FAST 3D POINT CLOUD SEGMENTATION USING SUPERVOXELS WITH GEOMETRY AND COLOR FOR 3D SCENE UNDERSTANDING

Francesco Verdoja[1], Diego Thomas[2], Akihiro Sugimoto[3]

[1] University of Turin, Computer Science dept., Turin, Italy (fverdoja@unito.it)

[2] Kyushu University, Fukuoka, Japan

[3] National Institute of Informatics, Tokyo, Japan

ABSTRACT

Segmentation of 3D colored point clouds is a research field with renewed interest thanks to recent availability of inexpensive consumer RGB-D cameras and its importance as an unavoidable low-level step in many robotic applications. However, 3D data's nature makes the task challenging and, thus, many different techniques are being proposed, all of which require expensive computational costs. This paper presents a novel fast method for 3D colored point cloud segmentation. It starts with supervoxel partitioning of the cloud, i.e., an oversegmentation of the points in the cloud. Then it leverages on a novel metric exploiting both geometry and color to iteratively merge the supervoxels to obtain a 3D segmentation where the hierarchical structure of partitions is maintained. The algorithm also presents computational complexity linear to the size of the input. Experimental results over two publicly available datasets demonstrate that our proposed method outperforms state-of-the-art techniques.

***Index Terms*—** segmentation, point cloud, supervoxels, hierarchical clustering

1. INTRODUCTION

Many robotic applications, like recognition, grasping and manipulation of unknown objects, require reliable information on the shape of the target to perform well. Therefore, the task of identifying portions of a scene that correspond to single semantic units (i.e., objects) is crucial for those applications. Such task is known as segmentation.

Object segmentation is a challenging task and it is even considered by some researchers as an ill-defined problem [1] mostly because the perception of what is the best segmentation depends heavily on the application and even changes among humans. Traditionally image segmentation was mostly studied on 2D images, as acquiring 3D data of general scenes was expensive and difficult. However, the recent availability of low-cost RGB-D cameras (e.g., Microsoft Kinect and Asus XtionPRO) enables us to capture 3D point clouds at video frame rate. This opens new possibilities for the segmentation of 3D data. Recent tools like KinectFusion allow us to build a complete 3D representation of a scene using measurements acquired from multiple viewpoints. As a result, a drastically increased amount of information becomes available to describe the shape of different objects in the target scene. This is not the case when using 2D images, since they are limited to a single viewpoint. However, the difficulty of the segmentation task (for 3D point clouds) is increased by the nature of the data themselves [2]: (a) point clouds are usually noisy, sparse and unorganized; (b) the sampling density of points is typically uneven due to varying linear and angular rates of the scanner; and (c) they have no statistical distribution pattern.

Anguelov *et al.* [3] address that a good 3D point cloud segmentation algorithm should have the following properties: (a) it should qualitatively take advantage of several kinds of features and know how to trade them off automatically; (b) it should be able to infer the label of a point which lies in sparsely sampled regions thanks to its neighbors; (c) it should adapt to any 3D scanner used. These three properties are crucial to address the aforementioned difficulties derived from the nature of 3D data.

In computer vision, segmentation of 2D images is a deeply explored problem. A wide range of techniques has been proposed to address this task; among them, one of the most popular approaches is graph clustering (e.g., Graph Cuts [4]). To face the high dimensionality of the image segmentation problem, a recent trend is to first perform an oversegmentation, obtaining a non-rigid grid of small regions, known as *superpixels*, that adhere to the edges in the image. These regions are constructed using devoted algorithms like SLIC [5], which is widely used and highly performant. Then, a tailored segmentation algorithm is applied to cluster these regions into a proper segmentation, e.g., [6].

In comparison to 2D image segmentation, segmentation of 3D point clouds has been less studied. According to Nguyen and Le [7], existing methods for 3D point clouds segmentation can be divided into five classes: (i) edge-based approaches: they detect boundaries in the point clouds to obtain segmented regions [8]; (ii) region-based approaches: they use neighborhood information to combine close points sharing similar properties [9]; (iii) attributes-based approaches: they are based on clustering attributes of point cloud data [10]; (iv) model-based approaches: they use geometric primitive shapes (e.g., sphere, cone, plane, and cylinder) to group points (a noticeable method of this class is the popular RANSAC [11]); and finally (v) graph-based approaches: they consider point clouds in terms of a graph [12].

However, most of state-of-the-art methods (e.g., [1, 8, 13, 14]) do not comply with Anguelov's properties, as they use just some of available features of the point cloud (e.g., just considering geometry alone or color alone) or they run on the captured RGB-D image, causing high dependency of those methods on the particular scanner used. We also note that 3D segmentation methods using RGB-D images (e.g., [15]) have not taken the optimal approach because they convert the point cloud to 2D projections (with loss of information) and then patch the obtained segmentation (in 2D) back to the 3D scene; segmenting 3D points by dealing with them directly in 3D is more preferable and reasonable [16].

Recently, *supervoxels*, the extension of the concept of super-

pixels to 3D point clouds, have been proposed [16]. Their use for segmentation, however, has not been explored yet, except for a few works: Stein *et al.* [1] proposed a general approach where supervoxels are clustered according only to a convexity-based geometric criterion; Yang *et al.* [17], instead, cluster supervoxels in urban scenes using some primitives of the objects they expect to find (e.g., houses, cars, trees). This last method is developed under a very specific scenario and is therefore not suitable for general purpose segmentation.

In this paper, we propose a novel segmentation method working directly on colored 3D point clouds. It starts by clustering the points into supervoxels, and then merges them iteratively using a distance metric until a stopping criterion is met. During the iterative merging process, we maintain the hierarchical structure of partitioned regions. The design of the distance metric itself is one of our main contributions: the metric exploits both geometric and color information and combines them adaptively into a single dissimilarity value.

The effectiveness of merging supervoxels based on both geometry and color for segmentation is confirmed by the fact that our proposed method outperforms most recent methods in the state-of-the-art, and that it has linear complexity in computational time with respect to the size of the point cloud. In particular, running time is an important factor for large-scale applications because consumer RGB-D cameras provide a huge amount of data with dense and large scale 3D models. Our method thus strengthens usability of 3D point cloud segmentation for real-world applications.

Finally, the proposed method complies with the three aforementioned essential features addressed by Anguelov *et al.* [3] for point cloud segmentation: our introduced metric leverages on both geometry and color, and trades the two features off automatically (Property (a)) the use of supervoxels defines patches in which features of a point are inferred also from those of its neighbours (Property (b)), and our method is independent from parameters of the scanner used to acquire the data because it directly deals with the 3D point cloud (Property (c)).

2. PROPOSED METHOD

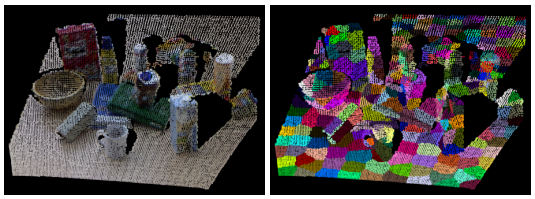
We propose a novel method to segment colored 3D point clouds using supervoxels. The target 3D point cloud is first oversegmented using a state-of-the-art supervoxel segmentation [16]. The initial segmentation is then refined by merging similar neighbouring supervoxels with respect to our proposed distance metric. Our proposed method is fast (runs in linear time) yet accurate, and it complies with all properties of a good 3D point cloud segmentation [3].

Given a point cloud $P = \{x_i\}_{i=1}^{N}$ with N points, we define an n-region segmentation as a partition $R = \{r_i\}_{i=1}^{n}$ of the points of P; more precisely, the regions must satisfy the following constraints:

$$\begin{array}{ll} \forall x \in P & (\exists r \in R \mid x \in r) \ ; \\ \forall r \in R & (\nexists r' \in R \setminus \{r\} \mid r \cap r' \neq \emptyset) \ . \end{array} \tag{1}$$

2.1. Framework

The segmentation of the 3D point cloud is initialized with an oversegmented partition. This first segmentation is obtained with a supervoxel segmentation algorithm. This initialization algorithm has to be chosen carefully as unexpected behaviors could drastically corrupt the final result. In particular, it is important that no supervoxel overlaps two different regions in the expected final segmentation. This is because our proposed method refines the segmentation by iteratively merging neighbouring regions, but is not designed to divide a region. For this reason the initial number of supervoxels has to be chosen accordingly to the expected minimum object size in the scene.



(a) The original point cloud (b) Supervoxels

Fig. 1: A point cloud divided into supervoxels by VCCS

In our method, VCCS [16] is employed, given its known computational efficiency and performances. VCCS initializes a grid of seed points and then adds all the points in a certain radius from each seed to that seed's cluster. Then it iteratively refines this first estimate until a convergence criterion is satisfied. This refinement is carried out by moving points from one cluster to another using a metric based both on color and spatial distance from the centroids of the clusters; Figure 1 shows an example of a point cloud divided into supervoxels by VCCS.

After partitioning the point cloud into m supervoxels, the merging process begins: at every iteration, our proposed algorithm merges the two most similar regions in the current partition, reducing the total number of regions by 1. This iteration is executed until the number of regions becomes 2. This iterative merging algorithm generates a full dendrogram, which inherits information about the hierarchy of regions in terms of their similarity. The generated dendrogram enables us to represent the merging process using a weighted graph. We can then cut the dendogram at the d-th level of the hierarchy to obtain d regions, where d is the desired number of objects. We note that when the number of regions is 2, a binary segmentation is obtained, which is nothing but the discrimination of just foreground and background.

We denote by R^{m-k} the current partition composed of $m-k$ regions at the k-th iteration ($k \in [0, m-2]$). When the algorithm starts, an undirected weighted graph $G^m = \{R^m, W^m\}$ is constructed over the supervoxel set R^m, where $W^m = \left\{w_{ij}^m\right\}, \forall i \neq j$ such that $r_i^m, r_j^m \in R^m \wedge A\left(r_i^m, r_j^m\right) = 1$, for some adjacency function A. Note that $w_{ij}^m = w_{ji}^m$ because G^m is an undirected graph. The weights represent the distance (or dissimilarity measure) between pairs of regions $w_{ij}^m = \delta\left(r_i^m, r_j^m\right)$. The distance function δ is presented in details in Section 2.2.

At the k-th iteration, our algorithm selects the pair of regions $r_p^{m-k}, r_q^{m-k} \in R^{m-k}$ having $w_{pq}^{m-k} = \min W^{m-k}$ and then merges them. As a result, we obtain a new partition $R^{m-(k+1)} = R^{m-k} \setminus \left\{r_q^{m-k}\right\}$ having all the points $x \in r_p^{m-k} \cup r_q^{k-m}$ assigned to the region $r_p^{m-(k+1)}$. Note that $R^{m-(k+1)}$ contains $m-(k+1)$ regions. After that, edges and corresponding weights are updated for subsequent iterations. $W^{m-(k+1)}$ is generated according to the following rule:

$$w_{ij}^{m-(k+1)} = \begin{cases} \delta\left(r_p^{m-(k+1)}, r_j^{m-(k+1)}\right) & \text{if } i = p \vee i = q \\ w_{ij}^{m-k} & \text{otherwise} \end{cases} . \tag{2}$$

Note that $w_{pq}^{m-k} \notin W^{m-(k+1)}$ since it does not exist anymore.

When $k = m-2$, the algorithm stops and returns the full dendrogram $\mathcal{D} = \left\{R^m, \ldots, R^2\right\}$, from which we can obtain the desired number of regions. If the number d of objects to be segmented

is known in advance, simply taking the level R^d of $\mathcal{D}$ is enough. However, typically this information is not known in advance. In this scenario, a proper threshold has to be set on some metric to automatically assess which level of $\mathcal{D}$ represents the best segmentation. The learning strategy should be tailored on the dataset; in Section 4 we discuss the strategy we adopted for our evaluation data.

2.2. Distance metric

The distance metric δ plays a crucial role in our proposed segmentation algorithm: it decides which regions are to be merged at each iteration. As a consequence, the distance metric should be designed carefully as it drastically impacts the segmentation result. In this paper, we adhere to the following three main criteria so that the above mentioned properties addressed by Anguelov *et al.* [3] are satisfied: 1. it should exploit both the color and the geometry information provided by the point cloud; 2. it should possess capability of adaptively and automatically adjusting the weights of the two components; 3. it should have values bounded in the range $[0, 1]$, where 0 corresponds to the maximum similarity and 1 to the minimum one.

We propose the first (to the best of our knowledge) distance metric clustering supervoxels on 3D point clouds that satisfies these three criteria. Our proposed metric is a combination of two terms: a color data term δ_C that captures the color distance between any two regions; and a geometric data term δ_G that captures the geometric distance between any two regions. Color data and geometric data are two inherently different types of data, with different dynamics and different distributions over the point cloud. Therefore the two data terms δ_C and δ_G have to be transformed to a unified domain for efficient combination. We define T_C and T_G the two transformations that transform δ_C and δ_G (respectively) from their initial (different) domains to a unified one that ranges between 0 and 1. For any two regions r_i and r_j, $(i, j) \in [0; m-k]^2$, our proposed distance value $\delta(r_i, r_j)$ between r_i and r_j can be written as:

$$\delta\left(r_i, r_j\right) = T_C\left(\delta_C\left(r_i, r_j\right)\right) + T_G\left(\delta_G\left(r_i, r_j\right)\right). \tag{3}$$

Note that the distance metric in (3) is modular: any of the components might be changed according to the specific needs of an application. In this work we present and evaluate some variations that we have designed for a general purpose segmentation.

2.2.1. Color-data term

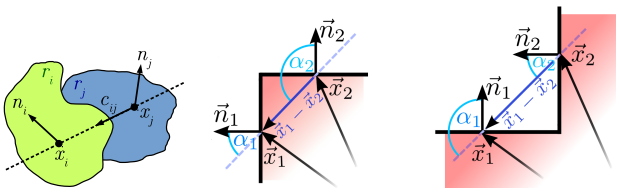
To better match human color perception, Lab color space and the standard CIEDE2000 color difference have been chosen [18]. Given two regions r_i and r_j, we compute the mean values of the L*a*b* components $M_i = \left(\mu_{(L^*,i)}, \mu_{(a^*,i)}, \mu_{(b^*,i)}\right)$ and $M_j = \left(\mu_{(L^*,j)}, \mu_{(a^*,j)}, \mu_{(b^*,j)}\right)$, and we define the distance between the two labels as

$$\delta_C\left(r_i, r_j\right) = \frac{\Delta E_{00}\left(M_i, M_j\right)}{R_{\Delta E_{00}}}, \tag{4}$$

where ΔE_{00} is the CIEDE2000 difference [18] and $R_{\Delta E_{00}}$ its range.

2.2.2. Geometric-data term

For any two adjacent regions r_i and r_j, we denote by $\vec{x}_i$ and $\vec{x}_j$ their respective centroids, by $\vec{n}_i$ and $\vec{n}_j$ their unit normals, and by $\vec{c}_{ij}$ the unit vector laying on the line connecting the two centroids. More precisely $\vec{c}_{ij} = \frac{\vec{x}_i - \vec{x}_j}{\|\vec{x}_i - \vec{x}_j\|}$. Figure 2a shows a graphical rapresentation of this formalism. We define the geometric data term δ_G



(a) (b) (c)

Fig. 2: Geometrical representation of two regions and their features (a) and illustrations of convex (b) and concave (c) angles.

between the two regions r_i and r_j as:

$$\delta_G\left(r_i, r_j\right) = \frac{\|\vec{n}_i \times \vec{n}_j\| + |\vec{n}_i \cdot \vec{c}_{ij}| + |\vec{n}_j \cdot \vec{c}_{ij}|}{3}, \tag{5}$$

where $\|\vec{v}\|$ indicates the magnitude of the vector $\vec{v}$, while $|a|$ is the absolute value of the scalar a. This distance function exploits the properties of cross and dot products between unit vectors for which

$$\|\vec{v} \times \vec{u}\| = \begin{cases} 0 & \text{iff } \vec{v} \parallel \vec{u} \\ 1 & \text{iff } \vec{v} \perp \vec{u} \end{cases}, \text{ and } |\vec{v} \cdot \vec{u}| = \begin{cases} 0 & \text{iff } \vec{v} \perp \vec{u} \\ 1 & \text{iff } \vec{v} \parallel \vec{u} \end{cases}. \tag{6}$$

Therefore, the geometric data term returns maximum similarity if the two normals are parallel to each other and at the same time they are both perpendicular to the line connecting the two centroids; i.e., if r_i and r_j are coplanar.

Stein *et al.* [1] proposed a segmentation method based on the property that faces of the same object usually make convex angles with each other, while two different objects touching each other will produce most of the time a concave angle. This property is observed and exploited also in [13, 19]. Figure 2b and Figure 2c show graphical representations of convex and concave angles. We designed a variation δ'_G of our proposed geometric data term that also incorporates a convexity criterion inspired by the one proposed in [1]. Namely, δ'_G is defined as:

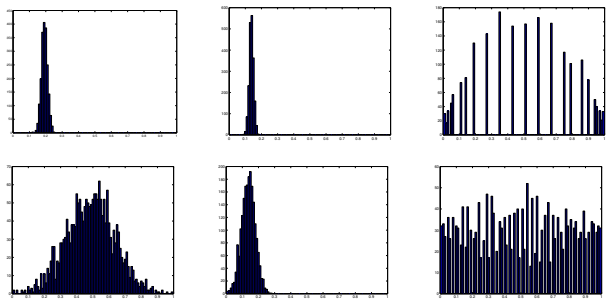
$$\delta'_G = \begin{cases} \frac{1}{2}\delta_G & \text{if } \vec{n}_i \cdot \vec{c}_{ij} - \vec{n}_j \cdot \vec{c}_{ij} \geq 0 \\ \delta_G & \text{otherwise} \end{cases}. \tag{7}$$

In other words, if the angle between the normal vectors of two adjacent regions is convex, we are halving the geometric data term presented in (5) computed for the two regions as it is more probable that they are parts of the same object. Note that $\vec{n}_i \cdot \vec{c}_{ij} - \vec{n}_j \cdot \vec{c}_{ij} \geq 0$ if and only if the angle between the normal vectors of the regions r_i and r_j is convex.

2.2.3. Unification transformations

Both δ_C and δ_G have ideally range $[0, 1]$, but they might have a drastically different dynamic and distribution when applied to different point clouds. For this reason, if they are merged without first applying a proper transformation, their values can't be reliably compared. Therefore, the goal of T_C and T_G is to transform δ_C and δ_G (respectively) so that, when they are merged together, none overcomes the other. Moreover, the two transformations should respect the following constraints:

1. $T_C : \mathbb{R} \to [0, c]$ and $T_G : \mathbb{R} \to [0, g]$ for some $c, g \in \mathbb{R}_0^+ \mid c + g = 1$.
2. T_C and T_G should be monotonically increasing.



(a) (b) (c)

Fig. 3: Histograms of the distribution of (a) δ_C and δ_G, (b) δ_C and δ_G after being transformed with Adaptive Lambda, and (c) δ_C and δ_G after being transformed with Equalization.

A possible choice might be to use some value $\lambda \in [0, 1]$ as weight; in that case we will have $T_C(a) = \lambda a$ and $T_G(b) = (1 - \lambda)\, b$. Since we would like it to adapt automatically we propose to define the value of λ as a function of the distributions of the two components. We assume that δ_C and δ_G have two unknown distributions with known means μ_C and μ_G, respectively. We then define $\lambda = \mu_G/(\mu_C + \mu_G)$, so that $\lambda\mu_C = (1 - \lambda)\,\mu_G$. We call this pair of transformations *Adaptive Lambda*.

Another option might be instead to use T_C and T_G to apply equalization to δ_C and δ_G; this way they will cover the whole range $[0, 1]$ with uniform distribution. Given a function $f \in [0, m]$ having unknown distribution, we can sample f on an input of n elements obtaining a sample distribution $\bar{f}$. Let's call n_i the number of occurrences of the output $i \in [0, m]$, $p_i = p(f = i) = \frac{n_i}{n}$ the frequency of the output i, and $cdf_f(i) = \sum_{j=0}^{i} p_j$ the cumulative distribution function of f. Then, to equalize the distributions of δ_C and δ_G we have to impose $T_C\,(a) = \frac{1}{2} cdf_{\delta_C}(a)$ and $T_G\,(b) = \frac{1}{2} cdf_{\delta_G}(b)$. The division by 2 is used to bring the range of both transformations in $[0, \frac{1}{2}]$, so that when they are summed according to (3) they respect the Property 3 defined at the beginning of this section. We will call this pair of transformations *Equalization*.

Figure 3 shows the effect of Adaptive Lambda and Equalization on two sample distributions for δ_C and δ_G. It can be noticed how Adaptive Lambda projects the two distributions onto one reference Gaussian distribution, while Equalization projects δ_C and δ_G onto the uniform distribution.

3. COMPUTATIONAL COMPLEXITY

Consumer-grade depth cameras generate dense 3D data at 30 fps from a hand-held sensor. The amount of points in the built 3D cloud of points is thus potentially enormous. For any 3D point cloud segmentation to be usable in practice with 3D models generated using a consumer-grade depth camera, the method has to scale well to large size point clouds. In this section, we discuss in details the time complexity of our proposed method with respect to the size of the point cloud. Our inputs are the point cloud P with N points and the number of supervoxels m ($m \ll N$).

Initialization The starting point is to compute the m supervoxel partition. For this purpose, we use VCCS that converges in $O(N)$ operations [16]. The three features of a region (i.e., centroid, normal vector and mean color) used to compute δ need to be evaluated once per region. Computing these features for all regions in the initial partition takes $O(N)$ time. This is because, according to Eq. (1), no two regions are overlapping and each point needs to be considered only once. For each pair (r_i, r_j) of adjacent regions computing $\delta(r_i, r_j)$ takes $O(1)$ operations because all features are already computed. The following update of W^m also takes $O(1)$ operations. For each region in the current partition there is a limited number of adjacent regions. Therefore, the number of adjacent regions is dominated by N and building the initial set W^m (and thus the graph G^m) takes $O(N)$ operations. As a consequence, the full initialization pipeline takes $O(N)$ operations. Note that the set of weights W^m can be sorted while being built to fasten computations in the clustering loop.

Clustering loop At the k-th iteration, identifying the pair (r_q, r_p) of regions with minimum weight takes $O(1)$ operation because W^{m-k} is sorted. When merging the two regions r_p and r_q, the three features are recomputed, which takes $O(s)$ operations, where $s = |r_p \cup r_q|$. Moreover, the weights w_{pj} for each pair (r_p, r_j) of adjacent regions is updated. This takes $O(1)$ operations as there is a limited number of regions adjacent to r_p. Then the recomputed weights need to be inserted in the sorted list of weights W^{m-k}. Inserting an element in a sorted data structure can take as lower as $O(\log n)$, in case of self-balancing B-tree; in our case n is the number of regions (so $m - k$, at iteration k). Therefore, the overall cost of the clustering step is $O(m(h + \log m))$, where h is the maximum number of points in a region. m is considerably smaller than N, while h is at most N: $m \ll N$ and $h < N$. Then $O(\log m) \ll O(N)$, $O(h + \log m) < O(N)$ and thus $O(m(h + \log m)) < O(N)$. In other words, the overall cost is dominated by $O(h)$, and therefore it is still in the order of $O(N)$.

In conclusion, the overall computational cost required for the proposed algorithm is linear with respect to the size of the point cloud. This means that our proposed method is promising for scaling up to large-scale 3D point clouds.

4. EVALUATION

To objectively assess quality of a segmentation, we used the Object Segmentation Dataset (OSD) [20] included in the PCL because it is the only dataset for 3D segmentation having ground-truth information publicly available, to our knowledge.

OSD contains 110 point clouds, divided in a training set of 45 and a test set of 65. Each of these point clouds captures a scene where a variable number of objects is present on the surface of a table, in some scenes stacked one on top of the other or occluding each other. Each scene is provided together with ground-truth manually segmented by a human. Running our algorithm over this dataset allows us to compare our results with those obtained by two other recent methods in the state-of-the-art: one is [1] and the other is [8]. The former, as already mentioned earlier, uses supervoxels like ours, but, differently from ours, it clusters supervoxels exploiting only geometric information. The latter employs edges information to obtain a first estimate of segmentation and then refines it using topological neighborhood. We remark that although OSD allows us to objectively compare performances with other state-of-the-art techniques, scenes in that dataset are quite similar between each other, under controlled environments and obtained from single RGB-D images.

To objectively evaluate results on the dataset, we employed the metrics below, both of which were also used in [1, 8]. The ground-truth $G = \{g_i\}_{i=1}^{K}$ is the set of K human annotated regions g_i.

Weighted Overlap For each region g_i, among the regions obtained by the segmentation to be tested, the region having the maximum overlap with g_i is selected as its best estimate. Then, we define

Table 1: Average scores obtained on OSD using the threshold t yielding the best result on the training set. Results of state-of-the-art techniques are provided for comparison. All scores are percentages.

		t	WOv	tp	fp	fn
Stein *et al.* [1]			87,00	90,70	4,30	9,30
Ückermann *et al.* [8]			-	92,20	**1,90**	7,80
Only color		0.10	85,34	87,10	7,03	8,60
Only geometry		0.29	86,91	89,45	6,14	7,96
Equalization	δ_G	0.94	89,39	91,59	5,37	8,41
	δ'_G	0.94	89,61	92,08	5,27	7,92
Adaptive λ	δ_G	0.18	92,96	94,00	2,43	6,00
	δ'_G	0.15	**92,98**	**94,36**	2,39	**5,64**

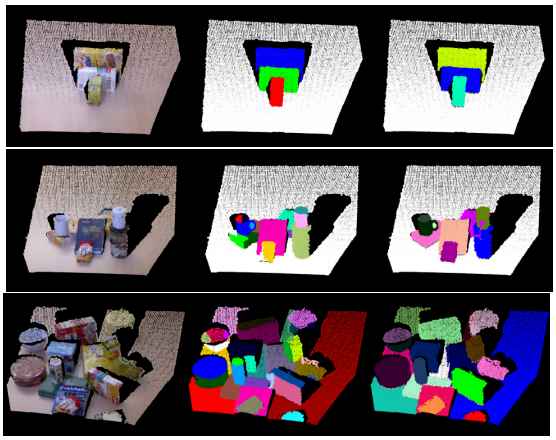


Fig. 4: Three sample segmentations; the first column hosts the original point clouds, the second the results of our segmentation using Adaptive Lambda and δ'_G, the third the ground-truths.

$Ov_i = \max_{r_j} \{|g_i \cap r_j| \,/\, |g_i \cup r_j|\}$. The *Weighted Overlap (WOv)* is computed as [14]: $WOv = \frac{1}{\sum_i |g_i|} \sum_i |g_i| \cdot Ov_i$. The range of the above metric is $[0, 1]$, where 1 correspond to the perfect overlap (i.e., the segmentation is the exact same partition as the ground-truth).

True- and False-positive rates We denote by s_i the region in the segmentation having maximum overlap with the region g_i in G. For each g_i, we define the true positive points by $TP_i = g_i \cap s_i$ while we define the false positive points by $FP_i = s_i \setminus TP_i$. We also define the false negative points by $FN_i = g_i \setminus TP_i$. Finally the average scores over all g_i in G are defined as [8]: $tp = \frac{1}{K} \sum_i \frac{|TP_i|}{|g_i|}$, $fp = \frac{1}{K} \sum_i \frac{|FP_i|}{|r_i|}$, and $fn = \frac{1}{K} \sum_i \frac{|FN_i|}{|g_i|}$. They all have $[0, 1]$ as range, but for tp higher is better, while for fp and fn lower is better.

As explained in Section 2.1, the proposed algorithm has the property to construct a whole hierarchy of segmentation ranging from m regions (i.e., the starting number of supervoxels) to 2 regions in terms of the dendogram. This dendogram needs to be cut at a proper point to obtain the segmentation. There are many ways to determine this cutting criterion. In our case, we simply stopped our algorithm when $\min W^{m-k} \geq t$, for a certain threshold t. We learned the best threshold for this dataset over the training set and then we evaluated the performance using that threshold over the whole dataset. To this end, we have run our algorithm on the training set using all values of $t \in [0, 1]$ with an increment of 0.01. Then we have taken as the best t the one giving the best average tp score over the training set. The value m was instead set to 250.

Table 1 illustrates the performances of the algorithm using the four different variations on the distance metric detailed in Section 2.2: using the geometric metric δ_G in its simple form or the metric weighted according to the convexity (i.e., δ'_G); and both the transformations, Adaptive Lambda and Equalization. Also, results obtained when using only the color-data term δ_C and only the geometry-data term δ_G are presented. Our results are compared also with those obtained by [1] and [8] as presented in their respective papers. For [1] the parameters used are $v = 0.5$, $s = 2$, $n_{filter} = 3$ and $\beta_{thresh} = 10$, while for [8] $n = 6$ and $\theta_{max} = 0.85$.

We observe that all results are in line with, when not better than, the state-of-the-art ones. Adaptive Lambda performs better than Equalization. This suggests that δ_C and δ_G distributions are important to segmentation and equalizing them (i.e., projecting them onto the uniform distribution) results in a loss in performance. Incorporating weights based on the convexity of the angles slightly improves the performance. The gain obtained by δ'_G over δ_G however is very small and could vary from domain to domain, given its hypothesis on the object shapes. So the use of δ_G might be preferred for some applications. In all cases the metric proposed outperforms the use of only one of the two components. The distance metric using Adaptive Lambda decisively outperforms the state-of-the-art methods. This difference in performance is statistically significant: we performed t-tests on the hypotheses that our technique using Adaptive Lambda and δ'_G achieves better results than any of the other test candidate; all hypotheses are confirmed with $\alpha = 0.01$ (significance level), and in many cases even with $\alpha = 0.005$. We also performed t-test on the hypothesis that the fp rate obtained by [8] is statistically lower than ours; this hypothesis is rejected (not confirmed even with $\alpha = 0.05$). It is also important to remark that these performances are obtained using a simple learning algorithm for the threshold; using a more refined learning strategy is expected to improve even more the performances of our method.

In terms of running time, the C++ implementation of the algorithm used for the evaluation process is able to segment 307,200 points in 1140 ms on average, which is enough for most real-time robotic applications, e.g., robot grasping [21]. Moreover, the implementation used doesn't exploit any parallel computation at the moment, although many parts of the algorithm may be easily parallelized. Any parallel implementation should lead to a considerable gain in running time. The machine used has an Intel® Core™ i7@2.20GHz CPU.

In Figure 4, three different examples of segmentations obtained by our algorithm are shown. It can be noticed how, even in complex scenes, our algorithm is still able to segment all the different objects. As the complexity of the scene increases, however, it tends to separate faces of the same object. This might arise from the fact that the training set of the dataset does not contain complex scenes while the test set does. Accordingly, the threshold learned on the training set results a little too low for scenes with much higher complexity. It should be possible to address this issue using a more robust learning strategy for the threshold.

To address the effectiveness of our proposed method in more challenging scenes, we also tested it on the *KinFu* dataset proposed by Kaparthy *et al.* [13]. It represents many indoor 3D scenes obtained from a RGB-D video stream processed with the KinectFusion algorithm to obtain a single fused point cloud. The dataset is more challenging than OSD in that it has a greater number of variegate objects in the scene though it has no ground-truth. Figure 5 illustrates few scenes in KinFu together with our segmented outputs

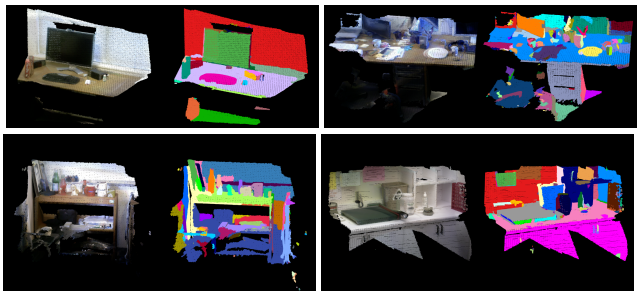


Fig. 5: Results on KinFu dataset using Adaptive Lambda and δ'_G

(with Adaptive Lambda and δ'_G) using the same threshold $t = 0.15$ that was learned on the different dataset OSD. We show only a qualitative result due to the absence of ground-truth for KinFu.

5. CONCLUSIONS

We have presented a novel algorithm tailored for segmenting 3D colored point clouds. The technique starts by dividing the point cloud into supervoxels; then clusters them using a hierarchical approach driven by a metric of our design which merges both color and geometric information to assess the similarity of adjacent regions. The contribution of the proposed approach is two-folded: the algorithm is fast, thanks to its complexity linear to the size of the input, and it is able to achieve performances better than those in the state-of-the-art. Another important contribution of this work is the study of a modular distance metric for 3D point clouds and the evaluation of the performances of some of the possible variations of the proposed metric. The importance of the proposed merging criteria is indeed confirmed by increased performances when compared with the use of color or geometry alone. Finally, we remark that our proposed metric, though evaluated in an iterative segmentation framework in this paper, is general enough to be considered for use in other state-of-the-art frameworks such as graph-based approaches. Investigation in such direction is left for future work.

6. REFERENCES

[1] S. C. Stein, F. Wörgötter, M. Schoeler, J. Papon, and T. Kulvicius, "Convexity based object partitioning for robot applications," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 3213–3220, IEEE.

[2] S. Sotoodeh, "Outlier detection in laser scanner point clouds," *Int. Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 36, no. 5, pp. 297–302, 2006.

[3] D. Anguelov, B. Taskar, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A. Ng, "Discriminative learning of Markov random fields for segmentation of 3D scan data," in *Computer Vision and Pattern Recognition*. 2005, vol. 2, pp. 169–176, IEEE.

[4] Y. Boykov and G. Funka-Lea, "Graph Cuts and Efficient N-D Image Segmentation," *International Journal of Computer Vision*, vol. 70, no. 2, pp. 109–131, 2006.

[5] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC Superpixels Compared to State-of-the-Art Superpixel Methods," *Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

[6] Z. Ren and G. Shakhnarovich, "Image Segmentation by Cascaded Region Agglomeration," in *Computer Vision and Pattern Recognition*. 2013, pp. 2011–2018, IEEE.

[7] A. Nguyen and B. Le, "3d point cloud segmentation: A survey," in *2013 6th IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, 2013, pp. 225–230.

[8] A. Ückermann, R. Haschke, and H. Ritter, "Real-time 3d segmentation of cluttered scenes for robot grasping," in *Humanoid Robots (Humanoids), 2012 12th IEEE-RAS International Conference on*. 2012, pp. 198–203, IEEE.

[9] J. Chen and B. Chen, "Architectural Modeling from Sparsely Scanned Range Data," *International Journal of Computer Vision*, vol. 78, no. 2-3, pp. 223–236, 2007.

[10] G. Vosselman, S. Dijkman, and others, "3d building model reconstruction from point clouds and ground plans," *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, vol. 34, no. 3/W4, pp. 37–44, 2001.

[11] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[12] R. B. Rusu, A. Holzbach, N. Blodow, and M. Beetz, "Fast geometric point labeling using conditional random fields," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009. IROS 2009*. 2009, pp. 7–12, IEEE.

[13] A. Karpathy, S. Miller, and L. Fei-Fei, "Object discovery in 3d scenes via shape analysis," in *International Conference on Robotics and Automation*. 2013, pp. 2088–2095, IEEE.

[14] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor Segmentation and Support Inference from RGBD Images," in *Computer Vision ECCV 2012*, number 7576 in Lecture Notes in Computer Science, pp. 746–760. Springer, 2012.

[15] I. Jebari and D. Filliat, "Color and Depth-Based Superpixels for Background and Object Segmentation," *Procedia Engineering*, vol. 41, pp. 1307–1315, 2012.

[16] J. Papon, A. Abramov, M. Schoeler, and F. Wörgötter, "Voxel Cloud Connectivity Segmentation - Supervoxels for Point Clouds," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2027–2034.

[17] B. Yang, Z. Dong, G. Zhao, and W. Dai, "Hierarchical extraction of urban objects from mobile laser scanning data," *ISPRS J Photogram Remote Sens*, vol. 99, pp. 45–57, 2015.

[18] G. Sharma, W. Wu, and E. N. Dalal, "The CIEDE2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations," *Color Research and Application*, vol. 30, no. 1, pp. 21–30, 2005.

[19] D. F. Fouhey, A. Gupta, and M. Hebert, "Unfolding an indoor origami world," in *European Conference on Computer Vision*. 2014, pp. 687–702, Springer.

[20] A. Richtsfeld, T. Mrwald, J. Prankl, M. Zillich, and M. Vincze, "Segmentation of unknown objects in indoor environments," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Oct. 2012, pp. 4791–4796, IEEE.

[21] J. Redmon and A. Angelova, "Real-time grasp detection using convolutional neural networks," in *International Conference on Robotics and Automation*, 2015, pp. 1316–1322.