

Color-based 3D Point Cloud Reduction

Mohamed Lamine Tazir, Paul Checchin and Laurent Trassoudaine

Institut Pascal, Université Blaise Pascal, UMR 6602, CNRS, 63177 AUBIERE, FRANCE

Email: tazir.med@gmail.com, paul.checchin@univ-bpclermont.fr, laurent.trassoudaine@univ-bpclermont.fr

Abstract—Significant advances have recently been obtained in 3D sensor domain, such as 3D lasers and the Microsoft Kinect, yielding synchronized depth and color information. Nowadays, these sensors are able to collect very dense point clouds. However the interest of such an amount of information is not always justified. The point cloud sampling procedure is an important processing step. It affects the relevance and the accuracy of the remaining steps of this processing chain. In this paper, we present a novel RGB-D down-sampling method. The proposed approach is based on the use of both color information and geometry of the scene. First, a voxelization is performed to maintain the topological details of the scene, then for each voxel, a color based classification of its points is done. Thereafter, only one point of each color class is maintained and all the remaining points are removed. We evaluate the method on the RGB-D data taken from 3D laser scanner and compare it to the method implemented in PCL. The results show that the new method gives better results than the state-of-the-art method. In addition, it opens interesting perspectives to merge color and geometric information.

I. INTRODUCTION

From past decades, 3D scanners have emerged and evolved as powerful tools which have been applied in various fields. This growth was driven by the need for accurate, efficient, safe and fast modeling enabling a comprehensive analysis of the scene [1]. Professionals from all sectors who usually worked with two-dimensional plans and schematic representations (robotics, aerospace, automotive, architecture, naval and etc.) now have the opportunity to give a new dimension to their tools. Several objectives can be at the origin of a three-dimensional modeling. It may be, as in this study, the three-dimensional representation of a ship hull, which allows an operator to define the areas to be treated and provide its coordinates to the robot. Global project consists in the development of robotic systems for the surface maintenance of ship hulls through autonomous mobile robots. In this project, the surveys of discrete three-dimensional measurements are accompanied by related images. Thus allowing areas affected by corrosion to be detected and interactively suggested to an operator for approval. Such surface defect detection is mainly done based on the color information present in 3D points.

The use of color in the fields of computer vision and robotics perception provides a rich and quality information. Undoubtedly, this quality of information has stimulated the performance of variety of applications in these areas.

For the purpose of simplifying the amount of 3D data, the traditional digital chain can be synthesized in three successive stages [2]: 3D acquisition, sampling and mesh generation. Figure 1 shows this process. Simplification is the process of reducing number of data points. In this paper, it is colored 3D point cloud.



Fig. 1: 3D data processing chain

With the high-speed evolution of RGB-D sensors, it is now possible to obtain very dense point clouds [3]. A sensor like Kinect produces a cloud with more than 300 000 (640×480) points, which represents a lot of information. Furthermore, 3D laser scanners provide higher resolution (around two orders of magnitude) [4].

As point cloud processing is directly proportional to number of available points, in terms of processing time and data storage, it is rather disadvantageous to have more points than required [5]. Because these additional points slow down the processing time and need data storage space, without giving any more additional information [1]. To the best of our knowledge, to perform a simple operation on every point of a cloud, it would be of $\mathcal{O}(n)$, n being the number of points. To compare every point with its k nearest neighbors, it would be $\mathcal{O}(nk)$. It is clear now why some algorithms processing the 3D point clouds take several seconds to run. Current scanner devices tend to provide a lot of redundant information (the region of interest represents a small percentage of acquired cloud). Thus, expensive pre-processing steps are required to reduce the number of points by selecting only the relevant ones. The successive steps leading to the mesh model tend naturally to decrease in number of points, because mesh operations are quite complex and computationally expensive [6].

The objective of our approach is to reduce the number of 3D points while preserving the amount of information carried by these points. In the remainder of this paper, a set of state-of-the-art point cloud reduction methods will be presented in Section II. The details of the proposed method with algorithms and data structures will be introduced in Section III. Finally, the experimental results accompanied by a comparison with a state-of-the-art method will be described in Section IV, just before our conclusion which constitutes the Section V.

II. RELATED WORK

Today's scanners are able to achieve millimeter accuracy with a high resolution [5], supplying the user with a huge amount of data e.g., a scan of an object of few centimeters can produce hundreds of thousands of points. Therefore, it is essential to simplify this redundant information to make it tractable for a standard computer [3]. Reducing the number of points may be performed during the measurement process by setting a suitable scanning resolution or by limiting the

scanning angles [1]. In this research, we are rather interested in reducing the point cloud after the measurement instead of limiting it during the acquisition time, and so acquiring as much information as possible, in order to detect the defects more precisely. Given the amount of work on this topic, we do not present an exhaustive survey. However, we will cover an overview of the subject where the main approaches are described.

A large number of pre-processing approaches aim to reduce the density of the cloud, by removing all points that are unnecessary. Within these approaches, there are several ways to achieve this, such as trim all outliers from the cloud, then perform segmentation to extract only the parts of interest. A common operation is also sampling, which returns an equivalent cloud to the original one, with fewer points. This equivalency was defined in [1] such that the data in the sampled cloud have similar properties (e.g., color, intensity, position, etc.) than those in the original one, while the sampling ratio is reduced as long as possible.

It is conventionally accepted that the point cloud reduction process is based on three main groups as shown in Fig. 2. The simplification by partitioning is done via voxelization [5]. First, the cloud is divided in multiple cube-shaped regions with the desired resolution. Then, all points within every voxel are processed such that one remains. The simplest way would be to randomly select one of them or the voxel center, but a more accurate approach would be to compute the centroid.

As a variation of this technique, Lee et al. [7] propose a point reduction method using 3D grids. The method compartmentalizes the points by a spatial decomposition process based on the principle of octrees. This spatial decomposition is performed using the standard deviation of the normal vectors estimated from points within each cell. Therefore, the cell size corresponds to the curvature generated by the points of each cell. The reduction is then obtained by selecting a representative point in each cell.

The authors of [8] present a point simplification algorithm using local coplanar analysis on the basis of an octree data structure. This algorithm performs a recursive and hierarchical partition of the point cloud. This subdivision is stopped when the points of each division form a single plane. This ensures a dynamic subdivision based on variations of cloud surfaces. The nearest point of the centroid of each voxel is chosen as a representative point in the simplification step. However, this method deforms considerably the shape of the model which contains many flat surfaces, where a large number of points are reduced to a single point.

In the work presented in [9], the point cloud reduction is not the principal purpose of this research, but it is an important step of the process which concerns the development of mobile system for 3D environmental data representation. The authors adopt the voxel-based simplification, with a centroid representative of original 3D point within each voxel.

Reduction points via voxelization method is simple, fast and memory-efficient [5]. Nevertheless, these methods still have some drawbacks which can be summarized by their

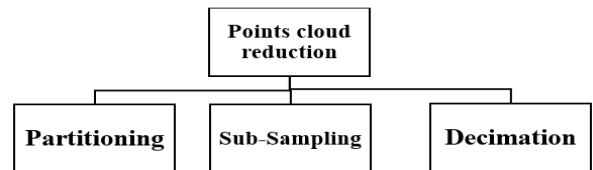


Fig. 2: Classification of 3D point cloud reduction methods

difficulty in controlling the partitioning surface resolution, and by the large topological errors that appear when the size of the voxel increases. The second group of point cloud reduction techniques is subsampling which may be implemented as a pre-calculated binary mask applied to the point cloud. As a general rule, the subsampling can be applied in a random way by just fixing the reduction rate [10] or with a uniform selection strategy [11], [12]. Boissonnat and Cazals [13] present a data reduction algorithm that randomly calculates a subset of points and builds 3D Delaunay triangulation. This algorithm adopts a costly pre-processing to preserve the original surface data before simplifying the point set. Alexa et al. [14] propose tools to increase or decrease the density of the points, thus, allowing an adjustment of the spacing among the points to control the fidelity of the representation.

The work introduced in [1] focuses on subsampling methods and how they impact the information carried by the points before the reduction. It provides two methods to reduce the point cloud data while maintaining the maximum content of the original cloud. These methods are called level histogram sampling and inversely weighted distance sampling. They use the information of the distance distribution in the cloud to carry out the reduction. These sampling methods are tested in three separate cases in which data are collected through terrestrial or mobile laser scanning system. Two reference methods (uniform and linear point picking sampling) are used to compare the results. The performance of the method is evaluated using two criteria. The first criterion is to preserve information carried by the original cloud. The second criterion is the change in the total processing time including the time spent for sampling.

The principle of decimation algorithms [3], [5] lies in the definition of an importance function for each point of the cloud. For this, several metrics are used such as those based on neighborhood, position or normal. Once this function is defined, points are sorted according to their importance in a priority queue. Then, the points that have minimal metric function are removed. For the points in the vicinity of deleted points, the importance function must be recalculated and the queue is updated. This process is repeated with every point of the cloud until the satisfaction of the stop condition that can be a number of target point or a maximum allowable error. Algorithm 1 summarizes the principle of decimation.

The first algorithm which uses decimation was proposed by the authors of [15]. It exploits the Voronoi diagram to detect the redundancy in the input point cloud and therefore the points to be deleted. An input parameter ρ is also used to determine the level of decimation. However, this method requires many computations [8].

Another paper that addresses the point cloud reduction by decimation is proposed by the authors of [2]. It is based on the Discrete Shape Operator (DSO) to classify all the points in a propriety queue. The DSO attributes high values for the feature points and a small values for the other points which present small variation. The simplifying process carries on the points with a small value in the priority queue. This procedure is achieved in two steps. The first one consists in detecting the neighboring points of each point: the distance to the nearest neighbor is multiplied by a constant, and the sphere with this value as radius is considered. Thus all points within this area are taken as neighbors of the given point. The second step is the computation of normal at each point. For that, the best-fit plane of neighboring points is estimated using the least square method. Thereafter, the method uses the normal vector and neighbors points to estimate a discrete shape operator DSO of each point. This latter is considered as the importance function of the decimation process. The DSO is calculated by combining two quantified functions, one to estimate the curvature and the second for the torsion. The curvature denotes the variation of the tangent vectors, and the torsion denotes the variation in the binormal vectors. The root of the sum of squares of the two functions denotes the degree of local surface variation.

Song and Feng [3] propose a similar algorithm for the decimation of mechanical parts with edges and corners. The basic idea is to keep the points pertaining to the edges and corners in the final reduction as they help to define geometrical shapes, while points pertaining to smooth surfaces are largely removed. In the first step, the algorithm detects points on or close to sharp edges and corners using the deviation of normal vectors of the neighboring 3D points. The second step consists in the simplification process. It is achieved by progressively removing points in smooth regions. The quantification of a points importance is based on points in its neighborhood and corresponds to the points contribution to the representation of local surface geometry. The algorithm calculates the average of the Euclidian distances from the considered point to the estimated tangent plane represented by its neighbors. The idea is to evaluate whether or not the local surface geometry of this point can be reliably implied from its neighboring points. A big value indicates that the position of this point cannot be reliably represented by its neighboring points. Therefore, it plays an important role in the definition of the local geometry. A small value indicates that even without this point the local form derived from its neighbors would not change much; therefore, it is not so important and is likely to be removed. The problems that arise here are how to select neighboring point, and how many points should be selected. Both steps of this algorithm require to establish neighborhoods for each data point and estimate its normal vector. This is very costly in computing time. Other flaw is that at each iteration, one point is deleted. Furthermore, for each point removed, the algorithm recalculates the importance value of all points in the vicinity. Finally, due to the exigency of maintaining all the edges and corner points, the number of points in the simplified point

Algorithm 1 Reduction by decimation

- 1: Sort the points according to their **importance** in a priority queue $\mathbf{F}$
 - 2: Remove the head point $\mathbf{F}$ (minimum error)
 - 3: **Optimisation:** recalculate the importance of the adjacent points of the deleted point
 - 4: Update $\mathbf{F}$
 - 5: Goto 1 until the stop condition
 - Number of target points
 - Maximum allowable error
-

cloud cannot be less than the total number of these points, this may cause problems for objects that have much textures.

In 2012, Li et al. [5] introduced a decimation algorithm with feature preservation. This algorithm extracts the feature points because they have a high importance since they describe the basic shape of the object. These points are positioned at convex or concave boundaries and sharp regions. In order to keep these points, the authors establish a curvature threshold defined by the sum of the distances from the k nearest neighbors of each point to the tangent plane at each point. For the rest of the points, the algorithm covers them by spheres with radius adapted to the density of the neighborhood. For each sphere, an optimized point is determined. It replaces all the points of the sphere.

As the new 3D measurement tools provide highly accurate data and can capture any surface variation, any reduction can lose textural details. Despite all the research presented above, it remains an interesting question about what could be an appropriate RGB-D reduction (based position and color) and how to use these information to reduce the size of raw clouds. Li and his colleagues in their paper [5] report that to ensure the quality of simplification the geometrical characteristics must be taken into account. We propose here to add the color feature in the reduction criterion so that the cloud retains the full of its original information. Our second contribution lies in the ability of our algorithm to process point clouds that can go up to several tens of millions of points. All these previously presented research test their algorithms on small clouds of points that do not exceed one million points. Major reduction rates can be achieved with our method, which can exceed the rate of 90 % of reduction while preserving the amount of information carried by the original cloud, which constitutes our latest contribution.

III. THE PROPOSED APPROACH

The information carried in the 3D point contains the position information, color and reflectance. In this paper, we show that reducing the huge amount of raw data points can be done without losing the essential information and its initial accuracy. The proposed approach is based on two different RGB-D cloud features: color and 3D location. Figure 3 illustrates the work flow of the proposed RGB-D down-sampling method that consists of three sub-tasks: (1) voxelization, (2) clustering and (3) subsampling. The first task performs a spatial grouping which attempts to preserve the topological information based on the 3D position of the points. A set of 3D cubic regions (voxels) is generated where all points within the voxel have

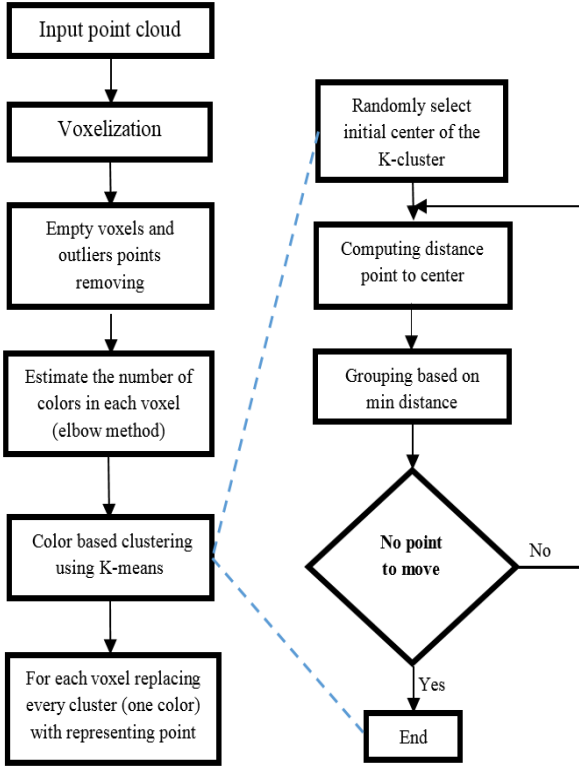


Fig. 3: The flowchart of the approach

very close spatial positions. The second task bundles all points of each voxel based on color. Once this grouping is done, we perform the last task which keeps one point for each color (cluster) in the voxel. This point is selected in function of the cluster centroid. This allow to have a much smaller representation while preserving spatial and color information of the original cloud. A detailed description of these three tasks is presented next.

A. Voxelization

At the beginning, the procedure applies a bounding box to the entire point cloud by finding the minimum and maximum positions of points along the three axes X , Y and Z . The number of voxels for this bounding box is determined by the voxel size. Figure 4 shows the output of this processing.

1) *Voxel assignment*: each voxel is identified by a unique linear index. If i , j , k represent the voxel indices in the X , Y , Z dimensions respectively, numDivX , numDivY are the number of voxels along X and Y axes, the formula to encode the linear index taken from [16] is:

$$\text{idx} = i + j * \text{numDivX} + k * \text{numDivX} * \text{numDivY} \quad (1)$$

According to this equation we assign an index idx to each point. This relationship allows direct access to the desired voxel, avoiding to browse the set of all voxels [4].

2) *Voxels suppression*: as the shape of the point cloud is arbitrary (see Fig. 4), the step of delimiting points by bounding box creates many empty voxels. All empty voxels will be removed and only voxels which contain points will remain.

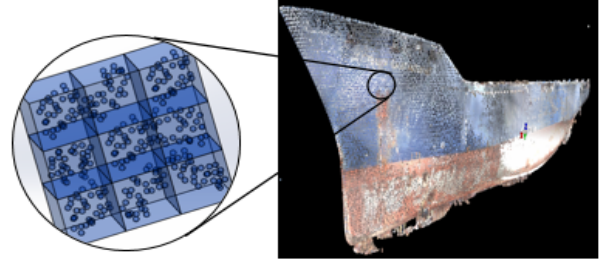


Fig. 4: Voxelisation of the ship hull. The topological features are retained by choosing a small voxel size

This step will also allow to remove noise and outliers which are easily detected at this stage: all voxels containing a number of points below than a threshold and having empty adjacent voxels are considered as voxels containing noise or outliers, and therefore, are deleted.

B. Clustering

Clustering can be defined as the task of automatically identifying groups of similar characteristics from a given set of data points [17]. In our case, this process is performed on points of each voxel in order to group those that have the same colors. This step is carried out using one of the most popular clustering techniques, which is the k-means algorithm [17], [18]. The fundamental principle of this method is based on minimizing the intra-cluster distance (the distance between points within each partition) and maximize inter-cluster distance (the spacing between the groups). This idea is formulated by the discovery of the following parameters: $\Omega_k = \{\mu_j\}_{j=1 \dots K}$ and the labels Γ , such that the function below is minimized:

$$\mathcal{L} = \sum_{i=1}^N \sum_{j=1}^K 1[\gamma_i = j] \|x_i - \mu_j\|_2^2 \quad (2)$$

$\mathcal{L}$ is the fitness function of the classification method k-means, where μ_j is the set of centroid of each class $j = 1 \dots K$. $1[\cdot]$ is an indicator function of the associated condition and $\|\cdot\|_2$ is the $L2$ norm or the Euclidean distance and N represents the number of points.

In order to cluster with the k-means method, the objective in (2) is evaluated iteratively until some convergence criteria are satisfied. Each iteration consists in assigning each point to the closest partition:

$$\gamma_i = \underset{j=1 \dots k}{\operatorname{argmin}} \|x_i - \mu_j\|_2^2, i = 1 \dots N \quad (3)$$

and in updating the mean (center) of each cluster μ_j :

$$\mu_j = \frac{\sum_{i=1}^N 1[\gamma_i = j] x_i}{\sum_{i=1}^N 1[\gamma_i = j]} \quad (4)$$

The k-means method starts by setting initial values of the positions of partitions centers Ω_k . These centers are set randomly. However, random initialization often generates a near-optimal solution because it cannot always guarantee the convergence to the global minimum. The convergence criteria

applied in this method consists in setting a maximum number of iterations or when two successive iterations lead to the same partition.

1) *Number of colors*: the k-means clustering method is an extremely simple and efficient method. However, it assumes prior knowledge of the data to choose the appropriate number of clusters (K). The correct choice of K is often ambiguous, so we use the Elbow method to find it [19] (see Algorithm 2). It consists in calculating $\mathcal{L}$ for different values of K .

The idea of the Elbow method for choosing the appropriate number of clusters is to compare the cost function $\mathcal{L}$ for different number of cluster solutions. The curve of $\mathcal{L}$ in function of the number of clusters shows that the error decreases as the extent as K becomes larger. Indeed, when the number of groups increases, the sizes of these groups become smaller, which also lead to distortion also be smaller. The appropriate cluster solution could be defined as the solution at which $\mathcal{L}$ decreases sharply. This produces an “Elbow effect” in the graph, as shown in Fig. 5 where the Elbow point is marked in green.

Algorithm 2 Elbow method

```

1: Cloud (X):
2: Let  $N$  = number of points in  $X$ 
3: Init a list  $D$ , of size  $N/2$ 
4: Let  $D[0] = 0$ 
5: for  $k = 1 \dots N/2$  do
6:   cluster  $X$  with  $k$  clusters (e.g. with k-means)
7:    $D[k] = \mathcal{L}(k)$ 
8: Define  $J$  of size  $(N/2) - 1$ , with  $J(0) = 0$ 
9: for  $i = 1 \dots N/2 - 2$  do
10:   $J(i) = D[i] - D[i + 1]$ 
11: Return the  $k$  between 1 and  $N/2$  that maximizes  $J$ 

```

C. Subsampling

When the clustering method is applied on points within each voxel, it generates several groups of color, according to the color variation in this voxel. The points in each group are considered unnecessary points insofar as they belong to the same voxel and have the same color. Therefore, keeping a single representative point for each class is sufficient. In our case, we keep the centroid of each color class. Figure 6 illustrates this process. This process of removing unnecessary points is called subsampling.

IV. RESULTS

In this section, we evaluate our method with a series of experiments. The results presented were obtained using a C++ implementation of the method under Linux Ubuntu 14.04 with an Intel Core i7-4800MQ, 2.7 GHz with 32GB of RAM. To the best of our knowledge, no other state-of-the-art method considers the color information in the reduction process. This is why it is practically impossible to evaluate and compare in some ways, fairly, our method with existing works. However, we provide a comparison with a known sampling method (PCL voxel grid) [20] which proceed only on 3D point clouds.

During the implementation of the method, we tried to be as simple as possible while maintaining a high efficiency.

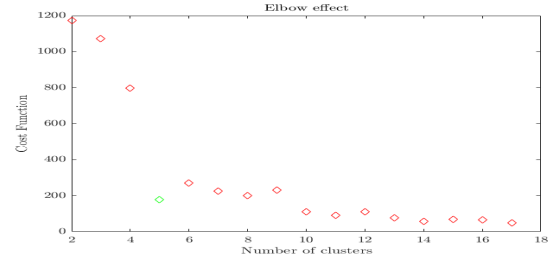
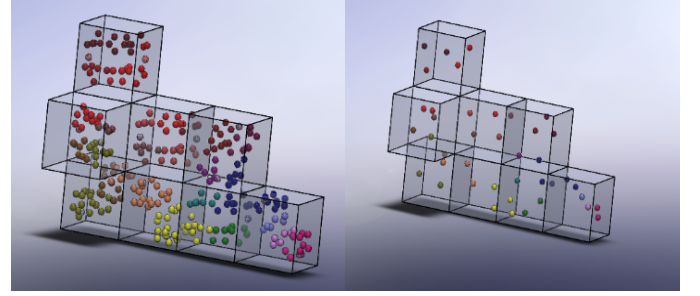


Fig. 5: Elbow method: in this graph example, the Elbow point is marked in green



(a) Point cloud before reduction (b) Point cloud after reduction

Fig. 6: Color-based point cloud reduction. The point cloud is voxelized and, in each voxel, a color-based clustering of points is performed. The centroid of each clusters is then chosen as the representative point and the remaining points are removed

This is the reason for the existence of a single parameter that adjusts all the algorithm. This parameter represents the voxel size. When increasing the size of the voxels, we increase the number of points enclosed within this voxel, which leads to a larger reduction rate. The efficiency is highlighted by a large rate of point reduction while preserving the maximum of original information. Figure 7 shows four point clouds used for the evaluation of the method. Because of the huge amount of points collected over the entire hull (almost 1 trillion points), making it very complicated to process, we limited our evaluation to clouds having between 2 and 8 million points. Table I presents the results of the reduction step. For each cloud, we indicate the original number of points and the total number of voxels. The reduction rate (ratio between the size of the original cloud and the size of the cloud after reduction) is then presented. Computation time of reduction process is also provided. The columns refer to the time taken by the steps that follow: (1: voxelization) (2: determining the number of colors in each voxel) (3: grouping with k-means using the right number of clusters). As expected, the step for detecting the number of colors in each voxel is the most costly part. This step, which is directly proportional to the number of points in each voxel, will benefit from parallelized implementation in coming versions of the algorithm. The computation time of this step may seem high, but it should be noted that for each n points of each voxel, the algorithm runs $n/2$ times the clustering process with k-means, and each time it calculates the cost function (distance between each point and its center).

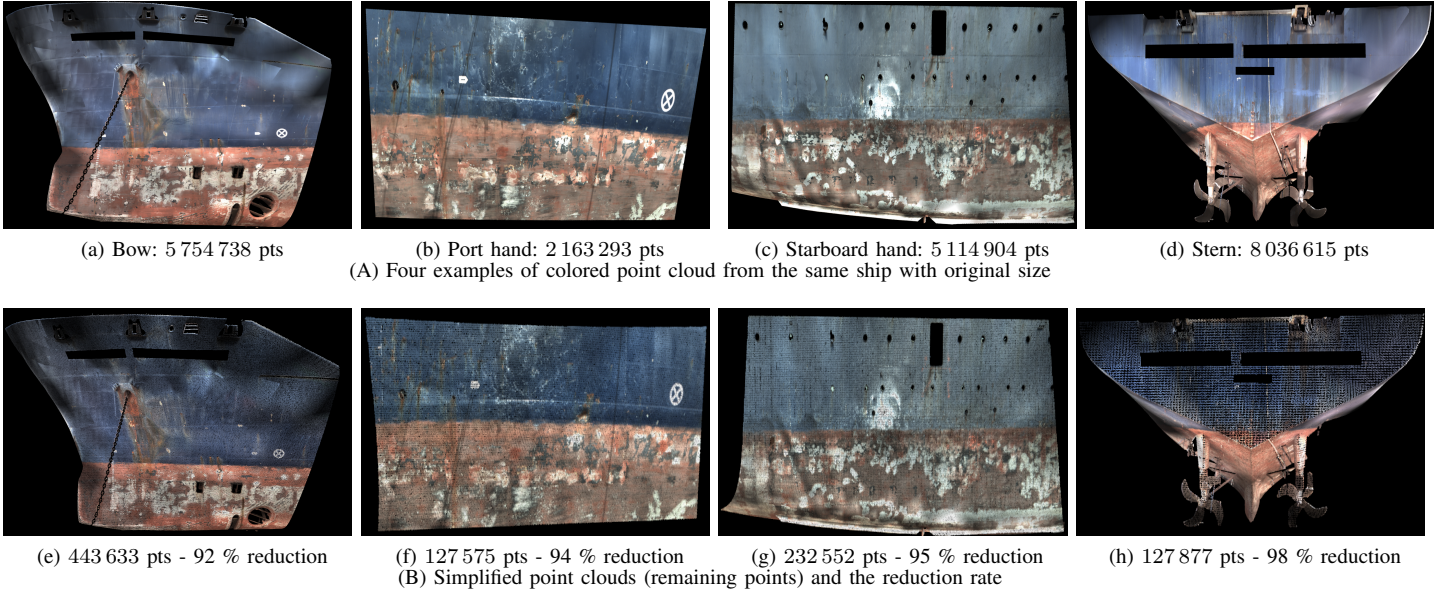


Fig. 7: Four examples of colored point clouds with different texturing results

TABLE I: Simplification results on ship hull datasets

Samples	Original clouds	Simplified clouds				Processing time			
	Number of points	Number of points	Simplification rate	Number of voxels	Average of points/voxels	1 (s)	2 (s)	3 (s)	Total (s)
Port hand	2 163 293 pts	127 575 pts	94 %	38 508	56 pts	0.106	12.089	0.002	12.197
Starboard hand	5 114 904 pts	346 944 pts	93 %	113 203	45 pts	0.270	47.806	0.001	48.077
Bow	5 754 738 pts	292 601 pts	95 %	88 102	65 pts	0.288	46.484	0.001	46.773
Stern	8 036 615 pts	373 519 pts	96 %	118 063	68 pts	0.385	179.038	0.001	179.424

The number of voxels and the average number of points per voxel may be the two other indicators of the time consumed in this step. The growth of either of these two leads to an increase in the computation time. This can be observed in the comparison of point clouds “Starboard hand” and “Stern” and between “Bow” and “Stern”. The comparison “Starboard hand” and “Stern” shows the effect caused by the increase in the average number of points per voxel. The two clouds have a comparable number of voxels (113 203 and 118 063), but the average number of points per voxel is different (45, 68). This difference induces a temporal score of 48 s and 179 s respectively. Similarly, the temporal effect caused by the increase in the number of voxels is shown by comparing “Bow” and “Stern”. The difference between these two clouds, that have both an average of 65 and 68 points per voxel, is in their total number of voxels, which is respectively 88 102 and 118 063. Calculation times are here 46 s and 179 s.

Figure 8 illustrates how the voxel size impacts the computation time to detect the required cluster number for the same point cloud “Bow” of 5 754 738 points. The corresponding clouds to this transition are given in Fig. 9. Evaluation of the results of a reduction process is considered as a challenge for the scientific community, as Zhang et al. assert in their paper [21] published in 2012. Indeed, almost the majority of research articles does not provide metrics to evaluate their results. They are based on the visual aspect to assess it. In

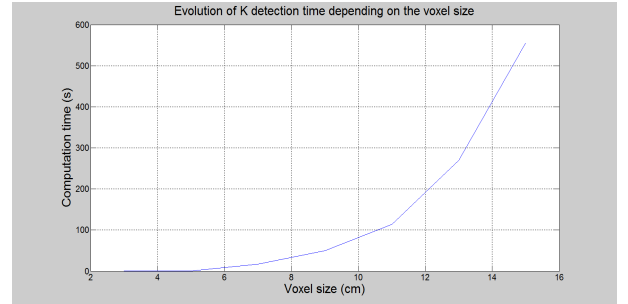


Fig. 8: Evolution of the processing time according the voxel size

order to make a comparison with the PCL method, we propose a quantification of the number of colors in the two resulting clouds after reduction. These two results are compared with the number of colors in the original cloud. For this, three voxelizations are done: one for each cloud with the same voxel size.

A color clustering process in each voxel is made. The result of this comparison, of 100 voxels randomly taken, is shown in Fig. 10. It can be seen that our method (green) performs far better than PCL method (red), while the original one being represented as blue. The main difference between our method and the PCL method lies in calculating the color of the representative point. The PCL method replaces all points



(a) 728 620 pts 87 % reduction (b) 443 633 pts 92 % reduction (c) 292 601 pts 95 % reduction (d) 206 169 pts 96 % reduction (e) 117 793 pts 98 % reduction

Fig. 9: Evolution of reduction process according to the change of voxel size for a single point cloud "Bow" [Original size: 5 147 008 points]

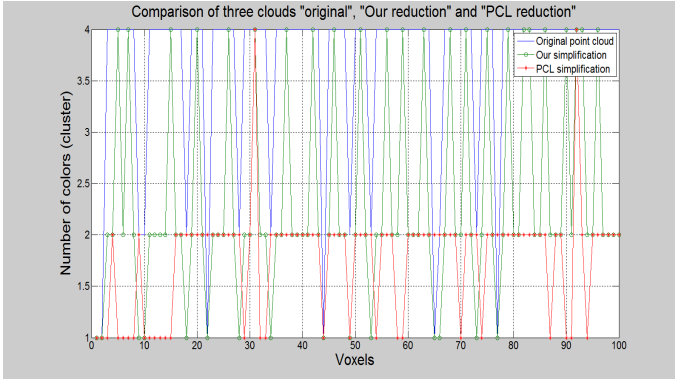


Fig. 10: Number of detected colors according the methods used. The original cloud is included for comparison

within a voxel by their centroid. The color of the resulting point will be calculated from the average color of all points in that voxel. Therefore, PCL creates new points that were not in the original cloud. In our case, however, the exact features of the original point cloud are maintained. This is done by keeping a single representative point of each color for each small voxel. The choice of smaller voxels is intended to limit the reduction rate and to preserve the topological details of the cloud.

V. CONCLUSION

The set of points obtained from a 3D laser scan can be up to several millions of points. A typical digitization can contain multiple scans. To the best of our knowledge, the bibliographic analysis has shown that no reliable method existed yet for the reduction of colored original point clouds. Most of the methods create some new vertices which replace the huge entry points, dismissing by that a large proportion of information carried by these input samples. Our goal was to introduce color features in the subsampling process. The proposed approach aims to find sets of 3D point with the same color in very small region and replacing each set with one point. Data are significantly reduced, while the properties of the scanned objects such as shape and color are preserved.

REFERENCES

- [1] E. Puttonen, M. Lehtomäki, H. Kaartinen, L. Zhu, A. Kukko, and A. Jaakkola, "Improved sampling for terrestrial and mobile laser scanner point cloud data," *Remote Sensing*, vol. 5, no. 4, pp. 1754–1773, 2013.
- [2] P. F. Lee and B. S. Jong, "Point-based simplification algorithm," *WSEAS Trans. on Computer Research*, vol. 3, no. 1, pp. 61–66, 2008.
- [3] H. Song and H. Y. Feng, "A progressive point cloud simplification algorithm with preserved sharp edge data," *Int. Journal of Advanced Manufacturing Technology*, vol. 45, no. 5-6, pp. 583–592, 2009.
- [4] T. Wiemann, M. Mrozinski, D. Feldschneiders, K. Lingemann, and J. Hertzberg, "Data Handling in Large-Scale Surface Reconstruction," in *13th Int. Conf. on Intelligent Autonomous Systems*, 2014, pp. 1–12.
- [5] Y. Li, M. Wei, J. Wu, and M. Pang, "Error-Controllable Simplification of Point Cloud," in *Trans. on Computational Science XVI*, 2012, pp. 149–162.
- [6] M. Pauly, M. Gross, and L. Kobbelt, "Efficient simplification of point-sampled surfaces," in *13th IEEE Visualization conference.*, 2002, pp. 163–170.
- [7] K. Lee, H. Woo, and T. Suk, "Point Data Reduction Using 3D Grids," *The Int. Journal of Advanced Manufacturing Technology*, vol. 18, no. 3, pp. 201–210, 2001.
- [8] P.-F. Lee, C.-H. Chiang, J.-L. Tseng, B.-S. Jong, and T.-W. Lin, "Octree Subdivision Using Coplanar Criterion for Hierarchical Point Simplification," in *IEEE Region 10 Conf. TENCN*, 2006, pp. 54–63.
- [9] L. Se-Ho, J. Seong-Gyun, C. Tae-Young, and K. Chang-Su, "Real-time acquisition and representation of 3D environmental data," in *Signal & Information Processing Association Annual Summit and Conf. (APSIPA ASC), Asia-Pacific*, 2012, pp. 1–5.
- [10] T. Masuda and N. Yokoya, "A Robust Method for Registration and Segmentation of Multiple Range Images," *Computer Vision and Image Understanding*, vol. 61, no. 3, pp. 295–307, 1995.
- [11] Z. Zhang, "Iterative point matching for registration of free-form curves," INRIA, Tech. Rep., 1992.
- [12] G. Blais and M. D. Levine, "Registering multiview range data to create 3D computer objects," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 820–824, 1995.
- [13] J.-D. Boissonnat and F. Cazals, "Coarse-to-fine surface simplification with geometric guarantees," *Computer Graphics Forum*, vol. 20, no. 3, pp. 490–499, 2001.
- [14] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. Silva, "Point set surfaces," in *12th IEEE Visualization Proceedings, VIS '01.*, 2001, pp. 21–28.
- [15] T. Dey and J. Giesen, "Decimating samples for mesh simplification," in *13th Canadian Conf. on Computational Geometry*, 2001, pp. 85–88.
- [16] T. Tibshirani, T. Wiemann, and J. Hertzberg, "Generating Topologically Consistent Triangle Meshes from Large Scale Kinect Fusion," in *IEEE European Conf. on Mobile Robots (ECMR)*, 2015, pp. 1–6.
- [17] L. Kaufman and P. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, Inc., 1990, vol. 33.
- [18] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, 2007, pp. 1027–1035.
- [19] R. Tibshirani, G. Walther, and T. Hastie, "Estimating the number of clusters in a data set via the gap statistic," pp. 411–423, 2001.
- [20] R. Rusu and S. Cousins, "3D is here: point cloud library," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2011, pp. 1 – 4.
- [21] F. Zhang, X. Che, and W. Zuo, "A Simplification Algorithm for Point Cloud Data Based on B-spline Curve," *Journal of Computational Information Systems*, pp. 1821–1828, 2012.