



Monte Carlo-based 3D surface point cloud volume estimation by exploding local cubes faces

Nicola Covre¹, Alessandro Luchetti¹, Matteo Lancini², Simone Pasinetti², Enrico Bertolazzi¹, Mariolino De Cecco¹

¹ Department of Industrial Engineering at the University of Trento, Via Sommarive 9, 38123 Trento, Italy

² Department of Mechanic and Industrial Engineering at the University of Brescia, Via Branze 38, 25121 Brescia, Italy

ABSTRACT

This article proposes a state-of-the-art algorithm for estimating the 3D volume enclosed in a surface point cloud via a modified extension of the Monte Carlo integration approach. The algorithm consists of a pre-processing of the surface point cloud, a sequential generation of points managed by an affiliation criterion, and the final computation of the volume. The pre-processing phase allows a spatial re-orientation of the original point cloud, the evaluation of the homogeneity of its points distribution, and its enclosure inside a rectangular parallelepiped of known volume. The affiliation criterion using the explosion of cube faces is the core of the algorithm, handles the sequential generation of points, and proposes the effective extension of the traditional Monte Carlo method by introducing its applicability to the discrete domains. Finally, the final computation estimates the volume as a function of the total amount of generated points, the portion enclosed within the surface point cloud, and the parallelepiped volume. The developed method proves to be accurate with surface point clouds of both convex and concave solids reporting an average percentage error of less than 7 %. It also shows considerable versatility in handling clouds with sparse, homogeneous, and sometimes even missing points distributions. A performance analysis is presented by testing the algorithm on both surface point clouds obtained from meshes of virtual objects as well as from real objects reconstructed using reverse engineering techniques.

Section: RESEARCH PAPER

Keywords: Monte Carlo; volume estimation; affiliation criterion; cube explosion; point cloud

Citation: Nicola Covre, Alessandro Luchetti, Matteo Lancini, Simone Pasinetti, Enrico Bertolazzi, Mariolino De Cecco, Monte Carlo-based 3D surface point cloud volume estimation by exploding local cubes faces, Acta IMEKO, vol. 11, no. 2, article 32, June 2022, identifier: IMEKO-ACTA-11 (2022)-02-32

Section Editor: Francesco Lamonaca, University of Calabria, Italy

Received November 22, 2021; **In final form** February 25, 2022; **Published** June 2022

Copyright: This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Funding: This project has received funding from the European Union's Horizon 2020 research and innovation program, via an Open Call issued and executed under Project EUROBENCH (grant agreement N° 779963).

Corresponding author: Nicola Covre, e-mail: nicola.covre@unitn.it

1. INTRODUCTION

Estimating the volume enclosed in a three-dimensional (3D) surface point cloud is a widely explored topic in several scientific fields. With the increase of technologies for the virtual reconstruction of 3D environments and objects, many devices, such as Kinect, Lidar, Real sense, make it possible to acquire a depth image with increasing accuracy and resolution [1]-[4]. Several fields, such as mobile robotics [5], reverse prototyping [6], industrial automation, and land management, require accurate and efficient data processing to extract geometrical features from the real environment, such as distances, areas, and volume estimations. Among different geometric features, volume estimation has been presented as a challenging issue and

widely studied with different approaches in the literature. From the literature contributions on object volume estimation based on the 3D point cloud, Chang et al. [7] used the slice method and least-squares approach achieving high accuracy by investigating mainly known and homogeneous solids. The same 3D point cloud volume calculation based on the slice method was applied by Zhi et al. [8]. In general, the main limitation of using the sliding method for volume estimation is the dependence on the quality of the point cloud and the impossibility to work with complex shapes.

Bi et al. [10] and Xu et al. [11] estimated the canopy volume measurement by using only the simple Convex Hull algorithm [12] with the problem of volume overestimation in the case of concave surfaces. Lin et al. [13] improved the convex hull algorithm to handle concave polygons for the estimation of the

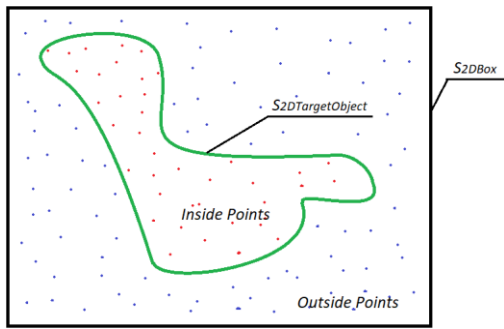


Figure 1. 2D example of Monte Carlo Integral approach - In green the geometric element under consideration ($S_{2DTargetObject}$), in black the 2D box (S_{2DBox}) of known area, in red the inside points, and blue the outside points.

tree crown volume, but their approach is still limited to providing a gross volume estimation that cannot be applied to complex objects with fine details.

Lee et al. [9] proposed a waste volume calculation using the triangular meshing method starting from the acquired point cloud. On one hand, this method results as accurate as the goodness of the acquired point cloud, on the other hand, it completely relies on mesh processing tools, such as MeshLab [14], and it cannot work if the 3D reconstructed object is an open domain.

We propose an innovative and competitive method to compute the volume of an object based on 3D point clouds via a modified extension of the Monte Carlo integration approach without the interpolation or the mesh reconstruction of the surface point cloud, it can handle homogeneous and non-homogeneous point cloud surfaces, complex and simple shapes as well as open and close domains.

1.1. The Monte Carlo Approach

Traditional methods for numerical integration on a volume, such as Riemann Integral [15] or Cavalieri-Simpson [16] partition the space into a dense grid, approximate the distribution in each cell with elements of known geometry and compute the overall volume by summing up all the contributions. In contrast, within a previously defined interval containing the distribution of interest, the Monte Carlo method generates random points with uniform distributions along the different dimensions to estimate the integral [17]. As shown in Figure 1 in the case of a 2D example we wish to calculate the area A_{target} of a closed surface. The geometric element under consideration (green line, $S_{2DTargetObject}$) is enclosed within a 2D box of known area A_{box} which encloses $S_{2DTargetObject}$. The total amount (N) of randomly generated points ($P_{generated}$) will fall inside S_{2DBox} . While some of them will fall outside $S_{2DTargetObject}$ (blue points), the others will fall inside (red points, $n_{InsidePoints}$). In the 2D case, $P_{generated}$ is identified with its 2D cartesian coordinates ($x_{P_{generated}}, y_{P_{generated}}$). An affiliation criterion (most often expressed by a simple mathematical equation) allows the identification and counting of points dropped in and out of the $S_{2DTargetObject}$.

In particular, these elements are regulated by the following expression:

$$A_{target} = \lim_{n \rightarrow \infty} \frac{n_{InsidePoints}}{N} \cdot A_{box} \quad (1)$$

1.2. 3D Extension of the Monte Carlo Approach

As reported by Newman et al. [18], this computational approach is particularly suitable for high-dimensional integrals.

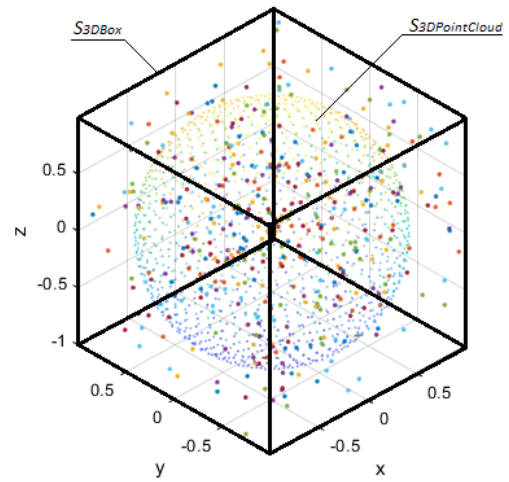


Figure 2. Extension of the Monte Carlo Integral approach to the point cloud of a 3D sphere ($S_{3DPointCloud}$) enclosed in a 3D box (S_{3DBox}).

For this reason, we extended the 2D Monte Carlo method described in the previous subsection to the calculation of the 3D volumes starting from their discrete surface's representation. Each point cloud can have, within a certain range, variable resolution, and spatial distribution homogeneity. In this case, N points are generated with uniform distribution along the three dimensions to estimate the volume of the unknown object ($V_{TargetObject}$) inside the prismatic element (S_{3DBox}). In the 3D case, $P_{generated}$ is identified with its 3D cartesian coordinates ($x_{P_{generated}}, y_{P_{generated}}, z_{P_{generated}}$). The $V_{TargetObject}$ is then calculated by counting the number of $P_{generated}$ that fell inside it ($n_{InsidePoints}$). Equation (1) becomes:

$$V_{targetObject} = \lim_{n \rightarrow \infty} \frac{n_{InsidePoints}}{N} \cdot V_{box} \quad (2)$$

Usually, having the target object represented by a continuous surface and described by a mathematical equation, as in the 2D case, the affiliation criterion is expressed by a continuous mathematical model. It is, therefore, easier to determine when a point falls within and without the $S_{3DTargetObject}$. However, the problem becomes more difficult when the $S_{3DTargetObject}$ is represented by the discrete distribution of some points lying on its surface ($S_{3DPointCloud}$), as can be shown in Figure 2. In this case, it is difficult to determine when a point falls within the $S_{3DPointCloud}$ or not. Moreover, in cases where the $S_{3DPointCloud}$ comes from a real acquisition, noise must also be taken into account. In fact, due to some acquisition errors not all the points of the cloud lie on the $S_{3DTargetObject}$.

This paper can be divided as follows:

- In the introduction we presented the problem of volume computation from point clouds, the state of the art, and our approach by describing the traditional Monte Carlo method, its 3D extension, and its limitations.
- In the following section, we describe our algorithm for volume estimation of point clouds based on the Monte Carlo approach.
- In the third section, we present the results obtained for the validation of the algorithm, testing it on both virtual and real objects.
- In the final section, we expose the drawn conclusions.

2. DEVELOPED ALGORITHM

The algorithm reported as pseudocode in Appendix A and explained here below takes the following parameters as input decided by the user:

- The point cloud acquired from the surface of an object whose volume is computed.
- The number of points N with which the Monte Carlo volume estimation must be performed.

The algorithm is composed of a pre-processing function and a classification function. The pre-processing of the point cloud checks its orientation, encloses the cloud surface in a box of known volume (S_{3DBox}), and performs a preliminary analysis of its distribution. The classification function is based on the "cube explosion" affiliation criterion, described in the following paragraphs.

2.1. Pre-processing

Given the total amount of points N , an efficient Monte Carlo approach should provide the smallest box which encloses the volume taken for the measurement. In fact, with a fixed N the bigger is S_{3DBox} the lower is the 3D point resolution and the worst is the accuracy of the Monte Carlo method. Hence the S_{3DBox} is defined by taking the minimum and maximum points in each one of the three principal directions x, y, z of the $S_{3DPointCloud}$. To further minimize the box dimensions a previous re-orientation of the $S_{3DPointCloud}$ is performed by applying the Principal Component Analysis (PCA). Once the S_{3DBox} is defined its volume is computed and used at the end of the affiliation criterion as V_{box} .

In the case of real objects, $S_{3DPointCloud}$ is collected by using depth cameras or other 3D scanners. The higher is the resolution of the tool, the denser is the point cloud. In the case of virtual objects, this point cloud is obtained by collecting the mesh nodes. The denser is the mesh, the more homogeneous is the point cloud distribution. However, the homogeneity of the point distribution is a factor that cannot be taken for granted. For this reason, a preliminary statistical analysis over the $S_{3DPointCloud}$ is carried out to obtain the parameters needed for the affiliation criterion, the core of the Monte Carlo method. In particular, the

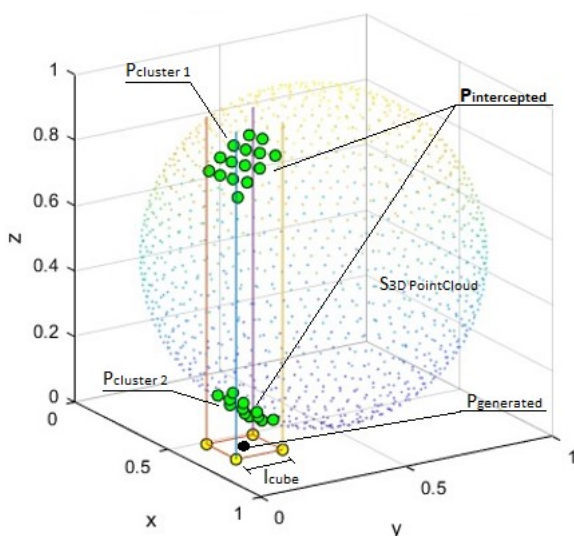


Figure 3. Example of explosion in the z -direction ($\eta = z$) of one cube's face from the position of a $P_{generated}$ - interception of 2 clusters of points (in green, $P_{intercepted}$).

parameters are obtained from the quantile distribution Q of the distances between each point and its closest neighbors.

2.2. Affiliation criterion using the "Explosion" of Cube Faces

The proposed affiliation criterion iteratively defines whether each $P_{generated}$ inside S_{3DBox} by the Monte Carlo method belongs to the external or internal domain of the $S_{3DPointCloud}$. The affiliation criterion that we have developed is based on the concept of "Explosions of Cube Faces". The idea is based on the generation of a cube of known edge (l_{cube}), around each $P_{generated}$, and iteratively extruding each one of its faces to determine how and how many times it encounters the $S_{3DPointCloud}$, Figure 3. Each one of the 6 face extrusions corresponds to a specific direction η along with one of the 3 main directions x, y, z and returns a binary judgment (J_η). J_η is 0 or 1 respectively if the point is supposed to be outside or inside the $S_{3DPointCloud}$. Eventually, by taking the mode of all J_η the final judgment (J) is assessed, and the point affiliation (internal or external) is defined. Each cube is oriented by using the same reference system of $S_{3DPointCloud}$. At each iteration, the procedure selects the direction η and extrudes the relative faces of the cube along their outgoing normal.

Initially, l_{cube} is determined by the following empirical equation:

$$l_{cube} = Q_{0.5} \cdot 3.5 \left(\frac{Q_{0.85}}{Q_{0.5}} - 1 \right) \quad (3)$$

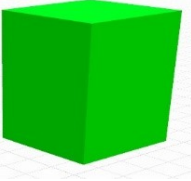
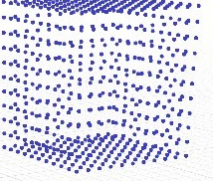
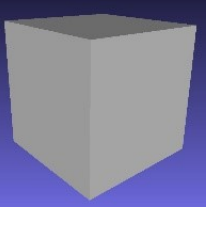
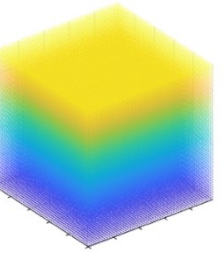
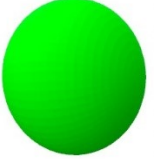
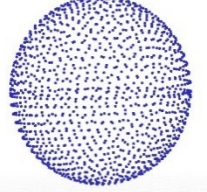
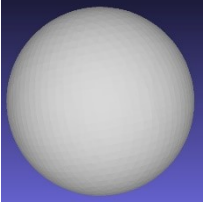
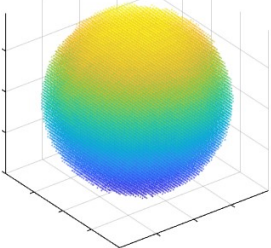



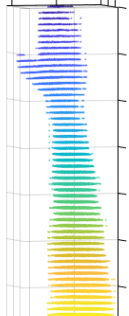



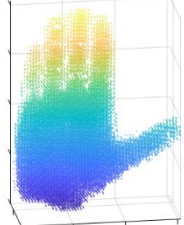
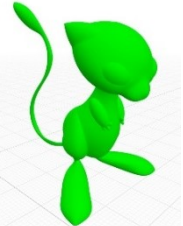
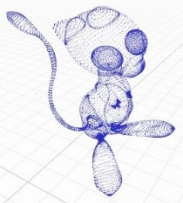
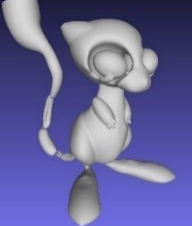
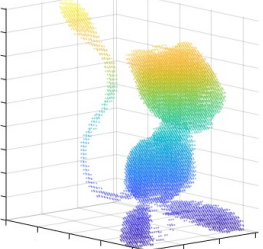
where $Q_{0.5}$ and $Q_{0.85}$ are the quantiles at 50 % and 85 % of the distribution of the distances between each point and its closest neighbors respectively. Equation (3) and the chosen quantiles are obtained by an empirical validation of the performances. In particular, the choice of $Q_{0.5}$ considers the median distance of two consecutive points and $Q_{0.85}$ highlights the $S_{3DPointCloud}$ sparse distribution and avoids initializing a small cube whose extruded faces pass through $S_{3DPointCloud}$ without touching its points.

Histograms of the distribution of the relative distances between each point and its closest neighbors for two different $S_{3DPointCloud}$ are shown in Figure 4. In particular, the first histogram is referred to as non-homogenous point cloud, specifically, Pokémon (Mew), while the second is referred to the homogeneous cloud of the geometric solid sphere, both reported in Table 1. On one hand from the first distribution is possible to observe that the ratio between $Q_{0.85}$ and $Q_{0.5}$ is 2.74, on the other hand, the quantile ratio of the second distribution is 1.01. The quantile ratio shows the proportion of the sparse portions of $S_{3DPointCloud}$ concerning the distribution of the average distances. In Mew's $S_{3DPointCloud}$ the ratio is higher because we have strong non-homogeneous distribution, such as a dense clustering of points for the eyes and sparse on the belly. In the sphere's $S_{3DPointCloud}$ the ratio is close to one as the difference between $Q_{0.85}$ and $Q_{0.5}$ is almost null due to the homogeneity of the point cloud distribution.

Each face extrusion may intercept a sub-portion of $S_{3DPointCloud}$ points ($P_{intercepted}$). If the total amount of intercepted points overcomes the threshold value ($th_{intercepted}$) of 3 points, l_{cube} is reduced of $l_{reduction}$ and the extrusion is repeated. The criterion for choosing $th_{intercepted}$ equal to 3 points depends on the clusterization checks introduced to strengthen the algorithm, as explained at the end of this section.

The value of $l_{reduction}$ has been empirically set equal to 10 % of the actualized value of l_{cube} as a compromise between final accuracy and computational time. The smaller $l_{reduction}$ is the higher the final accuracy is but the longer the final computational time.

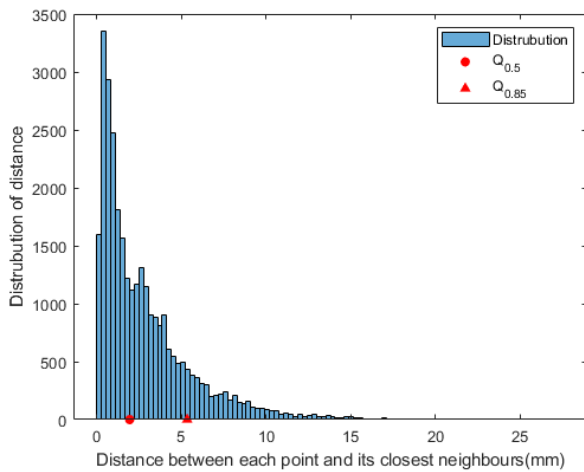
Table 1. Algorithms outputs with virtual objects.

Virtual Object Name	Original Virtual Model	Original Point Cloud ($S_{3DTargetObject}$)	MeshLab Reconstruction	Our Output
Cube				
Sphere				
Arm				
Hand				
Pokémon Mew				

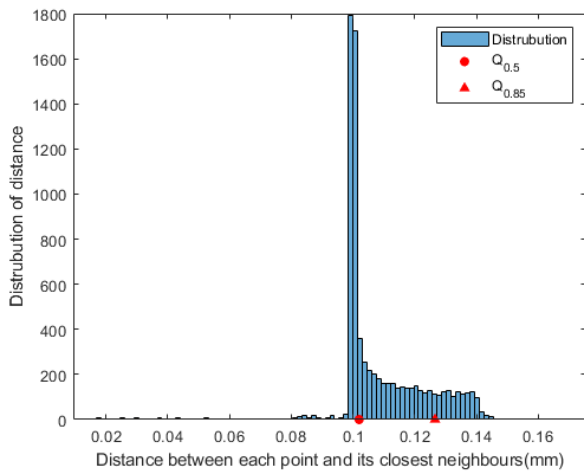
Following, a clustering algorithm subdivides the $P_{intercepted}$ into different clusters of points ($P_{cluster}$) along η . J_η assumes the values 1 or 0 if the total amount of clusters is, respectively, odd or even.

The clusterization is performed by re-ordering $P_{intercepted}$ along η . Computing the coordinate differences along η of each point from its consecutive, it is possible to obtain a sequence of distances. Ideally, considering $P_{cluster}$ orthogonal to η , this

difference within each $P_{cluster}$ should be null, as all the $P_{cluster}$ points lie on the same orthogonal plane. However, in real applications, several issues may occur, such as a slight inclination of $P_{cluster}$ concerning η or a random noise affecting the $P_{cluster}$ distribution along η . To overcome these problems, a threshold ($Th_{cluster}$) is set empirically equal to $Q_{0.5}$. Therefore, considering the $P_{intercepted}$ re-ordered along η , whenever a distance between



(A)



(B)

Figure 4. Distribution of distances between each point and its closest neighbors for (A) Pokemon Mew and (B) the sphere point clouds.

two consecutive points results to be less than $Tb_{cluster}$ the two points belong to the same $P_{cluster}$. Otherwise, a new $P_{cluster}$ begins. A further issue that affects the clusterization occurs when the extrusion intercepts tangentially $S_{3DPointCloud}$. In this case, a misleading clusterization is performed and further checks are needed to make the affiliation criterion more robust. In particular, this can be detected by performing an analysis of variance over each $P_{cluster}$. If the variance of one single $P_{cluster}$ is greater than $Tb_{cluster}$ the entire analysis along η is compromised and J_{η} needs to be discarded.

To make the affiliation criterion more robust a limitation on the maximum number of clusters encountered along each extrusion is also introduced. In particular, only those directions η are selected that have a total amount of clusters less than or equal to 1. This reduces the probability of encountering misleading clusters such as in the case of noisy point clouds acquired from real objects. The noise mostly appears as isolated points outside $S_{3DPointCloud}$, and rarely inside it. Furthermore, this allows justifying the choice of $tb_{intercepted}$ referring to the minimum number of points that define a plane.

3. RESULTS

This section reports the validation of the extended Monte Carlo algorithm on real and virtual objects considering different shapes. A total of nine objects, including regular geometric

solids, such as spheres or prisms as well as more complex shapes, such as human hands, arms, Pokemon (Mew), and the 3D scanning of ancient bronze statuettes of mythological figures were considered for this discussion. Point clouds of real objects were acquired from the real environment with an Azure Kinect ToF Camera and a Konica Minolta Vivid VI-9i 3D scanner for reverse engineering. The volumes used as a reference for the validation of the measures on virtual and real objects were respectively computed using the virtual mesh and the volume estimation by immersion in water [19].

The Monte Carlo algorithm accuracy increases with the total amount of points generated, as can be observed in Figure 5 and Figure 6. In particular Figure 6 reports the measurement of the mean and variance with a box plot of the relative error distribution as a function of the number of $P_{generated}$. As can be seen, the error is particularly high when few $P_{generated}$ are generated and gradually decreases as the number increases. For both virtual and real solids, depending on the resolution of the point cloud and the reported details, the asymptotic percentage error is below 7% computed with respect to the reference volume when the total amount of $P_{generated}$ is greater than 42875. On the contrary, with a low number of $P_{generated}$, the accuracy is low. It is also worth noting that the variance of the measures decreases as the number of $P_{generated}$ increases. This indicates that

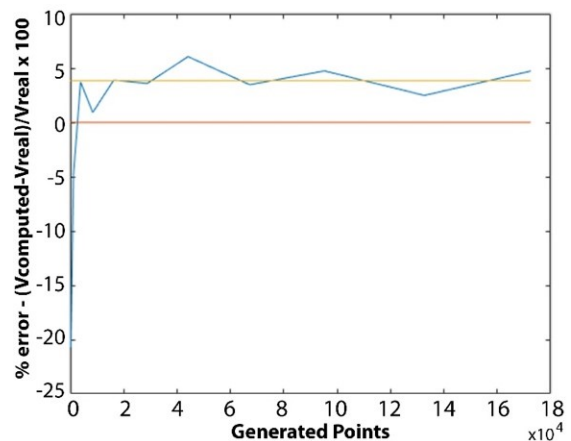


Figure 5. Error distribution of the Pokémon Mew volume estimation in % with respect to the increasing number of generated points with the Explosion Cubes Criterion.

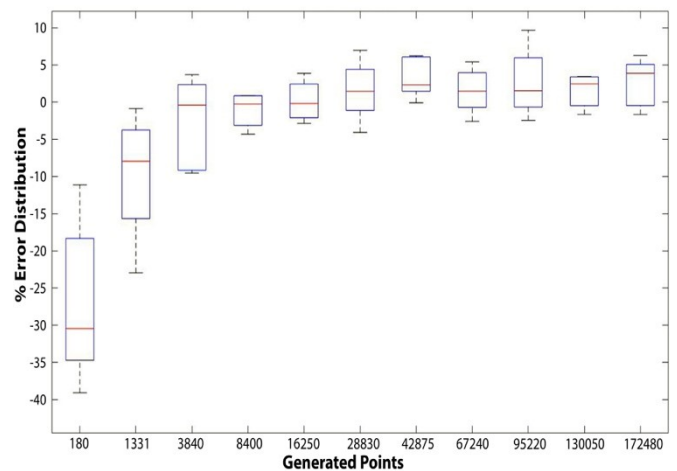

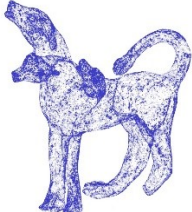
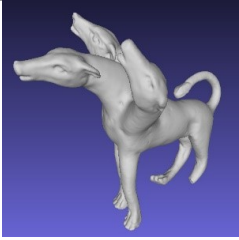
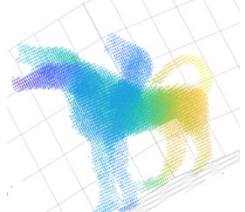
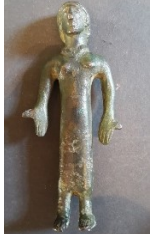
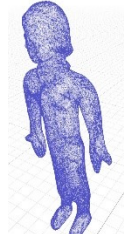

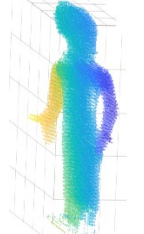



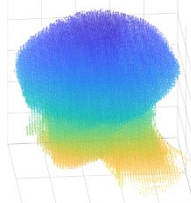

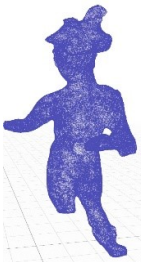

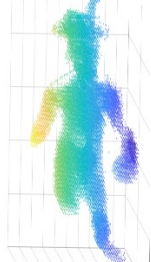


Figure 6. Box plot of the error distribution considering all the nine objects along with the increasing number of generated points with the Explosion Cubes criterion.

Table 2. Algorithms outputs with real objects.

Real Object Name	Real object	Original Point Cloud ($S_{3DTargetObject}$)	MeshLab Reconstruction	Our Output
Cerbero				
Ballerina				
Head				
Mercurio				

for all objects considered, even for convex, non-uniform, and folded point clouds, the relative error decreases and converges with the same trend.

As a general rule, on one hand, the performance of the algorithm can be driven by selecting the number of generated points. On the other hand, the more are the $P_{generated}$ the long is the computational time.

The average times spent by the proposed algorithm are shown in Figure 7. The tests were performed on a MacBook Pro 2 GHz Intel Core i5 quad-core 16 GB of RAM in MATLAB_R2020b environment. The average time grows as the number of points increases non-linearly as shown in Figure 7. However, with the same accuracy, the average computational time is less than or comparable to the time taken by the volume estimation methods reported in the literature.

Therefore, the user is allowed to choose the total amount of $P_{generated}$ with which Monte Carlo has to be executed as a trade-off between the level of desired accuracy, Figure 6, and the computational time, Figure 7. In addition, due to the asymptotic behavior of the error, the increase in performance becomes negligible after a threshold. For these reasons, it is convenient to choose a total amount of $P_{generated}$ just above the estimated volume that has stabilized (in our case 42875 $P_{generated}$).

However, the greater is the total amount of $P_{generated}$ the higher is the resolution of the point cloud generated by the Monte Carlo algorithm. This can be used as visual feedback to evaluate the goodness of the affiliation criterion and compare it with the

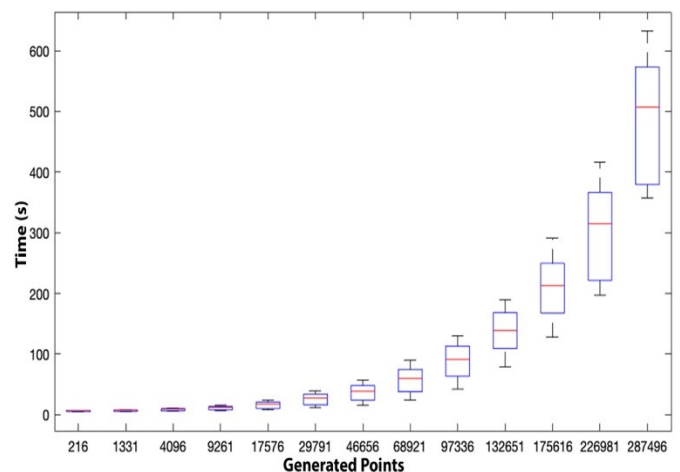


Figure 7. Computational times of the proposed algorithm changing the number of generated points ($P_{generated}$).

Table 3. Actual volume of the objects and its estimation with Meshlab and our method.

3D Object	Actual Volume in dm ³	MeshLab Volume in dm ³	Montecarlo Volume in dm ³	Monte Carlo error in %
Cube	1.00	1.00	0.99	0.10
Sphere	4.19	4.19	4.10	2.15
Arm	1.33	NA	1.39	4.51
Hand	0.412	NA	0.409	0.73
Pokemon Mew	0.539	NA	0.548	1.67
Cerbero	4.08	NA	4.18	2.45
Ballerina	1.24	NA	1.18	4.84
Head	4.54	4.52	4.50	0.88
Mercurio	4.53	4.57	4.66	2.82

algorithms that reconstruct the mesh. As shown in the last column of both tables Table 1 and Table 2, the representation of the inner points with the developed method returns a good representation of the original point cloud, $S_{3DPointCloud}$. On the contrary, mesh reconstructions are not always reliable. A mesh reconstruction using the default parameters of the Poisson's Reconstruction Method [8] in the MeshLab environment is shown in the fourth column of the same tables. As can be observed, good results are returned only for uniformly distributed point clouds and regular shapes, while the same cannot be assessed for complex shapes or non-uniform point clouds, such as Pokémon Mew and the Hand, with consequent negative repercussions on volumes calculation.

Another important aspect to consider when the volume is computed concerns the possibility to work with discontinuous and partially open surfaces, such as in Figure 8(A), where the acquired point cloud results to have huge discontinuities around the elbow. Most of the volume estimation algorithms based on the mesh reconstruction need manual fitting and adjustments to manage the missing clusters of points, otherwise, the measurement cannot be pursued. On the contrary, the proposed affiliation criterion for the Monte Carlo volume integration is robust to discontinuities due to the democratic judgment of the 6 cube faces. In fact, even if along a few directions the point cloud results to be open, and these faces extrusions will return $J_{\eta} = 0$, the final judgment J will still be equal to 1.

Table 3 shows the actual volume of the objects compared with that obtained with Meshlab, when possible, and our proposed method with its error percentage on the measurement. Nevertheless, given the choice of the external box and maintaining its ratio to the calculated volume, taken any object, its percentage error does not change by scaling its size. This means that the uncertainty on the measurement is proportional to the percentage error multiplied by the calculated volume.

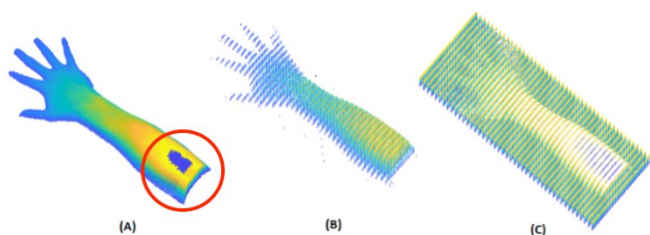


Figure 8. (A) Open Point cloud from a real acquisition of a Human arm, (B) Monte Carlo representation of inside generated Points, (C) Monte Carlo representation of outside generated Points.

4. CONCLUSIONS

This paper proposes an extension of the 3D Monte Carlo method for calculating the volumes of objects starting from their surface point clouds. The overall algorithm includes a pre-processing analysis that re-orient and evaluates the point cloud, an affiliation criterion based on the explosion of cube faces to discern the inner and the outer points of the Monte Carlo method, and a final volume estimate as described in Equation (2). As the reported results include also convex, complex, and folded surfaces, such as the Pokémon Mew or the Cerbero point cloud it was possible to show that the cube explosion affiliation criterion results be stable and reliable, returning consistent and repeatable measurements compared with gold-standard software for volume measurements, such as MeshLab.

The algorithm proves to be accurate with point clouds of different objects, both in terms of shape and distribution of points. The performances are then tested with the surface point clouds of 9 virtual and real objects, reporting an average percentage error on the tested samples lower than 7% with a computational amount of time of a few minutes depending on the desired accuracy.

REFERENCES

- [1] K. Khoshelham, S. Elberink, Accuracy and resolution of kinect depth data for indoor mapping applications, *Sensors*, Vol. 12, N. 2, 2012, pp. 1437-1454. DOI: [10.3390/s120201437](https://doi.org/10.3390/s120201437)
- [2] K. Khoshelham, Kourosh, Accuracy analysis of kinect depth data, *ISPRS workshop laser scanning*, Calgary, Canada, 29-31 August 2011, pp. 133-138. DOI: [10.5194/isprsarchives-XXXVIII-5-W12-133-2011](https://doi.org/10.5194/isprsarchives-XXXVIII-5-W12-133-2011)
- [3] J. Vaze, J. Teng, G. Spencer, Impact of DEM accuracy and resolution on topographic indices. *Environmental Modelling & Software*, Vol. 25, N. 10, 2010, pp. 1086-1098. DOI: [10.1016/j.envsoft.2010.03.014](https://doi.org/10.1016/j.envsoft.2010.03.014)
- [4] L. Keselman, J. Woodfill, A. Jepsen, A. Bhowmik, Intel realsense stereoscopic depth cameras. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, Honolulu, Hawaii, 21-26 July 2017, pp. 1267-1276. DOI: [10.1109/CVPRW.2017.167](https://doi.org/10.1109/CVPRW.2017.167)
- [5] D. Borrmann, A. Nüchter, M. Đakulović, I. Maurović, I. Petrović, D. Osmanković, J. Velagić. A mobile robot based system for fully automated thermal 3D mapping, *Advanced Engineering Informatics*, Vol. 28, N. 4, 2014, pp. 425-440. DOI: [10.1016/j.aei.2014.06.002](https://doi.org/10.1016/j.aei.2014.06.002)
- [6] D. Li, X. Feng, P. Liao, H. Ni, Y. Zhou, M. Huang, Z. Li Y. Zhu, 3D reverse modeling and rapid prototyping of complete denture. In *Frontier and future development of information technology in medicine and education*, Springer, Dordrecht, 2014, pp. 1919-1927. DOI: [10.1007/978-94-007-7618-0_226](https://doi.org/10.1007/978-94-007-7618-0_226)

- [7] W. Chang, C. Wu, Y. Tsai, W. Chiu, Object volume estimation based on 3d point cloud, 2017 International Automatic Control Conference (CACs), Pingtung, Taiwan, 12-15 November 2017, pp. 1-5.
DOI: [10.1109/CACS.2017.8284244](https://doi.org/10.1109/CACS.2017.8284244)
- [8] Y. Zhi, Y. Zhang, H. Chen, K. Yang, H. Xia, A method of 3d point cloud volume calculation based on slice method, International Conference on Intelligent Control and Computer Application (ICCA 2016), Atlantis Press, Zhengzhou, China, 19-17 January 2016, pp. 155-158.
DOI: [10.2991/icca-16.2016.35](https://doi.org/10.2991/icca-16.2016.35)
- [9] Y. Lee, S. Cho, J. Kang. A study on the waste volume calculation for efficient monitoring of the landfill facility. In Computer Applications for Database, Education, and Ubiquitous Computing, Springer, Berlin, Heidelberg, 2012, 978-3-642-35602-5, pp. 158-169.
- [10] Y. Bi, L. Qi, S. Chen, L. Li, S. Liu, Canopy volume measurement method based on point cloud data, Science & Technology Review, Beijing, China, Vol.31, No. 27, 2013, pp. 31-36. [In Chinese]
DOI: [10.3981/j.issn.1000-7857.2013.27.004](https://doi.org/10.3981/j.issn.1000-7857.2013.27.004)
- [11] W. Xu, Z. Feng, Z. Su, H. Xu, Y. Jiao, O. Deng, An automatic extraction algorithm for individual tree crown projection area and volume based on 3d point cloud data. Spectroscopy and Spectral Analysis, Vol. 34, No. 2, 2014, pp. 465-471.
DOI: [10.3964/j.issn.1000-0593\(2014\)02-0465-07](https://doi.org/10.3964/j.issn.1000-0593(2014)02-0465-07)
- [12] G. Klette, A recursive algorithm for calculating the relative convex hull, 25th International Conference of Image and Vision Computing, IEEE, New Zealand, 8-9 November 2010, pp. 1-7.
DOI: [10.1109/IVCNZ.2010.6148857](https://doi.org/10.1109/IVCNZ.2010.6148857)
- [13] W. Lin, Y. Meng, Z. Qiu, S. Zhang, J. Wu, Measurement and calculation of crown projection area and crown volume of individual trees based on 3d laser scanned pointcloud data, International Journal of Remote Sensing, Vol. 38, No. 4, 2017, pp. 1083-1100.
DOI: [10.1080/01431161.2016.1265690](https://doi.org/10.1080/01431161.2016.1265690)
- [14] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, G. Ranzuglia, Meshlab: an open-source mesh processing tool, In Eurographics Italian chapter conference, Salerno, Italy, 2008, pp. 129-136.
DOI: [10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136](https://doi.org/10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136)
- [15] R. McLeod. The generalized Riemann integral, Vol. 20. American Mathematical Soc., 1980.
- [16] M. Kiderlen, K. Petersen. The cavalieri estimator with unequal section spacing revisited. Image Analysis & Stereology, Vol. 36, No. 2, 2017, pp.133-139.
DOI: [10.5566/ias.1723](https://doi.org/10.5566/ias.1723)
- [17] W. Press, S. Teukolsky, W. Vetterling, B. Flannery. Numerical recipes: the art of scientific computing, Cambridge University Press, 1992.
- [18] M. Newman, G. Barkema. Monte Carlo methods in statistical physics chapter 1-4, Vol. 24, Oxford University Press: New York, USA, 1999.
- [19] D. M. K. S. Kaulesar Sukula, P. T. den Hoed, E. J. Johannes, R. van Dolder, E. Benda. Direct and indirect methods for the quantification of leg volume: comparison between water displacement volumetry, the disk model method and the frustum sign model method, using the correlation coefficient and the limits of agreement. Journal of biomedical engineering Vol.15, No. 6, 1993, pp. 477-480.
DOI: [10.1016/0141-5425\(93\)90062-4](https://doi.org/10.1016/0141-5425(93)90062-4)
- [20] M. Kazhdan, M. Bolitho, H. Hoppe, Poisson surface reconstruction, Proc. of the fourth Eurographics Symposium on Geometry processing, Vol. 7, 2006, pp. 61-70. Online [Accessed 21 April 2022]
<https://hhoppe.com/poissonrecon.pdf>

APPENDIX A

Algorithm 1 Pre-processing - Re-Orient the $S_{3DTargetObject}$ with PCA

Require: $S_{3DTargetObject}$
 $S_{3DTargetObject} \leftarrow$ coordinate matrix of the original point cloud $[X, Y, Z]_{N,3}$

- Compute Principal Component Analysis
- Compute the roto-translation matrix to orient the $S_{3DTargetObject}$ along the main main axes
- Orient the $S_{3DTargetObject}$ throughout matrix multiplication

Algorithm 2 Pre-processing - Cloud Analysis

Require: $S_{3DTargetObject}$, $n_{neighbours}$
 $S_{3DTargetObject} \leftarrow$ coordinate matrix of the original point cloud $[X, Y, Z]_{N,3}$
 $n_{neighbours} \leftarrow$ total amount of considered neighbors

- Compute the Matrix of Distances "D". The generic Element $D_{i,j}$ is the Euclidean distance of the point i from the point j
- \forall point select only the closest $n_{neighbours}$ neighbors, draw the histogram distribution and compute
 - $Q_{0.5}$ = average distance of \forall Cloud point from its neighbors
 - $Q_{0.85}$ = Quantile at the 85 % of the distribution

Algorithm 3 Pre-processing - S_{3DBox} Volume Computation

Require: $S_{3DTargetObject}$
 $S_{3DTargetObject} \leftarrow$ PointCloud matrix of coordinates $[X, Y, Z]_{N,3}$

- Select the maximum and minimum coordinate for each direction
 - $min_x = \min(X), max_x = \max(X)$
 - $min_y = \min(Y), max_y = \max(Y)$
 - $min_z = \min(Z), max_z = \max(Z)$
- Create a regular prism with 8 vertexes $[min_x, min_y, min_z], [min_x, min_y, max_z], etc...$ that encloses the whole point cloud (S_{3DBox})
- compute V_{box} equal to the Volume of S_{3DBox}

Algorithm 4 Cube Explosion Affiliation Criteria

Require: $S_{3DTargetObject}$, N
 $S_{3DTargetObject} \leftarrow$ coordinate matrix of the original point cloud $[X, Y, Z]_{N,3}$
 $th_{intercepted} \leftarrow 3$;
 $N \leftarrow$ the total amount of $P_{generated}$;
 $x_{P_{generated}} \leftarrow x_{min}$;
 $y_{P_{generated}} \leftarrow y_{min}$;
 $z_{P_{generated}} \leftarrow z_{min}$;
 $Points_{Inside} = Points_{Outside} = []$;
 $n_{InsidePoints} = n_{OutsidePoints} = 0$;
while $x_{P_{generated}} < x_{max}$ **do**
 while $y_{P_{generated}} < y_{max}$ **do**
 while $z_{P_{generated}} < z_{max}$ **do**
 Create a cube around $P_{generated}$;
 Initialize $l_{cube} = Q_{0.5} 3.5^{\left(\frac{Q_{0.85}}{Q_{0.5}} - 1\right)}$;
 for \forall direction x, y, z **do**
 $J_{\eta Array} =$ Empty Array;
 $\eta \leftarrow$ select the direction ;
 for positive and negative direction **do**
 Extrude the Cube face along η ;
 Intercept the $S_{3DTargetObject}$;
 $P_{intercepted} =$ intercepted points;
 $Tot =$ amount of $P_{intercepted}$;
 if $Tot > th_{intercepted}$ **then**
 $l_{cube} = 0.1 * l_{cube}$;
 Repeat the extrusion;
 else
 if $Tot < th_{intercepted}$ **then**
 Subdivide $P_{intercepted}$ in clusters along η ;
 Each cluster is $P_{cluster}$;
 $Tot_{clusters} =$ total amount of clusters;
 if $Tot_{clusters}$ is odd **then**
 $J_{\eta} \leftarrow$ inside ;
 else
 $J_{\eta} \leftarrow$ outside ;
 end if
 Save the J_{η} in $J_{\eta Array}$;
 end if
 end if
 end for
 $J = \text{mode}(J_{\eta Array})$;
 if $J == 1$ **then**
 Save $P_{generated}$ in $Points_{Inside}$;
 $n_{InsidePoints} += 1$;
 else
 Save $P_{generated}$ in $Points_{Outside}$;
 $n_{OutsidePoints} += 1$;
 end if
 $z_{P_{generated}} += N^{\frac{1}{3}} - 1$
 end while
 $y_{P_{generated}} += N^{\frac{1}{3}} - 1$
 end while
 $x_{P_{generated}} += N^{\frac{1}{3}} - 1$
end while

Algorithm 5 Final Volume Computation

Require: InsidePoints, OutsidePoints, V_{box}
 $FinalVolume = \frac{Total_{Inside}}{Total_{Inside} + Total_{Outside}} * V_{Box}$
