[1] **and Krzysztof Rutkowski**[2]

[1] Systems Research Institute, Polish Academy of Sciences,
01-447 Warsaw, Poland, Newelska 6,
[2] Cardinal Stefan Wyszyński University,
Faculty of Mathematics and Natural Sciences. School of Exact Sciences,
Warsaw, Poland, Dewajtis 5

**Abstract:** In this article, we propose a method for analysing 3D point clouds. To this aim, we take advantage of the octree data structure to compress and analyse the 3D point data. The motivation for the study undertaken comes from the field of Cultural Heritage. As 3D acquisition methods become more and more ubiquitous in this field, there is an increasing need for methods, which help to efficiently store, display, and share large data sets. There is also a need to segment and classify the recorded data. We show that the octree data structure provides an efficient tool for handling complex 3D data, coming from different applications. Although we tested our method on a data set, coming from the field of archaeology, based on photogrammetric method, we believe that the approach proposed has much wider use. The method proposed can be applied to any 3D data, represented in a given coordinate system.

**Keywords:** octree, 3D point cloud, data classification

## 1. Introduction

3D documentation methods are becoming standard in many domains, e.g. in the field of various cultural heritage applications, such as architecture documentation, large-scale landscape analysis, archaeological field and artifact documentation and more (see: Magnani et al., 2020). Nowadays, we observe constant improvement in remote sensing tools (e.g. laser scanning, photogrammetry, LiDAR) and 3D model generation (3D processing software). Also the accuracy and resolution

*Submitted:October 2025; Acceoted: November 2025.

of the widely available 3D data have greatly increased (see: Marin-Buzón et al., 2021).

We address two aspects of handling large 3D point clouds: size reduction for computation (see Section 5), and point labelling (see: Algorithm 2 and, for instance, Fig. 9.1). For both of these aspects, we apply the octree data structure approach. Our choice of the octree data structure is motivated by its simplicity, low computational cost and natural fit for 3D data analysis.

In the here presented method of applying the octree data structure, the considered data set is inscribed into the single cuboid and next, subsequently, divided in to smaller eight sub-cuboids (see Section 4 and Figure 4.1).

For a 3D point cloud to be useful, in most applications, the points need to be classified first. In the examples (presented in Section 9) we focus on the detection (and classification) of three types of regions of point clouds: 1) regions of surface and sufficient number of 3D point to produce reliable documentation, 2) overlapping parts of documented object, 3) regions of potentially insufficient point density.

The organization of the paper is as follows. In Section 2 we present the remote sensing tools used in data collecting and the literature devoted to octree data structure and its applications in various fields. In Section 3 we provide the motivation for our investigation and the description of the problem. In Section 4 we describe octree data structure for spatially referenced 3D point clouds. Section 5 describes the idea and the theory of data reduction for the purpose of analysis. In Section 6 we present the classification method that we then use in Section 7. Section 7 contains the main result of the paper, which is the pipeline for 3D point clouds processing and description of all of the algorithms, which are used in the presented pipeline. In Section 8 we describe data sets used in Section 9, where we present the results of numerical experiment. Section 10 concludes the paper.

## 2. State of the art

In this section we would like to present the background for both octree data structure and data collection used in numerical experiment, with brief description of data collecting and processing method.

The idea of octree data structure was first published by Donald Meagher in 1980 as a method to represent and process 3D objects in computer graphics (Meagher, 1980). In modern scientific literature, there is an abundance of publications on the application of octree data structure in different fields of computer science. Below we would like to mention just a couple of examples:

1. Octree Grid Topology – used to segment objects that have a known topology (Bai, Han and Prince, 2007, 2009),
2. Nearest neighbour search (Drost and Ilic, 2018; Chen et al., 2023),
3. Colour quantization (Park, Kim and Cha, 2016; Gervautz and Purgathofer, 1988),
4. Data compression (Fu et al., 2022; Schnabel and Klein, 2006).

3D point clouds creation and analysis are a time and resource-consuming process. There are many methods for creating point clouds, such as laser scanning or Structure from Motion (SfM), see Markiewicz et al. (2019). Until a few years ago these methods were available to professionals via expensive or specialized devices or software. This situation has changed radically in the recent years with the wide availability of tools for the creation of 3D point clouds and 3D models, like in the case of RealityCapture (now rebranded as RealityScan, see Epic Games, 2025). In the case of our experiment, we use the data generated with the SfM method. This method is based on collecting 2D images and processing them in a way to create a 3D object (Westby et al., 2012).

## 3. Problem statement

In the present contribution, we propose a scheme (data processing pipeline), based on octree data structure, for the size reduction and classification of 3D point clouds. The method presented can be applied to spatially referenced, not necessarily geo-graphically referenced, 3D point clouds. To reduce data, for computation, and classify given 3D point cloud we use the octree data structure, as described in Section 4. Points are included into a given cuboid, of the octree data structure, on the basis of their distances. In the method presented, points are included into 6 levels of depth of cuboids (levels 0 to 5). In general, the number of levels is optional and should be specified as the parameter $M$, described further on. The octree data structure subdivision is illustrated in Fig. 4.1. In consequence, points can be represented by cuboids of various sizes, to which they belong. Each point in the point cloud (data set) is represented by its position in space, in a given spatial reference system, and optional other features, e.g. its colour, given in the RGB system.

## 4. Preprocessing

As mentioned before (see Section 3) data must be spatially referenced. In general, the spatial reference of the data in the pipeline does not have to be geographic. It is also possible to apply the here presented procedure to the data referenced in the local spatial reference system.

From the data set, in a given $y, x, z$ coordinates system (axes), we extract the following information:

- $y_{\min}$, $x_{\min}$, $z_{\min}$ – minimal values in a given coordinate system of points in the data set,
- $y_{\max}$, $x_{\max}$, $z_{\max}$ – maximal values in a given coordinate system of points in data set,

and, basing on collected data, the level $lev = 0$ cuboid is generated as the one containing all points from the data set and all of the points are marked as belonging to this cuboid (see Table 1a).

It is characteristic for geographic data to reverse the order of the first two axes, namely the first coordinate values are given from the $y$ axis, the second coordinate values are given from the $x$ axis and the third coordinate values given are from the $z$ axis. In our application, we choose to work with the geographic order of the axes and therefore we used this order $yxz$ of data in our algorithm.

The cuboids at the division level $lev$ (in our case, $0 \leq lev \leq M$), which are non-empty/contain points, are of sizes

$$\frac{y_{\max} - y_{\min}}{2^{lev}} \times \frac{x_{\max} - x_{\min}}{2^{lev}} \times \frac{z_{\max} - z_{\min}}{2^{lev}},$$

where $lev$ represents the current level of division in the octree data structure (see Fig. 4.1). Summing up, the octree data structure constitutes the following procedure:

1. at the start, the given 3D point cloud is enclosed in the first cuboid ($zero$ level) of size
   $(y_{\max} - y_{\min}) \times (x_{\max} - x_{\min}) \times (z_{\max} - z_{\min})$,
2. if the cuboid, at the given level $lev$, contains any point from the 3D point cloud, it is splitted into 8 cuboids of equal sizes (by dividing each edge of this cuboid by two),
3. for each new cuboid, step 2 is repeated, as long as the level of depth $lev$ is less or equal to $M$.

The preview of this procedure is illustrated in Fig. 4.1.

The result of this procedure is constituted by the cuboids up to the desired level of division (nesting). From the cuboids of maximal depth ($lev = M$), we extract the non-empty cuboids (containing points). On the basis of these cuboids, we perform a labelling (see Algorithm 2).

The octree structure consists (in Matlab, see Sven, 2013) of the following entries:

- *double* Points - a matrix $n \times 3$, which contains information about the coordinates of each point in three dimensions format, where $n$ is the number of 3D points in the analyzed 3D point cloud,

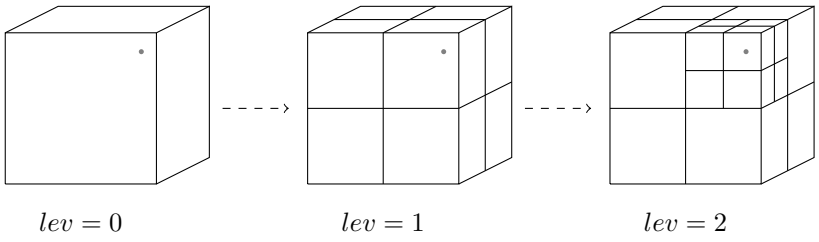



Figure 4.1: Octree procedure preview

- *double* PointBins - a matrix $n \times 1$, which contains information, for each point, about the number of the cuboid of the max depth, in which this point is located,
- *double* BinCount - number of all cuboids generated,
- *double* BinBoundaties - a matrix $m \times 6$, which contains information about the coordinates of each cuboid ($y_{\min}$,$x_{\min}$, $z_{\min}$, $y_{\max}$, $x_{\max}$, $z_{\max}$),
- *double* BinParents - a matrix $m \times 1$, which contains information, for each cuboid, about the number of the parent cuboid.

In our approach, we have chosen to use cuboids. It is, however, possible to implement an analogous method by using exclusively cubes, by initializing data to be contained in a cube. We found that cuboids are better suited for our application. In the real-life applications, the whole process benefits from the use of cuboids, instead of cubes, as cuboids are able to fit better into the investigated shapes.

It should be explained why data normalization or enforcing a cubic representation is not applied in the presented method. The proposed approach operates on relative spatial relationships rather than on absolute coordinate magnitudes. For instance, a dataset of sizes $y_{size} = y_{max} - y_{min} = 1$, $x_{size} = x_{max} - x_{min} = 2$, and $z_{size} = z_{max} - z_{min} = 3$, where $y_{size}$, $x_{size}$, and $z_{size}$ represent the sizes of $yxz$, respectively, can be directly represented as a cuboid. In this representation, the spatial relationships and proportional ratios between the axes are preserved. In contrast, forcing a cubic normalization would rescale the axes unevenly, thereby distorting the natural proportions and potentially reducing the facility of interpretation of spatial relationships.

## 5. Data reduction

For the purpose of data size reduction, we apply the octree data structure. This method significantly reduces the volume, of data required for further analysis. The data size is reduced from $q \times 3$, where $q$ is the number of points within the

cuboids, to $2 \times 3$ dimensions: $(min_y, min_x, min_z)$ and $(max_y, max_x, max_z)$, which define the bounding cuboid.

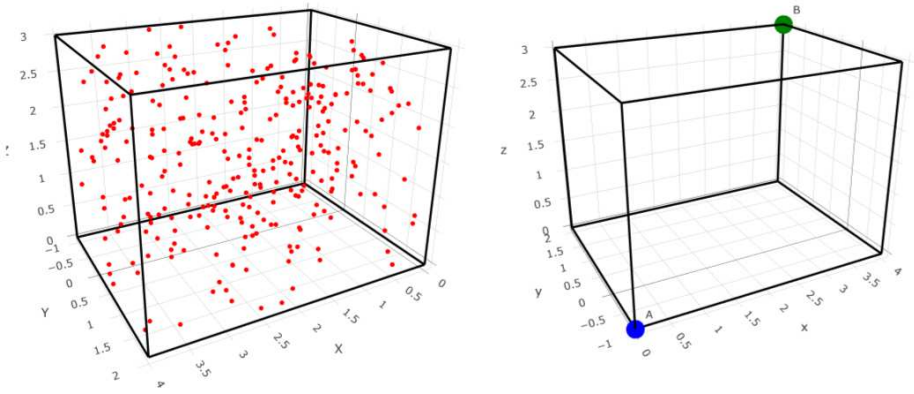



Figure 5.1: Data reduction from (left) 300 3D points of 3 coordinate components $yxz$ to 2 3D points A and B of 3 coordinates components $yxz$.

| | |
|---|---|
| 1 | $y_1, x_1, z_1$ |
| 2 | $y_2, x_2, z_2$ |
| 3 | $y_3, x_3, z_3$ |
| $\vdots$ | |
| $q$ | $y_q, x_q, z_q$ |

Table 1a: Full data in one cuboid

| | |
|---|---|
| $A$ | $y_A, x_A, z_A$ |
| $B$ | $y_B, x_B, z_B$ |

Table 1b: Reduced data in one cuboid

For example, consider the dataset 4 (see Section 8), represented as a point cloud, containing, at the depth $lev = 6$, 3 092 396 points, each with three coordinates $(y, x, z)$, resulting in 9 277 188 numerical values. Using the proposed octree reduction, the dataset can be represented by approximately 10 890 cuboids, each defined by two 3D points $(min, max)$, corresponding to 32 670 numerical values in total. This results in a data size reduction by nearly the factor of 300.

## 6. Classification method

In this section we present a method to classify the points of the given 3D point cloud. We use the following input data and notations:

- 3D point cloud with spatial reference, where each point is represented with $(y, z, z)$ coordinates,
- $M$ - number of levels of depth in octree structure,
- $lev$ - current level of depth, $lev = 0$ corresponds to the starting cuboid, in which a given 3D point cloud is enclosed, $0 \leq lev \leq M$,
- cuboids appearing at the level $M$, $lev = M$, are called cuboids of max depth,
- the values $z_k = z_{\min} + k \cdot \frac{z_{\max} - z_{\min}}{2^M}$, $k = 0, 1 \ldots, 2^M - 1$ represent the layers with respect to the $z$ value at level $M$.

The method to classify the point cloud is constructed on the basis of two algorithms, which sort and label the data as follows:

- (Sorting) for each rectangle $[y_i, y_{i+1}] \times [x_j, x_{j+1}]$, $i, j \in 0, \ldots, 2^M - 1$, where $y_i := y_{\min} + i \cdot \frac{y_{\max} - y_{\min}}{2^M}$, $x_j := x_{\min} + j \cdot \frac{x_{\max} - x_{\min}}{2^M}$, we find cuboids (i.e. $z_k$)

$$[y_i, y_{i+1}] \times [x_j, x_{j+1}] \times [z_k, z_{k+1}], \quad k = 0, 1 \ldots, 2^M - 1$$

  which contain at least one point (if there is such) and we store this data (i.e. $z_k$) in the increasing order.
- (Labeling) for each rectangle $[y_i, y_{i+1}] \times [x_j, x_{j+1}]$ we perform the labelling of cuboids by using $z_k$ as follows:
  - Let $z_{k_0}$ be a minimal element of this set. We find $z_l = z_{\min} + l \cdot \frac{z_{\max} - z_{\min}}{2^M}$ such that $l > k_0$ and cuboid $[y_i, y_{i+1}] \times [x_j, x_{j+1}] \times [z_l, z_{l+1}]$ is empty (i.e. $z_l$ is not in the set obtained from sorting). Then, each cuboid, obtained from sorting, with the corresponding $[y_i, y_{i+1}] \times [x_j, x_{j+1}]$, which has $z_k < z_l$, we classify as "surface",
  - All other cuboids in the corresponding $[y_i, y_{i+1}] \times [x_j, x_{j+1}]$ we classify as "above".

## 7. Pipeline description

In this section the proposed pipeline is presented. The pipeline is constructed with the following components and procedures:

1. Prepossessing of the data - described in Section 4 - the input data is a 3D point cloud and the output is octree data structure. This step allows us to reduce the data for analysis in the next steps (see Section 5).

2. Sorting algorithm is applied for the octree structure. The formal description of this algorithm is presented in Section 6 and the pseudocode is presented in this section (see Algoritm 1).
3. Classification of the cuboids of the octree data structure. The formal description of this algorithm is presented in Section 6 and the pseudocode is presented in this section (see Algoritm 2).

We present the pipeline with visualisation of data from the chosen steps in Fig. 7.1.

Recall that we use the following formulations:

$$y_i := y_{\min} + i \cdot \frac{y_{\max} - y_{\min}}{2^M}, \quad i = 0, 1 \ldots, 2^M - 1,$$
$$x_j := x_{\min} + j \cdot \frac{x_{\max} - x_{\min}}{2^M}, \quad j = 0, 1 \ldots, 2^M - 1,$$
$$z_k := z_{\min} + k \cdot \frac{z_{\max} - z_{\min}}{2^M}, \quad k = 0, 1 \ldots, 2^M - 1.$$

We are given the octree data structure as described in Section 4. To sort the data with respect to $z$ variable for each rectangle range $[y_i, y_{i+1}] \times [x_j, x_{j+1}]$, $i, j \in 0, 2^M - 1$ we perform Algorithm 1, as specified here.

**Algorithm 1** Sorting the cuboids

**for** $i, j \in 0, 1, \ldots, 2^M - 1$ **do**
  $\texttt{len}(i, j) = 0$
  **for** $k \in \{0, 1, \ldots, 2^M - 1\}$ **do**
    **if** cuboid $[y_i, y_{i+1}] \times [x_j, x_{j+1}] \times [z_k, z_{k+1}]$ contains points **then**
      $\texttt{temp}(i, j, \texttt{len}(i, j)) = k$
      $\texttt{len}(i, j) = \texttt{len}(i, j) + 1$
    **end if**
  **end for**
**end for**

The result of Algorithm 1 is constituted by the list `temp`, which contains an information on non-empty cuboid $[y_i, y_{i+1}] \times [x_j, x_{j+1}] \times [z_k, z_{k+1}]$ whenever $k = \texttt{temp}(i, j, v)$ for some $v$ as well as the length of the list for $\texttt{temp}(i, j, \cdot)$ for given $i, j \in \{0, 1, \ldots, 2^M - 1\}$. Let us note that this automatically implies the information on the data of cuboids that are of level depth $M$, which contain at least one point. Moreover, thanks to Algorithm 1, we know that for each index $(i, j)$ for given nonempty cuboid $[y_i, y_{i+1}] \times [x_j, x_{j+1}] \times [z_{\texttt{temp}(i,j,v)}, z_{\texttt{temp}(i,j,v)+1}]$ the above non-empty cuboid in the same range $[y_i, y_{i+1}] \times [x_j, x_{j+1}]$ is $[y_i, y_{i+1}] \times [x_j, x_{j+1}] \times [z_{\texttt{temp}(i,j,v+1)}, z_{\texttt{temp}(i,j,v+1)+1}]$ (whenever it exists).

Then, we perform the algorithm for labeling the cuboids, which is presented as the Algorithm 2.

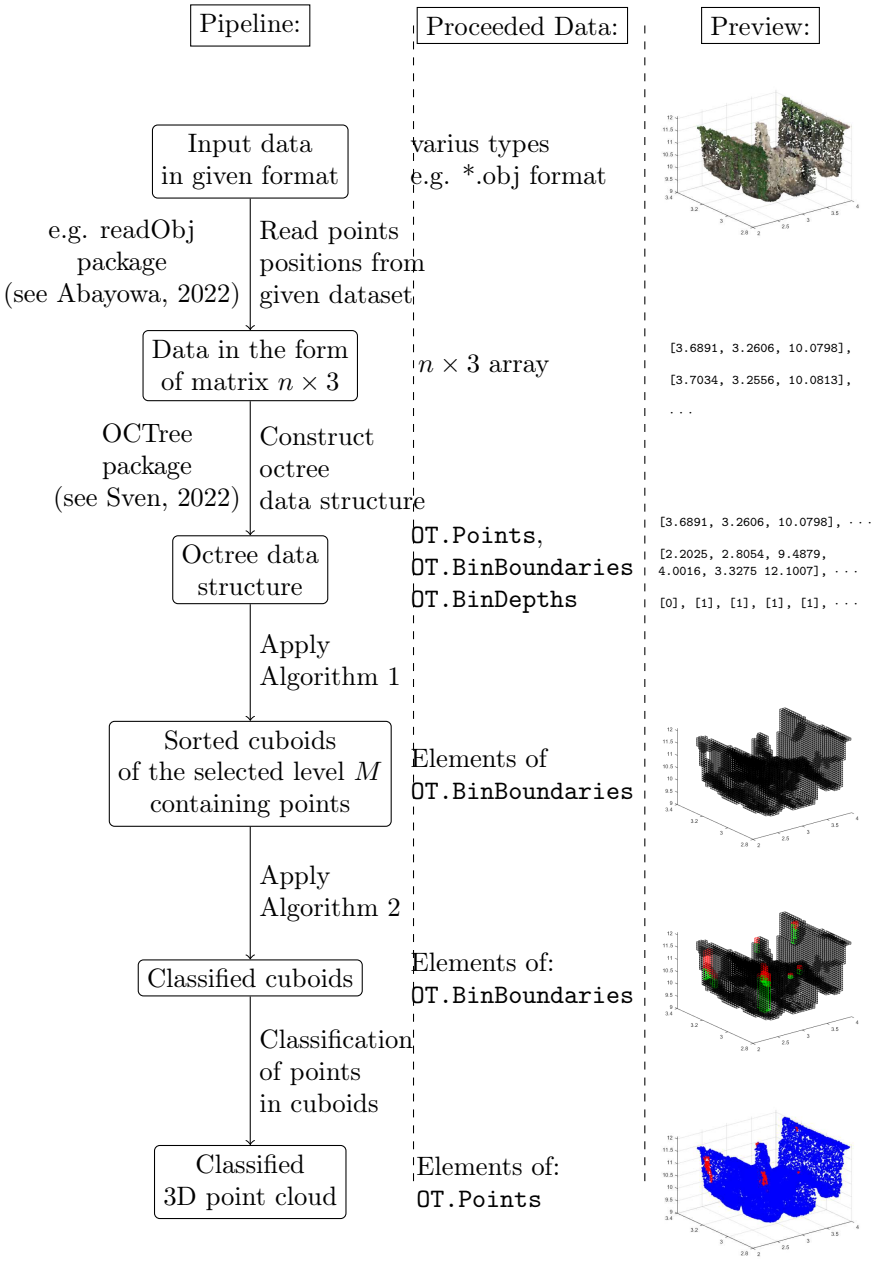



Figure 7.1: Pipeline of the approach with exemplary illustrations

**Algorithm 2** Labeling of the cuboids

**for** $i, j \in \{0, 1, \ldots, 2^M - 1\}$ **do**
  Set $\texttt{flag} = 0$
  **for** $v \in \{0, 1 \ldots, \texttt{len}(i, j) - 1\}^*$ **do**
    **if** $\texttt{flag} = 1$ **then**
      classify the cuboid $[y_i, y_{i+1}] \times [x_j, x_{j+1}] \times [z_{\texttt{temp}(i,j,v)}, z_{\texttt{temp}(i,j,v)+1}]$ as "above"
    **end if**
    **if** $\texttt{temp}(i, j, v + 1) - \texttt{temp}(i, j, v) \neq 1$ † **then**
      $\texttt{flag} = 1$
      (*)
    **end if**
  **end for**
**end for**
Classify the cuboids of selected depth level $M$ containing vertices which are not classified "above" as "surface".

Since each point from the cloud point data belongs to some selected *lev* depth cuboid, we may classify the points according to the labeling of the cuboid to which the data belongs.

We use Algorithm 3 to fill the gaps between cuboids "surface"-"above" or "above"-"above" for given $[y_i, y_{i+1}] \times [x_j, x_{j+1}]$ range. Below, we present the additional part, which is included in the original Algorithm 2.

**Algorithm 3** Filling the gaps

(*) Fill in the cuboid of range $[y_i, y_{i+1}] \times [x_j, x_{j+1}] \times [z_{\texttt{temp}(i,j,v)+1}, z_{\texttt{temp}(i,j,v+1)}]$

## 8. Data set

Below we present a short description of the data set used in the investigation. The data set comes from the photogrammetric documentation (based on image processing) of an archaeological trench and its closest surroundings. Photogrammetric documentation in our data set has been created in Agisoft Metashape based on the orthogonal photos taken from a drone.

*Let us note that if $\texttt{len}(i, j) - 1 = -1$ then the loop is not opening, meaning that there is no cuboid or only one in the range $[y_i, y_{i+1}] \times [x_j, x_{j+1}]$, containing at least one point.

†By this condition we ensure that the cuboids $[y_i, y_{i+1}] \times [x_j, x_{j+1}] \times [z_{\texttt{temp}(i,j,v)}, z_{\texttt{temp}(i,j,v)+1}]$ and $[y_i, y_{i+1}] \times [x_j, x_{j+1}] \times [z_{\texttt{temp}(i,j,v+1)}, z_{\texttt{temp}(i,j,v+1)+1}]$ have no common faces.

The documented trenches were located in Ukimerioni hill, in the surroundings of Bagrati Cathedral (Kutaisi, Georgia). The point cloud covers various areas, roughly between 20 by 120 meters horizontally. Vertically, the point cloud represents up to around 20 meters of height.

All points have location coordinates in a georeferenced coordinates system. In our case, it is the UTM coordinate system with data represented as latitude, longitude and elevation. The georeferenced system for our data set is WGS 84. For each 3D point, the data takes the following form: $yxz$ localization and other optional feature like RGB colour channels value.

## 9. Numerical experiments

### 9.1. General characteristics

In our experiments and analysis, we use a numerical data set coming from an area of cultural heritage interest (an archaeological trench and its surroundings, both documented during ongoing fieldwork).

The level of depth of cuboids was set to $M$ equal to 5 or 6 (starting from the initial level 0). Each of the points has a representation in $y, x, z$ values of the georeferenced coordinates system and the colour in the RGB colour system.

The software used for calculations was MATLAB® 2024b with the help of the readObj package (see Abayowa, 2022) and the OCTree package (see Sven, 2022).

We consider data from four sets. Sets 1, 2, 3, 4 are made up of 626 831 points, 1 219 669 points, 993 802 points, and 3 092 396 points, respectively. The level of depth of cuboids was set to 5 for the data sets 1-3 (starting from the initial level 0) and to 6 for the data set 4.

For the respective data sets the number of cuboids is as follows:

- For set 1 – in total 12 297 cuboids[‡] on levels 0-5, with 5 003 cuboids of level 5,
- For set 2 – in total 8 281 cuboids on levels 0-5, with 3 479 cuboids of level 5,
- For set 3 - in total 4 792 cuboids on levels 0-5, with 1 677 cuboids of level 5,
- For set 4 - in total 28 873 cuboids on levels 0-6, with 10 890 cuboids of level 6.

Each of the points has a representation in $y, x, z$ values of the georeferenced coordinates system and the colour in the RGB system. The numerical experiments

[‡]Let us note that if cuboid did not contain points at some given level e.g. $lev = 2$, it were not divided on further steps.

were performed on the computer with the following hardware parameters: processor Intel(R) Core(TM) i9-14900K, 64 GB RAM DDR5. The software used for calculations was MatLab 2024b with the help of the readObj package (see Abayowa, 2022) and the OCTree package (see Sven, 2022).

## 9.2. Results for individual data sets

### 9.2.1. Data set 1

Among 5 003 cuboids of level depth *lev*, which contains points, 4 920 are classified as "surface" and 19 are classified as "above". Moreover, there are 19 gap cuboids. The result in the classification of a total of 626 831 points is the following: 620 929 points are classified as "surface" and 5 902 points are classified as "above". The result of applying octree data structure on the data by choosing the most nested cuboids, which contain points of the data, is displayed in Fig. 9.1.

### 9.2.2. Data set 2

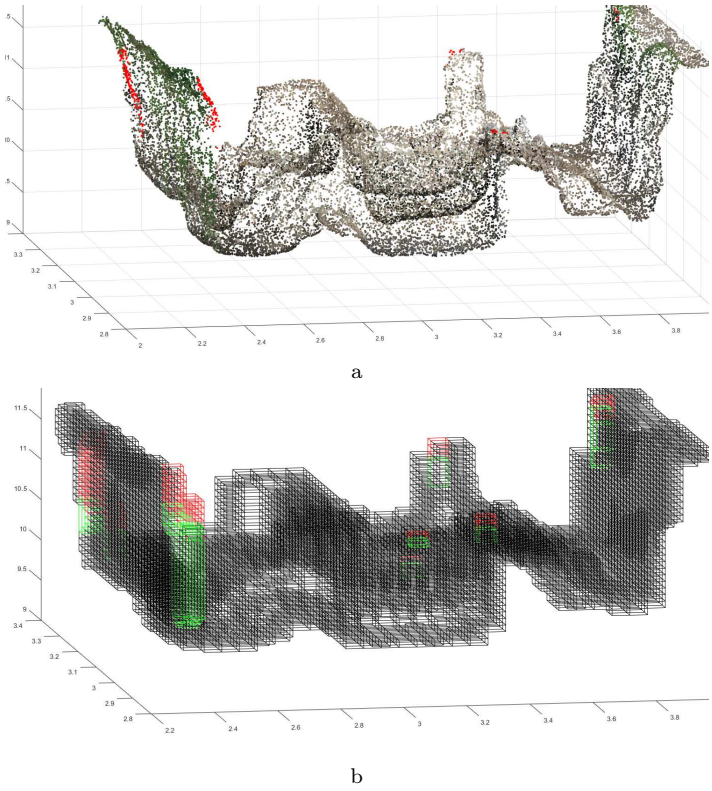
Among 3 479 cuboids of level depth *lev*, which contains points, 3 171 are classified as "surface" and 308 are classified as "above". Moreover, there are 74 gap cuboids. The result in the classification of a total of 1 219 669 points is the following: 1 146 545 points are classified as "surface" and 73 124 points are classified as "above". The result of applying octree data structure on the data by choosing the most nested cuboids, which contain points of the data, is displayed in Fig. 9.2.

### 9.2.3. Data set 3

Among 1 677 cuboids of level depth *lev*, which contains points, 1 192 are classified as "surface" and 485 are classified as "above". Moreover, there are 202 gap cuboids. The result in the classification of a total of 993 802 points is the following: 826 804 points are classified as "surface" and 166 998 points are classified as "above". The result of applying octree data structure on the data by choosing the most nested cuboids, which contain points of the data, is displayed in Fig. 9.3.

### 9.2.4. Data set 4

Among 10 890 cuboids of level depth *lev*, which contains points, 9 517 are classified as "surface" and 1 373 are classified as "above". Moreover, there are



a

b

Figure 9.1: Data set 1 representation (a) - coloured points (here, we visualize only 10% of all the points with true colors; in red colour the points, classified as "'above"', are marked). (b) - Cuboids of maximal depth containing points with the coloured regions of interest, black - "surface", red - "above", green - "gaps"

385 gap cuboids. The result in the classification of a total of 3 092 396 points is the following: 2 874 103 points are classified as "surface" and 218 293 points are classified as "above". The result of applying octree data structure on the data by choosing the most nested cuboids, which contain points of the data, is displayed in Fig. 9.4.

Let us note that in order to classify the data cuboid as "above", i.e. a cuboid of dimensions $[y_i, y_{i+1}] \times [x_j, x_{j+1}] \times [z_k, z_{k+1}]$, there must exist a cuboid of level $lev$, containing points of dimensions $[y_i, y_{i+1}] \times [x_j, x_{j+1}] \times [z_k \times z_{k+1}]$, where $z_s < z_k$. Since our original data comes from the SfM method, we might be missing information about the surface below, which is fully covered by an obstacle (like

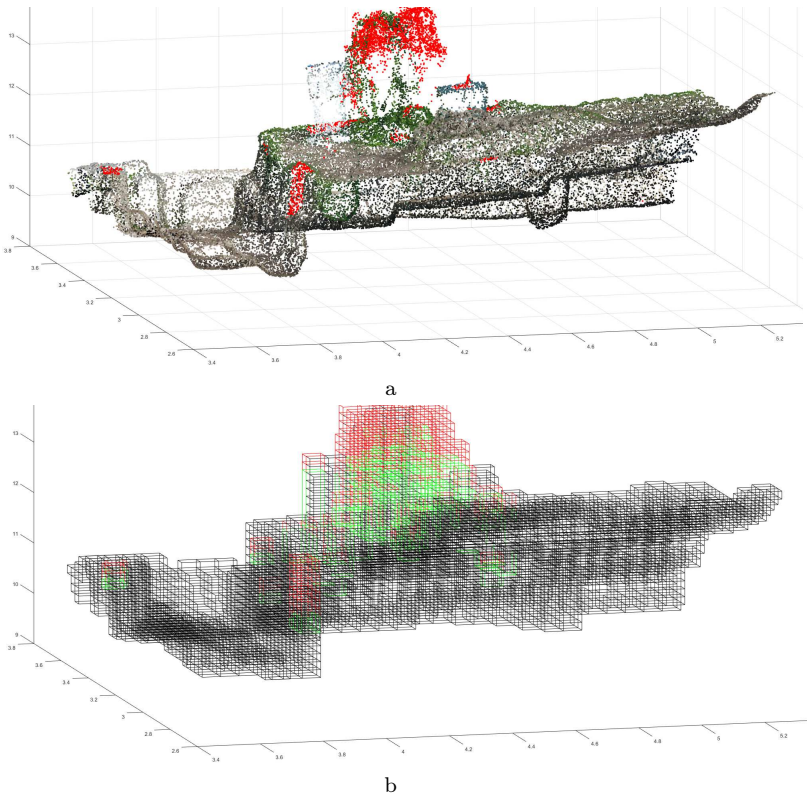


Figure 9.2: Data set 2 representation (a) - coloured points (here, we visualize only 10% of all the points with true colors; in red colour the points, classified as "'above"', are marked). (b) - Cuboids of maximal depth containing points with the coloured regions of interest, black - "surface", red - "above", green - "gaps"

a roof), as illustrated in Fig. 9.4 (a) and (b). The further investigation of such obstacles remains a topic of our future research.

## 9.3. Advantages of using octree data structure in 3D point cloud analysis

The advantages of the presented approach are the following:

1. by taking advantage of the octree data structure, we achieved compression of large amounts of points from a data set for the purposes of conducted analysis,

2. octree data structure allows for scalable processing of extremely large datasets. This ensures that computational resources are allocated proportionally to data complexity, improving performance for both high- and low-density datasets,
3. through the selection of the maximal level depth of the octree data structure, we can recover a more efficient approximation of the analysed objects.

## 10. Conclusions

Computational experiments confirm that using octree data structure can give a significant reduction of data size, which is of practical value in pattern recognition and increases the efficiency of 3D point cloud classification. This results in the possibility of efficiently displaying and interpreting (via classification) the point cloud data. The proposed method of size reduction and classification within the framework of octree structure gives very promising results and will be further explored in future research.

## References

Abayowa, B. (2007) *readObj*. https://www.mathworks.com/matlabcentral/fileexchange/18957- readobj. [Online; accessed 3-December-2022].

Bai, Y., Han, X. and Prince, J.L. (2007) Octree Grid Topology Preserving Geometric Deformable Model for Three-Dimensional Medical Image Segmentation. In: N. Karssemeijer and B. Lelieveldt, eds., *Proceedings of the Information Processing in Medical Imaging*. Springer, Berlin, Heidelberg, 556–568.

Bai, Y., Han, X. and Prince, J.L. (2009) Digital topology on adaptive octree grids. *Journal of Mathematical Imaging and Vision*, 34, 165–184.

Chen, F., Ying, R., Xue, J.,Wen, F. and Liu, P. (2023) ParallelNN: A Parallel Octree-based Nearest Neighbor Search Accelerator for 3D Point Clouds. In: *Proceedings of the 2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 403–414.

Drost, B.H. and Ilic, S. (2018) Almost constant-time 3D nearest-neighbor lookup using implicit octrees. *Machine Vision and Applications*, 29, 299–311. https:// doi.org/10.1007/s00138-017-0889-4.

Epic Games (2025) *RealityScan - 3D models from images and laser scans*. https://www.realityscan.com/en-US. Accessed: 2025-10-31.

Fu, C., Li, G., Song, R., Gao, W. and Liu, S. (2022) Octattention: Octree-based large-scale contexts model for point cloud compression. In: *Proceedings of the Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI Press, Palo Alto, California 36, 625–633.

Gervautz, M. and Purgathofer, W. (1988) A simple method for color quantization: Octree quantization. In: *Proceedings of the New Trends in Computer Graphics: Proceedings of CG International'88*. Springer, 219–231.

Magnani, M., Douglass, M., Schroder, W., Reeves, J. and Braun, D.R. (2020) The digital revolution to come: Photogrammetry in archaeological practice. *American Antiquity*, 85, 737–760.

Marín-Buzón, C., Pérez-Romero, A., López-Castro, J.L., Ben Jerbania, I. and Manzano-Agugliaro, F. (2021) Photogrammetry as a new scientific tool in archaeology: Worldwide research trends. *Sustainability*, 13, 5319.

Markiewicz, J., Pilarska, M., Łapiński, S., Kaliszewska, A., Bie ńkowski, R. and Cena, A. (2019) Quality assessment of the use of a medium format camera in the investigation of wall paintings: An image-based approach. *Measurement*, 132, 224–237. https://doi.org/10.1016/j.measurement.2018.07.001.

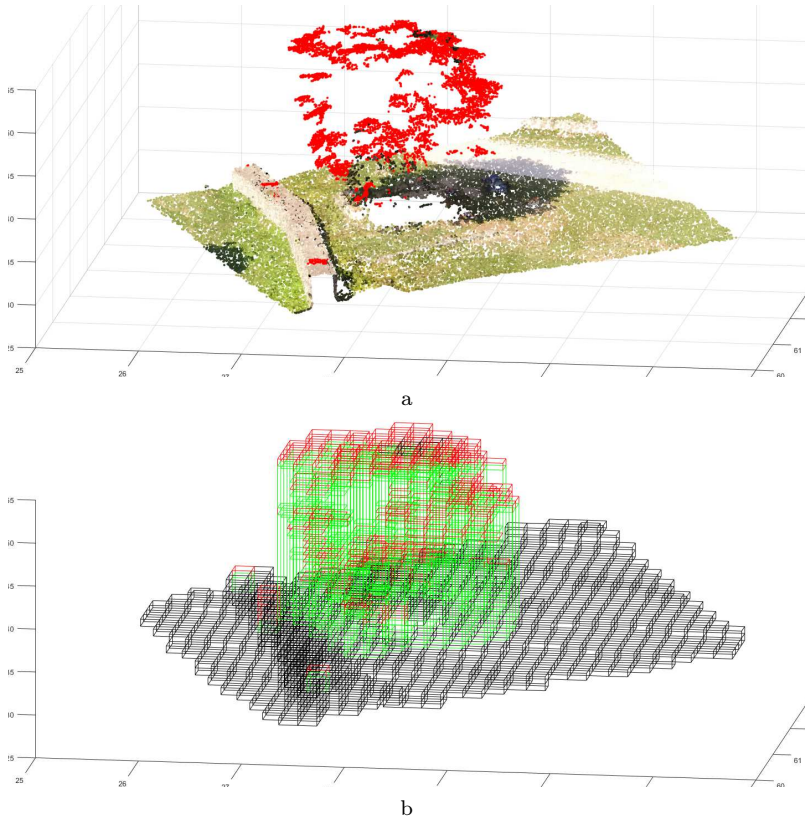
Meagher, D. (1980) *Octree Encoding: A New Technique for the Representation, Manipulation and Display of Arbitrary 3-D Objects by Computer* Rensselaer Polytechnic Institute, Image Processing Laboratory.

Park, H., Kim, K. and Cha, E.Y. (2016) An Effective Color Quantization Method Using Octree-Based Self-Organizing Maps. *Computational Intelligence and Neuroscience 2016*. Wiley Online Library, first published 14 January 2016. https:// doi.org/ 10.1155/2016/5302957.

Schnabel, R. and Klein, R. (2006) Octree-based Point-Cloud Compression. *PBG@ SIGGRAPH*. Symposium on Point Based Graphics 06. The Eurographics Association, 3.

Sven (2013) *Octree – partitioning 3D points into spatial subvolumes*. https:// www.mathworks.com/ matlabcentral/fileexchange/40732-octree-partitioning-3d-points-into-spatial-subvolumes. [Online, accessed 3-December-2022].

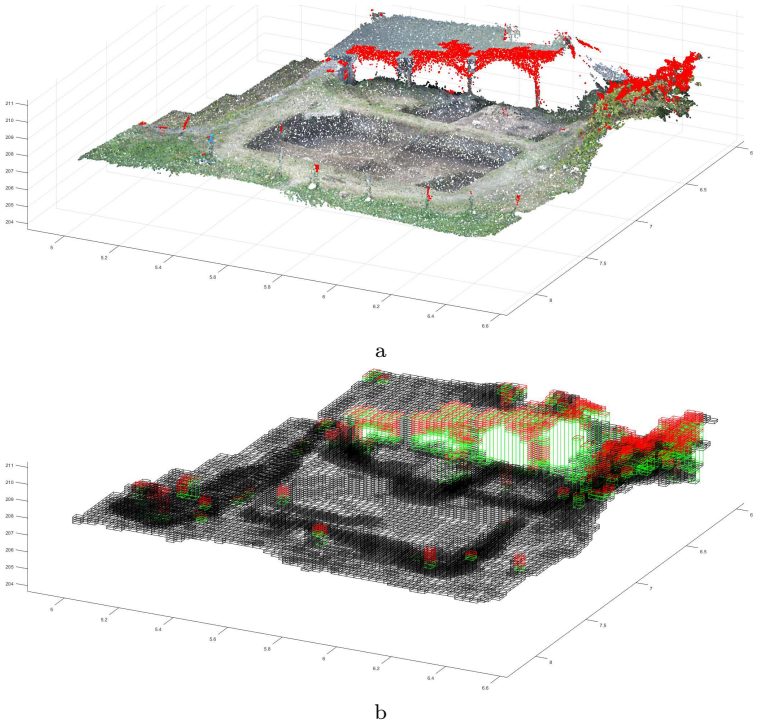
Westoby, M., Brasington, J., Glasser, N., Hambrey, M. and Reynolds, J. (2012) Structure-from-Motion photogrammetry: a novel, low-cost tool for geomorphological applications. *Geomorphology*, 3, 300–314.



a

b

Figure 9.3: Data set 3 representation (a) - coloured points (here, we visualize only 10% of all the points with true colors; in red colour the points, classified as "‘above"’, are marked). (b) - Cuboids of maximal depth containing points with the coloured regions of interest, black - "surface", red - "above", green - "gaps"



a

b

Figure 9.4: Data set 4 representation (a) - coloured points (here, we visualize only 10% of all the points with true colors; in red colour the points, classified as "‘above"’, are marked). (b) - Cuboids of maximal depth containing points with the coloured regions of interest, black - "surface", red - "above", green - "gaps"