UDC 004.421.2
DOI https://doi.org/10.26661/2786-6254-2023-2-02

ALGORITHMS FOR CONVERTING VOXEL 3D MODELS INTO POLYGONAL ONES

Duvanov S. S.
Postgraduate Student at the Department of Information Technologies
Sumy State University
M. Sumtsova str., 2, Sumy, Ukraine
orcid.org/0000-0001-7766-6837
samduvanov@gmail.com

Baranova I. V.
Candidate of Technical Sciences, Associate Professor,
Associate Professor at the Department of Information Technologies
Sumy State University
M. Sumtsova str., 2, Sumy, Ukraine
orcid.org/0000-0002-3767-8099
i.baranova@cs.sumdu.edu.ua

Key words: *3d model, voxel, algorithm, converting, polygonal mesh.*

This document explores the using of voxels today in such areas as medicine, 3D scanning, game industry and why do we need to convert them to polygonal mesh. It also describes and analyses the currently existing algorithms for converting voxel 3D models into polygonal ones. Aim of converting algorithms is extracting the surface information from the voxel grid and generate a polygonal mesh composed of triangles that approximate the original shape. The research identified about 10 existing algorithms: Marching Cubes, Dual Contouring, Surface Nets, Voxel Carving, Sparse Voxel Octree, Surface Extraction from Volume Data (SEV), Cubical Marching Squares, Adaptive Grid Subdivision, Occupancy Networks, Deep Implicit Fields. Their features, pros and cons were described. It was chosen among the converting algorithms, taking into account the simplicity, efficiency and availability of implementation, one algorithm, which can become the basis of the program for converting voxel 3d models into polygonal mesh. The choice of algorithm depends on various factors, including the desired surface quality, computational efficiency, handling of sharp features, and implementation complexity, but there were several main criteria selected. The chosen algorithm is Marching Cubes, because it is widely used, has available implementations, has good performance. Future research will focus on creating a converter program since there are currently no freely available converters that would produce a high-quality editable polygonal mesh. All existing voxel editors and online converters export the voxel 3d model into polygonal mesh without any approximation, so the export result is a cube set, poorly united into one surface with surface breaks. Export polygonal mesh models also have a problem with a large number of triangles, which makes the model hard to edit. To create a converter that solves those problems, first, we must choose an algorithm that will become the basis of the future converter. Target algorithm will be customized in future to improve the quality of output polygonal mesh comparing to the existing convertors.

АЛГОРИТМИ ПЕРЕТВОРЕННЯ ВОКСЕЛЬНИХ 3D МОДЕЛЕЙ В ПОЛІГОНАЛЬНІ

Дуванов С. С.
аспірант кафедри інформаційних технологій
Сумський державний університет
вул. М. Сумцова, 2, Суми, Україна
orcid.org/0000-0001-7766-6837
samduvanov@gmail.com

Баранова І. В.
кандидат технічних наук, доцент,
доцент кафедри інформаційних технологій
Сумський державний університет
вул. М. Сумцова, 2, Суми, Україна
orcid.org/0000-0002-3767-8099
i.baranova@cs.sumdu.edu.ua

Ключові слова: *3d модель, воксель, алгоритм, конвертація, полігональна сітка.*

У даній статті досліджується використання вокселів на сьогодення в таких галузях, як медицина, 3D-сканування, ігрова індустрія і чому нам потрібно перетворювати їх в полігональну сітку. Також описуються і аналізуються наявні алгоритми для перетворення воксельних 3D-моделей в полігональні. Метою алгоритмів конвертування є видобування інформації про поверхню з воксельної сітки та генерація полігональної сітки, складеної з трикутників, які наближають оригінальну форму. У дослідженні ідентифіковано близько 10 існуючих алгоритмів: Marching Cubes, Dual Contouring, Surface Nets, Voxel Carving, Sparse Voxel Octree, Surface Extraction from Volume Data (SEV), Cubical Marching Squares, Adaptive Grid Subdivision, Occupancy Networks, Deep Implicit Fields. Були описані їх характеристики, переваги і недоліки. Серед алгоритмів конвертування було обрано один алгоритм, враховуючи простоту, ефективність та доступність реалізації, який може стати основою програми для перетворення воксельних 3D-моделей в полігональну сітку. Вибір алгоритму залежить від різних факторів, включаючи бажану якість поверхні, обчислювальну ефективність, обробку гострих деталей та складність імплементації, але було вибрано кілька основних критеріїв. Обраним алгоритмом є Marching Cubes, оскільки він широко використовується, має доступні реалізації та високу продуктивність. Майбутні дослідження будуть спрямовані на створення програми-конвертора, оскільки на сьогоднішній день не існує вільно доступних конверторів, які б створювали високоякісну редаговану полігональну сітку. Усі існуючі редактори воксельних моделей та онлайн-конвертори експортують воксельну 3D-модель в полігональну сітку без будь-якої апроксимації, тому результат експорту – це набір кубів, погано об'єднаних в одну поверхню, що має розриви. Моделі полігональної сітки також мають проблему з великою кількістю трикутників, що ускладнює редагування моделі. Для створення конвертора, який вирішує ці проблеми, спочатку ми повинні вибрати алгоритм, який стане основою майбутнього конвертора. Цільовий алгоритм буде дороблено в майбутньому для покращення якості вихідної полігональної сітки порівняно з існуючими конверторами.

Introduction

The representation of three-dimensional objects is a fundamental aspect of computer graphics and computer-aided design. Voxel-based models, which divide space into small volumetric elements called voxels, provide a straightforward and intuitive way to represent 3D shapes.

However, in many applications, it is desirable to convert voxel models into polygonal representations composed of triangles, as polygons are widely supported by rendering engines and 3D graphics software.

The choice of conversion algorithm depends on various factors, including the desired surface quality, computational efficiency, handling of sharp features, and implementation complexity. Each algorithm has its own set of pros and cons, and selecting the most appropriate one requires careful consideration of the specific requirements and constraints of the application.

In this paper, we will explore and compare several algorithms for converting voxel 3D models into polygonal representations. These algorithms aim to extract the surface information from the voxel grid and generate a mesh composed of triangles that approximate the original shape.

We will discuss their underlying principles, advantages, disadvantages, and trade-offs. By understanding the strengths and limitations of these algorithms, we can make informed decisions when choosing the most suitable approach for a given task, balancing simplicity, computational efficiency, and the quality of the resulting polygonal meshes.

Future research will focus on creating a converter program since there are currently no freely available converters that would produce a high-quality editable polygonal mesh. All existing voxel editors and online converters export the voxel model into polygonal mesh without any approximation, so the export result is a cube set, poorly united into one surface with surface breaks. Export mesh models also have a problem with a large number of triangles, which makes the model hard to edit. To create a converter that solves those problems, first, we must choose an algorithm that will become the basis of the future converter. The choice of such an algorithm is the goal of this research.

Literature Review

The problem of converting voxels to polygons is not new and takes roots from visualizing computed tomography results [1; 2]. Voxels are widely employed in various fields, including medicine [3; 4; 5; 6] and 3D scanning [7; 8], due to their versatility and ability to represent complex three-dimensional structures with precision and detail. Voxel-based 3D scanning finds applications in various industries, including manufacturing, architecture, and entertainment. It is used for tasks such as reverse engineering, quality control, and digital preservation of cultural artifacts. Additionally, it plays a crucial role in computer graphics and gaming, where voxels are used to generate realistic and interactive 3D environments.

Algorithms for converting voxel 3D models into polygonal ones are important in this conversion process. Various algorithms have been developed over the years, each with its own strengths and limitations, offering different trade-offs between simplicity, computational efficiency, and the quality of the resulting mesh.

There are several algorithms and techniques used for converting voxel-based 3D models into polygonal representations. Let's take a closer look at the most well-known conversion algorithms, which are mentioned in literary sources.

1. Marching Cubes: The Marching Cubes algorithm is one of the most popular methods for converting voxel data into polygonal meshes. It works by evaluating the voxel grid and creating surface polygons based on the density values of neighboring voxels. The algorithm determines the configuration of the surface within each voxel and generates triangles to approximate the surface [9]. It can handle arbitrary resolutions of voxel grids, provides good performance and efficiency, produces smooth surfaces for most cases. It has the following disadvantages: can create non-manifold and self-intersecting surfaces, may produce topological inconsistencies for certain configurations, requires special handling for sharp features and thin structures, triangle quality can vary, resulting in uneven surface representation.

2. Dual Contouring: Dual Contouring is another popular technique for voxel-to-polygon conversion. It focuses on generating higher-quality meshes compared to Marching Cubes by using the actual intersection points between voxel edges and the surface to create vertices. It allows for more accurate representation of complex shapes and smooth surfaces [10]. It handles sharp features and thin structures effectively, can generate watertight and manifold meshes, provides better control over mesh topology. It has the following disadvantages: more computationally expensive than Marching Cubes, requires additional steps to handle irregular voxel grids, can generate more triangles compared to other algorithms, sensitive to noisy voxel data, leading to surface artifacts.

3. Surface Nets: The Surface Nets algorithm is a variation of Marching Cubes that aims to generate watertight and manifold meshes. It constructs the surface by placing polygons on the edges where the surface crosses the voxel grid. Surface Nets can provide more consistent triangle sizes and better preserve sharp and thin features [11]. It also can handle irregularly sampled voxel grids. It has the following disadvantages: less widely used compared to Marching Cubes, can produce lower-quality surfaces for complex shapes, may have difficulty representing complex topologies, more computationally expensive compared to Marching Cubes.

4. Voxel Carving: Voxel Carving is a technique that starts with a large voxel grid encompassing the entire object and progressively carves away voxels to refine the shape. By iteratively removing voxels based on their consistency with the observed data, a polygonal representation of the object can be obtained [12; 13]. It can handle complex and irregularly shaped objects, does not require explicit surface extraction algorithms, can produce detailed and accurate representations. It has the following disadvantages: requires careful parameter tuning for optimal results, may generate noisy or incomplete surfaces if not properly configured, can be computationally expensive, especially for large voxel grids, requires careful handling of occlusion and self-intersection.

5. Sparse Voxel Octree: Sparse Voxel Octree (SVO) is a data structure that represents a voxel model as an octree, where each node in the tree either contains subnodes or represents a voxel. By traversing the octree and determining the surface at different levels of resolution, polygonal meshes can be extracted [14; 15]. It efficiently represents complex structures with varying resolution, allows for adaptive level-of-detail representation, can handle large-scale voxel data efficiently, supports accurate surface extraction. It has the following disadvantages: more complex to implement compared to other algorithms, requires additional memory overhead for storing the octree, can introduce artifacts near octree boundaries, may produce lower-quality surfaces for certain configurations.

6. Surface Extraction from Volume Data (SEV): SEV is an algorithm that generates polygonal meshes by directly extracting surface information from volume data. It operates by analyzing the voxel connectivity and marching along the surface to create polygons. SEV can handle irregularly sampled voxel grids and can generate high-quality meshes [16; 17]. It provides good surface quality and accuracy, allows for efficient extraction of the surface information, can handle complex topologies and sharp features effectively. It has the following disadvantages: requires additional steps for post-processing and mesh refinement, may have difficulty preserving fine details, can be computationally expensive for large voxel grids, sensitivity to noise in the voxel data can result in surface artifacts.

7. Cubical Marching Squares: Cubical Marching Squares operates on a 3D voxel grid, generating polygons based on the voxel densities. It can produce watertight meshes and handle sharp features and thin structures effectively [18]. It provides good control over triangle quality and surface topology. It has the following disadvantages: less widely used compared to Marching Cubes, requires additional steps for handling irregular voxel grids, can generate more triangles compared to other algorithms, sensitivity to noisy voxel data can lead to surface artifacts.

8. Adaptive Grid Subdivision: This technique subdivides the voxel grid into smaller cells, allowing for more accurate surface representation. By recursively subdividing cells based on the presence of surface intersections, adaptive grid subdivision methods can produce detailed and smooth polygonal meshes [19]. It handles complex shapes and varying levels of detail effectively, provides control over the level of refinement, can generate high-quality meshes with consistent triangle sizes. It has the following disadvantages: more computationally expensive compared to other algorithms, requires additional steps for adaptive subdivision and refinement, higher memory requirements for storing the subdivided grid, may produce higher triangle counts for highly detailed meshes.

9. Occupancy Networks: Occupancy Networks are deep learning-based methods that learn to predict the occupancy of each voxel in the 3D space. By training a neural network on voxel data, the network can generate a polygonal mesh by predicting the surface based on the learned occupancy probabilities [20]. It can handle complex shapes and topologies, provides continuous and smooth surface representations, can generate high-quality meshes with accurate surface details. It has the following disadvantages: requires training a deep learning model on a large dataset, computationally intensive during training and inference, difficulties in handling fine details and sharp features, limited control over the mesh topology and triangle count.

10. Deep Implicit Fields: Deep Implicit Fields are another deep learning approach for voxel-to-polygon conversion. Instead of predicting occupancy, these methods learn to directly model the implicit surface representation. By training a neural network to encode the implicit surface function, polygonal meshes can be extracted from the learned model [21; 22]. It can handle complex shapes and topologies, provides continuous and smooth surface representations, allows for high-quality mesh generation. It has the following disadvantages: requires training a deep learning model on a large dataset, computationally intensive during training and inference, difficulties in handling fine details and sharp features, limited control over the mesh topology and triangle count.

These algorithms provide a range of approaches for converting voxel-based 3D models into polygonal representations. Depending on the specific requirements, application constraints, and desired output quality, one or a combination of these algorithms can be employed to achieve the desired results.

Methods

In the research process, an analytical method was used for systematic comparison and analysis of existing voxel conversion algorithms. This made it possible to examine in detail the characteristics of each algorithm, to determine their advantages and disadvantages, and to identify trends in their use.

During the use of the analytical method, parameters and stages of work of each algorithm were carefully considered, which contributed to an objective comparison of their efficiency and suitability for specific tasks. The analysis included a logical understanding of the operation of each algorithm, taking into account its capabilities and limitations.

The application of the analytical method also made it possible to identify key factors when choosing the optimal algorithm: Surface Quality, Topological Consistency, Performance, Handling Complex Shapes, Sharp Features and Thin Structures, Availability and Implementation. This approach turned out to be extremely useful for determining the optimal voxel conversion algorithm that takes into account the specific application requirements in the research area and the desired output quality.

Results

The priority of applying the considered algorithms was analyzed taking into account the above key factors.

1. Surface Quality: If achieving high-quality surface representation with accurate details is a priority, algorithms like Dual Contouring, Surface Nets, or Occupancy Networks may be suitable choices.

2. Topological Consistency: If preserving topological consistency is crucial, algorithms like Surface Nets or Cubical Marching Squares offer better control over mesh topology and can generate watertight and manifold meshes.

3. Performance: If computational efficiency is a primary concern, algorithms like Marching Cubes or Voxel Carving may be more suitable, as they are generally faster and have been widely optimized and implemented.

4. Handling Complex Shapes: If your voxel models contain complex shapes, algorithms like Dual Contouring, Occupancy Networks, or Deep Implicit Fields can handle intricate topologies more effectively.

5. Sharp Features and Thin Structures: If your voxel models include sharp features or thin structures that need to be accurately represented, algorithms like Dual Contouring or Cubical Marching Squares provide better preservation of such details.

6. Availability and Implementation: Consider the availability of existing implementations, libraries, or frameworks that provide the algorithm you choose. This factor can affect the ease of implementation and integration into your existing workflow.

The obtained comparison results are illustrated in Table 1.

The simplicity of implementation and mesh quality are the key considerations of the research, so here are two algorithms that strike a good balance:

1. Marching Cubes: Marching Cubes is a widely used algorithm for converting voxel data into polygonal meshes. It offers a good balance between simplicity and mesh quality. The algorithm is well-established and has numerous implementations available, making it easier to find code examples and resources for implementation. While Marching Cubes may not generate the highest-quality meshes in all cases, it typically produces smooth surfaces and can handle a variety of voxel grids efficiently. It is a popular choice due to its simplicity and versatility.

2. Cubical Marching Squares: Cubical Marching Squares is an extension of the traditional Marching Squares algorithm to 3D voxel grids. It offers a good compromise between simplicity and mesh quality. Like Marching Cubes, it is relatively straightforward to implement and provides good control over the resulting mesh topology. Cubical Marching Squares is particularly effective at preserving sharp features and thin structures in the generated meshes. While it may not be as widely used as Marching Cubes, it is still a viable option that offers simplicity and good mesh quality.

Both algorithms strike a balance between simplicity and mesh quality, making them accessible choices

Table 1

Algorithms comparison results

	Is widely used	Has available implementations	Good performance	High-surface quality
Marching Cubes	Yes	Yes	Yes	No
Dual Contouring	Yes	Yes	No	Yes
Surface Nets	Yes	Yes	No	Yes
Voxel Carving	No	Yes	Yes	No
Sparse Voxel Octree	Yes	Yes	No	No
Surface Extraction from Volume Data (SEV)	No	No	No	No
Cubical Marching Squares	No	Yes	No	No
Adaptive Grid Subdivision	No	No	No	No
Occupancy Networks	No	No	No	Yes
Deep Implicit Fields	No	No	No	Yes

for many applications. Marching Cubes is more commonly used and offers broader support, while Cubical Marching Squares provides better preservation of sharp features and thin structures. You can choose the algorithm based on your specific needs and priorities, keeping in mind the trade-offs between simplicity, mesh quality, and the desired characteristics of your resulting meshes.

So, focusing on the balance of simplicity and efficiency, we can say that the most suitable algorithm for the converter can be Marching Cubes.

Discussion

The conversion of voxel-based 3D models into polygonal representations is a fundamental task in computer graphics and computer-aided design. A review of available sources showed that there are currently no effective converters for creating editable polygon meshes and no criteria for choosing effective conversion algorithms, so this research is relevant.

This paper has explored several algorithms for this purpose, examining their underlying principles, advantages, disadvantages, and trade-offs.

The analysis of the most common conversion algorithms made it possible to identify the key factors for choosing the optimal algorithm depending on the application requirements and the desired output quality. The comparative characteristics of the algorithms according to these factors, given in Table 1, made it possible to single out two algorithms that have a good balance of simplicity and efficiency. As a result of the comparison of the algorithms, it was concluded that the Marching Cubes algorithm is the most suitable for implementation in the future converter. This choice is justified by considering various factors such as simplicity, efficiency, and the availability of implementations.

This initiative defines the practical relevance of the research and its significance for practical applications in industrial and technical fields. This makes the article a resource for future research, that will focus on refining existing converting Marching Cubes algorithm, developing new techniques to address the limitations of current approaches and solving issues, related to exporting voxel models to polygon meshes, with better approximation.

BIBLIOGRAPHY

1. Frieder G., Gordon D., Reynolds R. Back-to-Front Display of Voxel Based Objects. *IEEE Computer Graphics and Applications*. 1985. Vol. 5, № 1. P. 52–60. URL: https://doi.org/10.1109/mcg.1985.276273.
2. Copula, a new approach for optimum design of Voxel-based GNSS tropospheric tomography based on the atmospheric dynamics / R. Mousavian et al. *GPS Solutions*. 2022. Vol. 26, № 4. URL: https://doi.org/10.1007/s10291-022-01340-1.
3. Dotremont K. From medical images to 3D model: processing and segmentation. *Handbook of Surgical Planning and 3D Printing*. 2023. P. 65–91. URL: https://doi.org/10.1016/b978-0-323-90850-4.00009-0.
4. 3D Reconstruction of Human Body Biometry / G. Trujillo-Hernández et al. *Optoelectronic Devices in Robotic Systems*. Cham, 2022. P. 195–225. URL: https://doi.org/10.1007/978-3-031-09791-1_8.
5. Whole-Body Voxel-Based Personalized Dosimetry: The Multiple Voxel S-Value Approach for Heterogeneous Media with Nonuniform Activity Distributions / M. S. Lee et al. *Journal of Nuclear Medicine*. 2017. Vol. 59, № 7. P. 1133–1139. URL: https://doi.org/10.2967/jnumed.117.201095.
6. SimCVD: Simple Contrastive Voxel-Wise Representation Distillation for Semi-Supervised Medical Image Segmentation / C. You et al. *IEEE Transactions on Medical Imaging*. 2022. P. 1. URL: https://doi.org/10.1109/tmi.2022.3161829.
7. Li Huanmei. 3D Indoor Scene Reconstruction and Layout Based on Virtual Reality Technology and Few-Shot Learning. *Computational Intelligence and Neuroscience*. 2022. Vol. 2022. P. 1–9. URL: https://doi.org/10.1155/2022/4134086.
8. The Polygonal 3D Layout Reconstruction of an Indoor Environment via Voxel-Based Room Segmentation and Space Partition / F. Yang et al. *ISPRS International Journal of Geo-Information*. 2022. Vol. 11, № 10. P. 530. URL: https://doi.org/10.3390/ijgi11100530.
9. History of the Marching Cubes Algorithm / W. E. Lorensen et al. *IEEE Computer Graphics and Applications*. 2020. Vol. 40, № 2. P. 8–15. URL: https://doi.org/10.1109/mcg.2020.2971284.
10. Dual contouring of hermite data / T. Ju et al. *ACM Transactions on Graphics*. 2002. Vol. 21, № 3. P. 339–346. URL: https://doi.org/10.1145/566654.566586.
11. Gibson S. F. F. Constrained elastic surface nets: Generating smooth surfaces from binary segmented data. *Medical Image Computing and Computer-Assisted Intervention – MICCAI'98*. Berlin, Heidelberg, 1998. P. 888–898. URL: https://doi.org/10.1007/bfb0056277.
12. Voxel carving-based 3D reconstruction of sorghum identifies genetic determinants of light interception efficiency / M. Gaillard et al. *Plant Direct*. 2020. Vol. 4, № 10. URL: https://doi.org/10.1002/pld3.255.
13. Persistent Homology for 3D Reconstruction Evaluation / A. Gutierrez et al. *Computational Topology in Image Context*. Berlin, Heidelberg, 2012. P. 139–147. URL: https://doi.org/10.1007/978-3-642-30238-1_15.
14. Laine S., Karras T. Efficient sparse voxel octrees. *The 2010 ACM SIGGRAPH symposium*, Washington, D.C., 19–21 February 2010. New York, New York, USA, 2010. URL: https://doi.org/10.1145/1730804.1730814.

15. McGraw T. High-quality real-time raycasting and raytracing of streamtubes with sparse voxel octrees. *2020 IEEE Visualization Conference (VIS)*, Salt Lake City, UT, USA, 25–30 October 2020. URL: https://doi.org/10.1109/vis47514.2020.00011.
16. Feature sensitive surface extraction from volume data / L. P. Kobbelt et al. *The 28th annual conference*, Not Known. New York, New York, USA, 2001. URL: https://doi.org/10.1145/383259.383265.
17. Uribe Lobello R., Denis F., Dupont F. Adaptive surface extraction from anisotropic volumetric data: contouring on generalized octrees. *Annals of telecommunications*. 2013. Vol. 69, № 5–6. P. 331–343. URL: https://doi.org/10.1007/s12243-013-0369-4.
18. Cubical Marching Squares: Adaptive Feature Preserving Surface Extraction from Volume Data / C. C. Ho et al. *Computer Graphics Forum*. 2005. Vol. 24, № 3. P. 537–545. URL: https://doi.org/10.1111/j.1467-8659.2005.00879.x.
19. Real-time 3D reconstruction at scale using voxel hashing / M. Nießner et al. *ACM Transactions on Graphics*. 2013. Vol. 32, no. 6. P. 1–11. URL: https://doi.org/10.1145/2508363.2508374.
20. Occupancy Networks: Learning 3D Reconstruction in Function Space / L. Mescheder et al. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, 15–20 June 2019. URL: https://doi.org/10.1109/cvpr.2019.00459.
21. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation / J. J. Park et al. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, 15–20 June 2019. URL: https://doi.org/10.1109/cvpr.2019.00025.
22. XU, Qiangeng, et al. DISN: Deep implicit surface network for high-quality single-view 3d reconstruction. *Advances in neural information processing systems*, 2019, 32.

REFERENCES

1. Frieder G., Gordon D. (1985). Back-to-Front Display of Voxel Based Objects. IEEE Computer Graphics and Applications, 5(1), 52–60. Retrieved from: https://doi.org/10.1109/mcg.1985.276273.
2. Mousavian R., Hossainali M. M., Lorenz C., Kunstmann H. (2022). Copula, a new approach for optimum design of Voxel-based GNSS tropospheric tomography based on the atmospheric dynamics. GPS Solutions, 26(4). Retrieved from: https://doi.org/10.1007/s10291-022-01340-1.
3. Dotremont K. (2023). Chapter 3 – From medical images to 3D model: processing and segmentation. Handbook of Surgical Planning and 3D Printing, 65–91. Retrieved from: https://doi.org/10.1016/b978-0-323-90850-4.00009-0.
4. Trujillo-Hernández G., Flores-Fuentes W., Rodríguez-Quiñonez J.C., Hernández-Balbuena D., Real-Moreno O., Miranda-Vega J.E., Bhateja V. (2022). 3D Reconstruction of Human Body Biometry. Optoelectronic Devices in Robotic Systems, 195–225. Retrieved from: https://doi.org/10.1007/978-3-031-09791-1_8.
5. Lee M.S., Kim J.H., Paeng J.C., Kang K.W., Jeong J.M., Lee D.S., Sung J. (2018). Whole-Body Voxel-Based Personalized Dosimetry: The Multiple Voxel S-Value Approach for Heterogeneous Media with Nonuniform Activity Distributions. Journal of Nuclear Medicine, 59(7), 1133–1139. Retrieved from: https://doi.org/10.2967/jnumed.117.201095.
6. You C., Zhou Y., Zhao R., Staib L., Duncan J.S. (2022). SimCVD: Simple Contrastive Voxel-Wise Representation Distillation for Semi-Supervised Medical Image Segmentation. IEEE Transactions on Medical Imaging, 41(9), 2228–2237. Retrieved from: https://doi.org/10.1109/tmi.2022.3161829.
7. Huanmei L. (2022). 3D Indoor Scene Reconstruction and Layout Based on Virtual Reality Technology and Few-Shot Learning. Computational Intelligence and Neuroscience. Retrieved from: https://doi.org/10.1155/2022/4134086.
8. Yang F., Li Y., Che M., Wang S., Wang Y., Zhang J., Cao X., Zhang C. (2022). The Polygonal 3D Layout Reconstruction of an Indoor Environment via Voxel-Based Room Segmentation and Space Partition. ISPRS Int. J. Geo-Inf., 11(10), 530. Retrieved from: https://doi.org/10.3390/ijgi11100530.
9. Lorensen W.E. (2020). History of the Marching Cubes Algorithm. IEEE Computer Graphics and Applications, 40(2), 8–15. Retrieved from: https://doi.org/10.1109/mcg.2020.2971284.
10. Ju T., Losasso F. (2002). Dual contouring of hermite data. ACM Transactions on Graphics (TOG), 21(3), 339–346. Retrieved from: https://doi.org/10.1145/566654.566586.
11. Frisken S. (1999). Constrained Elastic SurfaceNets: Generating Smooth Models from Binary Segmented Data. Retrieved from: https://doi.org/10.1007/bfb0056277.
12. Gaillard M., Miao C., Schnable J.C., Benes B. (2020). Voxel carving-based 3D reconstruction of sorghum identifies genetic determinants of light interception efficiency. Plant Direct, 4(10). Retrieved from: https://doi.org/10.1002/pld3.255.

13. Gutierrez A., Monaghan D., Jimenez M. J., O'Connor N.E. (2012). Persistent Homology for 3D Reconstruction Evaluation. Proceedings of the 4th international conference on Computational Topology in Image Context. Retrieved from: https://doi.org/10.1007/978-3-642-30238-1_15.
14. Laine S., Karras T. (2010). Efficient sparse voxel octrees. ACM SIGGRAPH symposium on Interactive 3D Graphics and Games, 55–63. Retrieved from: https://doi.org/10.1145/1730804.1730814.
15. McGraw T. (2020). High-quality real-time raycasting and raytracing of streamtubes with sparse voxel octrees. IEEE Visualization Conference (VIS). Retrieved from: https://doi.org/10.1109/vis47514.2020.00011.
16. Kobbelt L.P., Botsch M., Schwanecke U., Seidel H.P. (2001). Feature Sensitive Surface Extraction from Volume Data. Retrieved from: https://doi.org/10.1145/383259.383265.
17. Lobello R.U., Denis F., Dupont F. (2014). Adaptive surface extraction from anisotropic volumetric data: contouring on generalized octrees. Annals of Telecommunications, 69 (5-6), 331–343. Retrieved from: https://doi.org/10.1007/s12243-013-0369-4.
18. Ho C.C., Wu F.C., Chen B.Y., Chuang Y.Y., Ouhyoung M. (2005). Cubical Marching Squares: Adaptive Feature Preserving Surface Extraction from Volume Data. Computer Graphics Forum, 24(3), 537–545. Retrieved from: https://doi.org/10.1111/j.1467-8659.2005.00879.x.
19. Niessner M., Nießner M. (2013). Real-time 3D reconstruction at scale using voxel hashing. ACM Transactions on Graphics (TOG), 32(6), 169. Retrieved from: https://doi.org/10.1145/2508363.2508374.
20. Mescheder L., Oechsle M., Niessner M. (2019). Occupancy networks: Learning 3D reconstruction in function space. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 4460–4470. Retrieved from: https://doi.org/10.1109/cvpr.2019.00459.
21. Park J., Florence P., Straub J., Newcombe R., Lovegrove S., Fox D. (2019). DeepSDF: Learning continuous signed distance functions for shape representation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 165–174. Retrieved from: https://doi.org/10.1109/cvpr.2019.00025.
22. Xu, Q., Wang, W., Ceylan, D., Mech, R., & Neumann, U. (2019). DISN: Deep implicit surface network for high-quality single-view 3d reconstruction. Advances in neural information processing systems, 32.