Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

A Supervoxel Segmentation Method with Adaptive Centroid Initialization for Point Clouds

V. ANIRUDH PULIGANDLA[1], (Member, IEEE), SVEN LONČARIĆ[2], (Senior Member, IEEE)

[1]Image Processing Group, Department of Electronic Systems and Information Processing, Faculty of Electrical Engineering and Computing, University of Zagreb, Zagreb, Croatia (e-mail: apuligandla@fer.hr)

[2]Image Processing Group, Department of Electronic Systems and Information Processing, Faculty of Electrical Engineering and Computing, University of Zagreb, Zagreb, Croatia (e-mail: sven.loncaric@fer.hr)

Corresponding author: V Anirudh Puligandla (e-mail: apuligandla@fer.hr).

This work is supported by the ImmerSAFE (Project number 764951) project funded under the EU's H2020-MSCA-ITN-2017 call and is part of the Marie Sklodowska-Curie Actions - Innovative Training Networks (ITN) funding scheme.

ABSTRACT Supervoxels find applications as a pre-processing step in many image processing problems due to their ability to present a regional representation of points by correlating them into a set of clusters. Besides reducing the overall computational time for subsequent algorithms, the desirable properties in supervoxels are adherence to object boundaries and compactness. Existing supervoxel segmentation methods define the size of a supervoxel based on a user inputted resolution value. A fixed resolution results in poor performance in point clouds with non-uniform density. Whereas, other methods, in their quest for better boundary adherence, produce supervoxels with irregular shapes and elongated boundaries. In this article, we propose a new supervoxel segmentation method, based on k-means algorithm, with dynamic cluster seed initialization to ensure uniform distribution of cluster seeds in point clouds with variable densities. We also propose a new cluster seed initialization strategy, based on histogram binning of surface normals, for better boundary adherence. Our algorithm is parameter-free and gives equal importance to the color, spatial location and orientation of the points resulting in compact supervoxels with tight boundaries. We test the efficacy of our algorithm on a publicly available point cloud dataset consisting of 1449 pairs of indoor RGB-D images, i.e., color (RGB) images coupled with depth information (D) mapped per pixel. Results are compared against three state-of-the-art algorithms based on four quality metrics. Results show that our method provides significant improvement over other methods in the undersegmentation error and compactness metrics and, performs equally well in the boundary recall and contour density metrics.

INDEX TERMS Clustering methods, supervoxels, over-segmentation, point clouds

I. INTRODUCTION

Like superpixels in 2D images, supervoxels are a collection of 3D points or pixels of a 3D image that are grouped together based on closeness between their spatial location and other textural features. For this work, we define supervoxels as disjoint clusters of points in a point cloud. Supervoxels represent regions in a point cloud that share common features, such as spatial location, color, and orientation. Subsequent computationally intensive image processing algorithms work on supervoxels instead of individual points or pixels to save computational time. Supervoxels find applications in various fields, such as point cloud segmentation and classification, [1], 3D semantic segmentation of point clouds, [2], [3], medical imaging, [4], [5], object detection, [6] and saliency detection, [7], to name a few. Despite so many applications, there is few literature that deals with clustering methods tailored for point clouds.

The desirable properties in a supervoxel include: (1) boundary adherence, i.e., a supervoxel should preserve object boundaries and should overlap with only one object and not cross over the boundaries, (2) compactness, i.e., supervoxels should have a regular shape and should not have elongated and arbitrary boundaries, and (3) efficiency, i.e., they should be computed fast enough not to decrease the efficiency of subsequent algorithms that use supervoxels. Existing supervoxel segmentation methods fall short on some

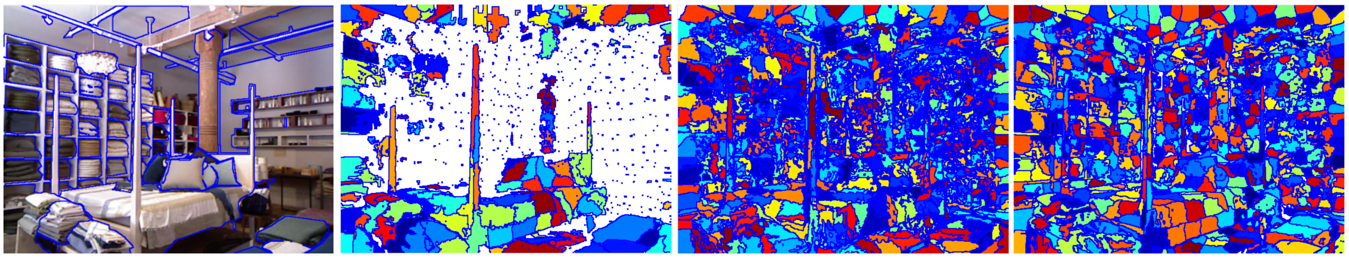
(a) Ground truth (b) vccs (c) ssp (d) proposed

FIGURE 1. An example showing deficiencies in existing supervoxel segmentation methods and improvements in those aspects achieved by our proposed method

of these desirable aspects, especially, compactness. Fig. 1 shows an example of some of the shortcomings of existing methods. Fig. 1 shows a point cloud of a complex scene with many objects and high depth of field. Methods that rely on a constant user-inputted supervoxel resolution value fail when the point cloud density or depth varies steeply (see Fig. 1(b)), while other methods that adapt well to variable point density and can provide good boundary adherence for general point clouds, produce irregularly shaped supervoxels (see Fig. 1(c)). Non-compact supervoxels introduce spatial discontinuity and are not a desirable property in good supervoxels. Moreover, the artificially elongated boundaries may influence the values of comparison metrics to show more accuracy but do not look visually appealing.

In this work, we propose a new clustering method for colored point clouds. Our method is based on the k-means algorithm, like the SLIC algorithm by Achanta et al., [8]. The distance metric used in our method gives equal importance to color, points' spatial position and their orientation and is thus, free from implicit parameters. This property produces compact clusters with regular shapes. To maintain sensitivity to varying point density, we introduce a cluster seed re-initialization strategy to dynamically remove cluster seeds with very few assigned points or to add additional cluster seeds in regions with considerable number of unlabeled points. For better boundary adherence, we introduce a new strategy to first create a histogram of all points based on their surface normals and initialize cluster seeds according to the histogram bins. This strategy allows us to create cluster seeds in small, isolated regions or objects that may be missed otherwise when cluster seeds are distributed uniformly across the spatial extent. We tested the efficacy of our method on the publicly available NYU Depth V2 dataset, [9], and compared it against three state-of-the-art supervoxel segmentation methods based on four evaluation metrics.

Results show that our method performs best in terms of undersegmentation error and compactness metrics. While the performance of our method in the boundary recall metric is comparable to existing methods, we show that our method produces compact supervoxels with fine boundaries which make it look visually appealing than other methods. Although dynamic cluster seed initialization introduces additional computational overhead, the gained accuracy in terms of boundary adherence and compactness can be beneficial for algorithms that are not seriously restricted in time. We previously introduced this method in [10] where we used it to cluster a set of 3D points with surface normals representing camera poses on the surface of a vehicle's 3D model. It was introduced as a pre-processing step to reduce the input complexity of an optimal camera placement (OCP) problem for vehicle surround vision. The method showed promising results for the OCP problem by significantly reducing the overall computational time (up to 160 times). However, the supervoxel method was not analyzed as it was used as a pre-processing step. In this article, we present a detailed analysis of the method on colored point clouds to compare its efficacy against state-of-the-art supervoxel segmentation methods. The rest of the document is organized as follows: Section II details relevant literature, Section III details our proposed clustering method and the results are discussed in Section IV.

II. BACKGROUND WORK

Superpixels are 2D versions of supervoxels. While superpixels are extensively studied in the field of image processing, [8], [11]–[14], supervoxels have not been studied enough despite their requirement due to recent advances in 3D image analysis. In the beginning, video sequences or stacks of 2D images collected over time were considered as 3D images. Therefore, the first 3D extensions of superpixel methods were tailored to deal with stacks of images, with time being the third dimension. [8], [14], [15] are some of the first supervoxels methods that extended their work to video sequences. Moore et al., [14] produced over-segmentation on videos by iteratively partitioning pixels into clusters by horizontal and vertical cutting in 3D grid. Achanta et al., [8] proposed an efficient and widely successful approach based on the k-means algorithm. In their method, they distribute cluster seeds uniformly across a 2D or 3D grid, search a local neighbourhood around each cluster seed and assign points to the closest cluster center based on a distance metric that relates pixels to a cluster center using position and color information. The primary idea behind the clustering method we propose here is based on this method.

In [16], Veksler et al., proposed another supervoxel method for videos where they formulate it as an energy minimization problem and solve it using graph cuts. [17]–[19] are some of the pioneering works on supervoxel segmentation for

RGB-D images. In [17], the authors extended their previous work on depth-adaptive superpixels to RGB-D videos. They used color and point normal information to construct a graph of spatio-temporal supervoxels and used spectral graph clustering to partition the graph into spatio-temporal segments. Gao et al., [18], proposed a new cluster seed initialization scheme for dense cluster seed initialization in salient regions of the image. Their motivation for adaptive cluster seed initialization is like ours and such a strategy works well to improve overall accuracy by producing clusters with non-uniform sizes and densities. Zhou et al., [20], used hierarchical edge weighted Voronoi tessellation to propose a multi-scale supervoxel algorithm that gradually constructs supervoxels at higher levels based on the supervoxels constructed at lower levels. The method that we propose here works on point clouds in 3D space. It is like other methods for RGB-D data only in the sense that we use mapped RGB-D to construct point clouds.

Papon et al., [21] proposed one of the first supervoxel segmentation methods (vccs) tailored for point clouds. They first voxelate the point cloud and cluster them based on voxel adjacency. They initialize cluster seeds uniformly across the voxelated point cloud and use the same distance metric as SLIC, [8]. They use voxel adjacency graphs to iteratively add neighbors to cluster seeds until all the voxels are assigned a label. Their method is simple and fast but, the voxel resolution parameter fails to adapt well to point clouds with variable density. Also, voxelization produces an approximation of the underlying points, thereby decreasing the quality of the method's boundary adherence. Our proposed method is like vccs in some aspects, but the primary difference is that our method works directly on the points. Lin et al., [22], more recently proposed a new supervoxel segmentation method for point clouds while citing the limitations of vccs. They formulate it as a subset selection problem based on an energy function that can be optimized to find optimal subsets. Their method does not require initialization of cluster seeds and is claimed to produce supervoxels with non-uniform resolution to better adapt to boundaries. Their method, however, produces irregularly shaped supervoxels with arbitrarily elongated boundaries (see Fig. 1(c)).

More recently, supervoxel methods for point clouds have garnered increased research interest. In [23], the authors propose modified versions of the vccs algorithm that are better suited for point clouds. In the method, they gather point neighbours without voxelization and combine neighbours computed by different methods to create supervoxels directly on the point cloud. Dong et al., [24] proposed a method that is capable of GPU acceleration. They divide the algorithm into two stages, where they produce an initial segmentation based on energy functions in the first stage and improve the result by minimizing segmentation energy in the second stage. Ni et al., [25], proposed a new supervoxel segmentation method based on local allocation. They propose a novel cost function for preserving boundaries which is claimed to achieve satisfactory results through local minimization enforcement. Lastly, [26], [27] use deep learning methods for the learning geometrical features of point clouds and produce supervoxel segmentation. However, none of these recent methods emphasise on the compactness of supervoxels.

III. SUPERVOXEL SEGMENTATION FOR POINT CLOUDS

Our clustering algorithm is an iterative process like *k-means* algorithm. It is dynamic in nature as at each iteration, we allow for new clusters centers to be added and/or existing ones to be removed depending on the number of outlier points and the clusters' sizes. The algorithm works on a set of N points, $\mathcal{P} = \{p_1, \ldots, p_N\}$, where each point is represented by its position in 3D, (x, y, z), color in RGB space, (r, g, b), and a unit surface normal vector, (u, v, w), as,

$$p_i = [x_i\ y_i\ z_i\ r_i\ g_i\ b_i\ u_i\ v_i\ w_i]^T\ \forall i = 1 : N. \tag{1}$$

The goal is to group them into K disjoint subsets $\mathcal{S} = \{S_1, \ldots, S_K\}$, where each subset, S_k represents a supervoxel with the label k. At the end of the clustering process, it is expected that every point in $\mathcal{P}$ is assigned a label $k \in [0, \ldots, K]$, depending on which supervoxel the point belongs to. Each supervoxel is represented by its centroid, C_k, that is a 9D vector calculated as the mean of all points assigned to it. The points are assigned to a supervoxel based on a similarity metric, D, calculated as the Euclidean distance between a point p_i and a cluster center C_k. Supervoxel segmentation algorithms have individual strategies to tackle outlying points. At the end of all iterations of our algorithm, we assign a label $k = 0$ to the outlier points to mark them as *unlabeled*.

The algorithm requires one input parameter, i.e., the number of supervoxels, K. Our proposed algorithm differs from VCCS, [21], in three aspects: (1) for cluster center initialization, instead of uniformly sampling the point cloud we exploit the surface geometry to identify important regions in the point cloud, (2) instead of projecting the points into lab color space, we propose a method to use the similarity metric in the RGB color space, and (3) we allow to add or remove supervoxels dynamically to ensure that the entire point cloud is covered by the over-segmentation. The following subsections detail the individual steps of the algorithm, i.e., cluster center initialization in Section III-A and, assignment and update steps in Section III-B.

A. INITIALIZATION

We propose a novel approach to select initial cluster seeds based on points' orientation while ensuring that they are not initialized close to one another. Like the strategy used in [8], we assume that supervoxels are regular in shape and estimate the sidelength of each supervoxel as $S = \sqrt{\frac{N}{K}}$. Two cluster centers placed close to one another will have a significant overlap in their search spaces. This results in competition for the same set of points in every iteration and the algorithm may never converge. Therefore, we first voxelate the point cloud with a voxel sidelength of S and select K voxels as

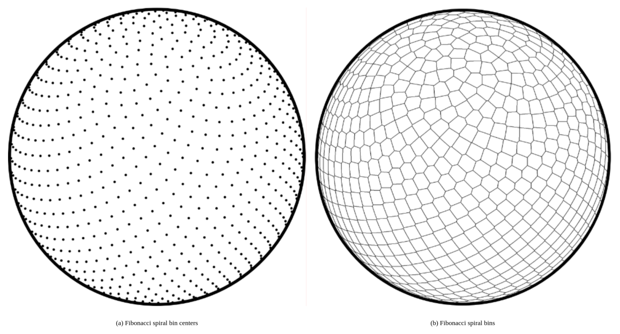

FIGURE 2. Fibonacci spiral bins on a unit sphere, [28]

initial cluster seeds. In the process of voxelation, a uniform grid is placed over the point cloud and the value of each grid cell (voxel) is given as the average of all the points that lie within that cell. Points from cloud that are spatially closest to the K selected voxel centers are chosen as the initial cluster seeds. While methods like vccs, [21], select all the voxels in the voxelized point cloud as initial cluster seeds, we propose a new point orientation-based histogram binning to select only K important points as seeds from all the voxels. This strategy ensures that we do not overfit the data and create only K supervoxels as specified by the user.

To identify geometrically important regions, we construct a histogram of the voxels' surface normals using the Fibonacci spiral binning technique as described in [28]. Fibonacci spiral binning works by creating a Fibonacci spiral on the sphere from the north to the south pole with each bin location placed at equal increments along the spiral. Fig. 2(a) shows an illustration of bin centers initialized along the Fibonacci spiral on a sphere. An illustration of Fibonacci bins is shown in Fig. 2(b). Authors in [28] argue that Fibonacci spiral produces uniformly distributed bins around the sphere when compared with other binning techniques (e.g., equiangle grid). This binning technique requires that an odd number of bins must be created to have an equal number of bins in the two hemispheres. If we want to create $b_n = K$ number of bins, then the bin centers are given in spherical coordinates as,

$$B_{[\theta,\phi]}(d) = \left[sin^{-1}\left(\frac{2d}{b_n}\right) + \frac{\pi}{2}, \frac{2\pi}{\tau} mod(d,\tau)\right], \quad (2)$$

where, θ and ϕ are the azimuthal and polar angles, respectively, $\tau = \frac{1+\sqrt{5}}{2}$ is the golden ratio, and $d \in \frac{1-b_n}{2}, \ldots, \frac{b_n-1}{2}$ is an integer used to represent the bin centers. The unit surface normals of all the voxels from the voxelated point cloud are first projected into spherical coordinates and then assigned to closest bin center by either a brute force approach or by a faster implementation as proposed in [28]. We use the implementation proposed in [28] as it is faster than the brute force approach especially when K is large.

For indoor scenes taken by one still camera (like the data used here), the surface normals lie within only one hemisphere as the normals point towards the camera. Therefore, for better binning accuracy in indoor point clouds, we create $b_n = 2K$ number of bins. After creating the histogram of surface normals, all the bin centers without any assigned normals are deleted and one voxel from each of the remaining bins (say we have b'_n bins with at least one assigned normal) is selected as a cluster seed. The remaining $K - b'_n$ cluster seeds are initialized at equal intervals in the remaining voxels across all bins. Through this strategy, we give more importance to the geometry of the scene than to the spatial distribution of points. Identifying important regions through binning of normals allows for greater representation of small distinct objects, while at the same time, there is a higher chance that large objects (e.g., walls) get multiple cluster seeds.

B. ALGORITHM

In the assignment step, the points p_i are assigned to the cluster center C_k, based on a distance metric D. D is computed as a combination of the Euclidean distances between the position, color, and normal vectors of a point p_i and a cluster center C_k. For a given point p_i and a cluster center C_k, $d_s = \sqrt{(x_i - x_k)^2 + (y_i - y_k)^2 + (z_i - z_k)^2}$ is the distance between position vectors and $d_n = \sqrt{(u_i - u_k)^2 + (v_i - v_k)^2 + (w_i - w_k)^2}$ is the distance between normal vectors. In [29], the authors propose a novel low-cost approximation for calculating the distance between two colors in RGB space directly. They cite subjective experiments to claim that their proposed formulation overcomes limitations of LUV color space. Moreover, as general datasets have color information given in RGB space, computations to convert from RGB space to LUV space can be avoided through this non-linear distance metric. We propose to calculate the color distance between a point and a cluster center as $d_c = \sqrt{f_r \cdot (r_i - r_k)^2 + f_g \cdot (g_i - g_k)^2 + f_b \cdot (b_i - b_k)^2}$, where, $f_r = 2 + \frac{r_m}{256}$, $f_g = 4$, and $f_b = 2 + \frac{255-r_m}{256}$ are weights for the respective colors, and $r_m = \frac{r_i+r_k}{2}$ is the mean of red color. The distance metric for comparing two points is then given as,

$$D = \sqrt{d_n^2 + \frac{1}{n_s} \cdot d_s^2 + \frac{1}{n_c} \cdot d_c^2}, \quad (3)$$

where, n_s and n_c are normalization factors for spatial and color distances, respectively. Only the points lying inside a spherical neighbourhood of radius S are processed for each cluster center. This implies that a point can be at the most S units from its cluster center. Therefore, we set $n_s = S$. Similarly, as the range of color values range in $[0, 256]$, we set $n_c = 256$. d_n does not require normalization as a point's orientation is given by unit surface normals.

The algorithm starts by going through each cluster center sequentially and searching a spherical neighbourhood of radius S around it. The encountered points are assigned to the cluster center if it is the smallest distance, D, the point has seen so far. After this operation, the outlying points (unassigned points) are collected (Let us call the count of outlying points as N_{ol}). As these point clouds are not dense enough to be considered as volumes, we can ignore the depth dimension of the points and assume that the approximate size of each supervoxel to be $c_{size} = \frac{N}{K}$. This estimated size of a supervoxel is used later in the update step of the algorithm to estimate the number of cluster centers to be added. By keeping c_{size} constant throughout the algorithm, we can keep the final number of supervoxels close to the user-specified value K.

In the update step all the cluster centers with the number of assigned points less than 10% of c_{size} are removed. If K_r cluster centers are removed, the remaining $K - K_r$ cluster centers are updated as the mean of all the points, p_i, assigned to each. After updating the cluster centers, additional cluster centers are initialized in the set of unassigned points. A new point cloud is created from the set of unassigned points and new clusters are initialized in it following the same procedure as described in Section III-A. The number of cluster centers that need to be initialized from the unassigned points is given as $c_{add} = \frac{N_{ol}}{c_{size}}$. The process of dynamically adding and/or removing cluster centers results in different number of clusters (say K') from the user selected value K. The actual value of the difference $K' - K$ depends on the spatial distribution of the points. However, the process of dynamic cluster seed initialization has two advantages: (1) small but geometrically distinct regions get their own cluster seeds, and (2) outlier points do not get incorrectly assigned to any clusters.

The assignment and update steps are repeated iteratively until, either the number of outliying points changes by less than 10% from the previous iteration, or if there are no more cluster centers required to be added. For example, when $N_{ol} < c_{size}$, c_{add}, as an integer division, becomes zero, implying that no additional cluster centers can be initialized. Our experiments show that the algorithm usually runs until there is no possibility of adding any more clusters. Hence, when the stopping criteria is satisfied, it is only the outlier points that remain unassigned. As a last step, the K' cluster centers are updated as the mean of all points assigned to each of them. By leaving the outlier points as unassigned, our algorithm achieves better accuracy and object boundary adherence as most datasets consist of a category of unlabeled points. The complete algorithm is detailed in Algorithm 1. The clustering algorithm has a time complexity of $\mathcal{O}(N)$.

Algorithm 1: Clustering based on point orientation

```
Input: K,$p_i = [x_i\ y_i\ z_i\ r_i\ g_i\ b_i\ u_i\ v_i\ w_i]^T\ \forall$
  $i = 1:N$;
Result: $labels_i\ \forall\ i = 1:N$, $C_k\ \forall\ k = 1:K'$
$S = \sqrt{\frac{N}{K}}$;
Initialize: $C_k = [x_k\ y_k\ z_k\ r_k\ g_k\ b_k\ u_k\ v_k\ w_k]^T\ \forall$
  $k = 1:K$;
$D_i = \inf$, $labels_i = -1\ \forall\ i = 1:N$;
$K' = K, t = 0, N_{ol}(t) = N, c_{size} = \frac{N}{K}$;
while (1) do
    for $k = 1$ to $K'$ do
        for $p_i$ in neighbourhood of radius $S$ do
            D = D($p_i, C_k$) as in equation 3;
            if $D < D_i$ then
                $D_i = D$;
                $labels_i = k$;
            end
        end
    end
    outliers = collect points with label == $-1$;
    $N_{ol}(t+1)$ = size(outliers);
    remove all centers with $size(C_k) < 0.1 \times c_{size}$;
    Re-estimate $C_k$ as mean of all $p_i$ with
      $labels_i == k$;
    $c_{add} = \frac{N_{ol}(t+1)}{c_{size}}$;
    if $(\frac{N_{ol}(t) - N_{ol}(t+1)}{N_{ol}(t)} < 0.1)$ || $(c_{add} < 1)$ then
        break;
    end
    Initialize $c_{add}$ clusters and append to $C_k$;
    $K' = K$ after adding and/or removing clusters;
    $t = t + 1$;
end
```

IV. RESULTS

We test the proposed clustering method's efficacy on the openly available NYU-V2 Depth dataset, [9]. The dataset consists of 1449 densely labelled pairs of aligned RGB and depth images. The aligned depth information was mapped to corresponding pixels to obtain 3D point clouds with color information in RGB space. The proposed method is compared against three state-of-the-art clustering algorithms: (1) the original voxel cloud connectivity segmentation (vccs) method that works on voxelated point clouds, [21], (2) a supervoxel segmentation method framed as a subset selection problem (ssp), [22], and (3) a K-nearest-neighbours version of vccs method (vccs-knn) that works directly on the point clouds without voxelation, provided by the authors in [22]. The vccs method is available as part of PCL (Point Cloud Library), [30], and it was tested using the default parameter settings. Voxel resolution for VCCS method was set at 0.1m for all experiments. The ssp and vccs-knn methods were tested using their openly available source code[1] with the parameters for both methods set to the values as proposed in their article, [22]. While the vccs, ssp and vccs-knn methods implicitly compute the surface normals for the point clouds, for our method they were computed using the standard nearest-neighbours-based method provided by PCL with number of neighbours equal to 30. Our method requires only one input parameter, i.e., the number of clusters K. All the experiments were run on a computer with an Intel Core i7-8700K CPU and 16GB of RAM. Our method is coded in C++ programming language.

[1]https://github.com/yblin/Supervoxel-for-3D-point-clouds.git

We compare the performance of the algorithms using four evaluation metrics. For evaluations, we first represent the labelled point clouds as 2D labeled images and compare them against the labeled 2D ground truth images. Boundary recall (R) measures the fraction of ground truth boundaries that fall within a distance ϵ of at least one estimated supervoxel boundary. We follow the definition of boundary recall as proposed in [31]. Given a ground truth boundary image G and an estimated boundary image B, R is computed as the fraction of true positives (TP) and the sum of true positives and true negatives (TN), i.e., $R = \frac{TP}{TP+TN}$, where TP are defined as the number of boundary pixels in G for whose exist a boundary pixel in B in range ϵ, and TN as the number of boundary pixels in G for whose do not exist a boundary pixel in B in range ϵ. In this article, we use $\epsilon = 2$ pixels. An R value of 1 reflects best performance indicating the methods precision in identifying object boundaries whereas, a value of 0 reflects otherwise. The second metric we use is the undersegmentation error (UE). According to [31], UE measure to what extent estimated segmentation boundaries cross-over the ground truth boundaries. Generally, the number of pixels of a segment that cross over the boundary are measured. However, this method imposes a high penalty on large supervoxels with only a small overlap with the ground truth segment. To avoid this, In [31], they propose a new method where the smaller value of either the region that crosses over the boundary or the region that lies within the segment is counted depending on whichever is smaller. It is defined as,

$$UE = \frac{1}{N}\left[\sum_{S\in GT}\left(\sum_{P:P\cap S\neq 0} min(P_{in}, P_{out})\right)\right], \quad (4)$$

where, S are the ground truth (GT) segments, P are the estimated segments, N is the total number of pixels, P_{in} is the part of the estimated segment that lies within S and P_{out} is the part of the estimated segment that crosses over the ground truth segment's boundary. A UE value of 0 implies that the method has best adherence to object or segment boundaries whereas, $UE = 1$ indicates otherwise.

The third metric we use is the compactness (C) of the supervoxels. In mathematics, compactness of a shape is commonly measured through the isoperimetric quotient which compares the area of a shape to the area of a circle with the same perimeter as this shape, [32]. If A_P is the area and L_P is the perimeter of a superpixel (supervoxel projected in 2D), P, then the radius of a circle with the same perimeter as P is given as, $r = \frac{L_P}{2\pi}$. If A_S is the area of the circle with radius r, then the isoperimetric quotient is given as,

$$Q_P = \frac{A_P}{A_S} = \frac{4\pi A_P}{L_P^2}\,. \quad (5)$$

Therefore, if I is the set of all segments in a segmented image, then the compactness measure is given as,

$$C = \sum_{P\in I} Q_P \cdot \frac{|P|}{N}\,, \quad (6)$$

where, $|P|$ is the size of the segment and N is the total number of pixels in the image (or points in the point cloud). $C = 1$ implies that the estimated segments are perfect circles whereas, $C = 0$ implies that the segments have highly irregular and non-convex shapes. Lastly, we also compare the algorithms based on the contour density (CD) metric, [11]. CD measures the fraction of boundary pixels in the segmentation image. Given a set of boundary pixels, B, of an estimated segmentation contour density is defined as, $CD = \frac{|B|}{2N}$, where N is the total number of pixels. The fraction is divided by two because computation of segment boundaries produces edges that are two-pixels wide. The contour density metric also indicates regularity of the boundaries as higher values of CD mean that there exist more number of boundary pixels for the same number of supervoxels. Higher values of CD indicate that the object boundaries are irregular and elongated.

A. EVALUATION

All the above-mentioned algorithms were tested on the NYU V2 Depth dataset and compared using the above-mentioned metrics. While supervoxel resolution as a measure has geometrical significance, we believe that the number of clusters is easier to interpret for a general user. The complexities of all algorithms is given in terms of the input size. As a result, a user can have better control on estimating the complexity of subsequent algorithms which would be used on the segmented point cloud when there is a direct control on the number of supervoxels that need to be produced in a point cloud over-segmentation. Moreover, it is important to note that each algorithm produces a different number of supervoxels for a point cloud at any given supervoxel resolution. In all cases, ssp method produces the highest number of supervoxels between all the algorithms for any given resolution. It is well known that as the number of supervoxels increases the performance in terms of metrics also increases. Due to this reason, as one method produces more supervoxels than another for the same segmentation, comparison by the supervoxel resolution becomes an unfair comparison for methods that produce fewer number of supervoxels. Therefore, we choose to compare results based on the number of supervoxels produced.

We tested the vccs, ssp and vccs-knn algorithms at supervoxel resolutions (in meters) 0.10, 0.11, 0.12, 0.13, 0.14, 0.16, 0.18, 0.20, 0.25 and 0.35. To keep the number of supervoxels produced by our method in similar range as other methods, we set the output number from vccs method as the input number of clusters for our method. Although the number of clusters produced by our method is dynamic, it is usually within a range of ± 100 clusters from the user-chosen number K. To maintain uniformity in comparison, for each experiment, we round the output number of clusters to the nearest 100 and order the results according to these multiples. We believe that this strategy allows for a fair comparison as results from some point clouds may get rounded to higher multiple of hundred while a similar number of point cloud

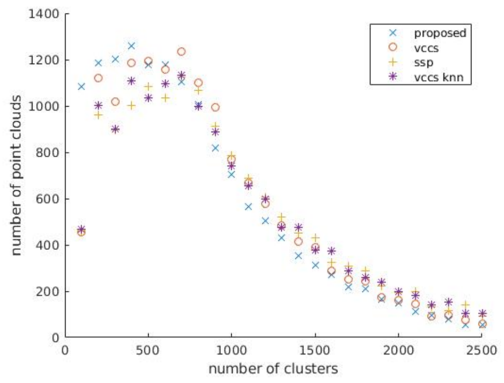

FIGURE 3. histogram distribution of number of point clouds for which results were obtained, binned according to the number of clusters.

results may get rounded to a lower multiple of hundred. As the number of supervoxels found varies with the method, density, or spatial spread of the point cloud, and the supervoxel resolution, it is not possible to have a control on how many point clouds produce results for a given number of clusters. A histogram of the number of point clouds, ordered by the cluster entries for which results were obtained, is shown in Fig. 3. We present results only when there exist at least 30 point clouds contributing to the results for a given number of clusters entry.

Quantitative results of the four metrics obtained from testing the four methods on the entire 1449 image pairs of the dataset are shown in Fig. 4. Overall, our proposed method performs best in the undersegmentation error and compactness metrics while the ssp and vccs methods perform best in the boundary recall and contour density metrics, respectively. The ssp method and our proposed clustering method perform similarly well in R and UE metrics. With a relative difference of 0.0086 between both the methods' overall mean R for all the tests, the ssp method performs marginally better than our method in the boundary recall metric. Whereas, in the UE metric, our method is relatively 3.17% better than ssp method. Moreover, a student's t-test on the results from our method and ssp shows that the means of UE of our method are significantly better than those of ssp, with a $p-value = 1.32e-6$ and a $t-value = 6.38$ in the confidence interval $CI = [0.0070, 0.0136]$. As evident from Fig. 4(a), both these methods show significantly better boundary adherence (significantly better performance in both R and UE metrics) than both the variants of vccs method. The major advantage of our method lies in the regularly shaped supervoxels that it produces. The plot in Fig. 4(c) shows that our method significantly outperforms the three other methods in terms of producing most regularly shaped and compact supervoxels. The ssp method performs worst in this metric as their method produces supervoxels with highly irregular boundaries. This quality of ssp method is also reflected in the CD metric (see Fig. 4), where ssp method performs worst out of all the methods. Large values of CD for ssp method, or a large number of boundary pixels for a given number of supervoxels, reflects the irregularly shaped supervoxels produced by it.

It is to be noted that a higher number of boundary pixels may result in artificially inflated values of R as there is a greater chance that a segmentation boundary lies in close distance to a given ground truth boundary. Therefore, it is possible that the ssp method may not produce results that are visually as appealing as reflected by their quantitative analysis values. The same can be verified from Fig. 5, where it can be seen that although the supervoxels produced by ssp method agree to object boundaries to an extent, the resulting supervoxels are irregular in shape. Irregularly shaped supervoxels are undesirable for subsequent applications as it introduces spatial discontinuity within the segmentation. With the least CD values of all methods, the vccs method can be expected to produce the most compact supervoxels. Although it produces more regular shaped supervoxels than ssp and vccs-knn methods, our method outperforms it because the vccs method fails on noisy point clouds or on point clouds with low spatial density (or high variation in depth). The same can be verified visually from the second row of Fig. 5(b), where there exist empty regions (seen in white) and small isolated supervoxels in the segmentation. While missing regions or small and isolated supervoxels do not contribute to the CD, they impose a serious penalty on the compactness metric, thereby resulting in lower values of C for vccs method.

Fig. 5 shows visualizations of segmentations from the four methods on some example point clouds. In general, our method produces visually appealing segmentations with compact and regularly shaped supervoxels that addhere to object boundaries well. An exception where our method fails to produce compact supervoxels can be seen in the second row of Fig. 5. The mesh doors at vicinity of the viewpoint act as noise in that region of the point cloud as they appear as scattered points at a different depth than the background, resulting in irregular supervoxels. The superior compactness of supervoxels produced by our method is visible in the fourth row of Fig. 5. The highlighted region consists of planar regions with very few objects. Yet, all methods except ours produced supervoxels with arbitrary shapes in that region. The shown segmentations are for 500, 700, 2000 and 1100 supervoxels for the first to fourth rows, respectively. The result for vccs-knn method in the third row of the figure consists of only 800 supervoxels as the method produced only those many at a resolution of 0.1m, while the rest of the methods produced about 2000 supervoxels. Finally, it can be said that this figure presents an accurate visual reiteration of the results shown as part of quantitative analysis. Visualizations show that the vccs method fails in the case of point clouds with complex scenes and when the point clouds have varying depth and points density. The vccs-knn method produces supervoxels with irregular shapes, while the ssp method also produces irregularly shaped suypervoxels with elongated boundaries but with greater accuracy. Our method

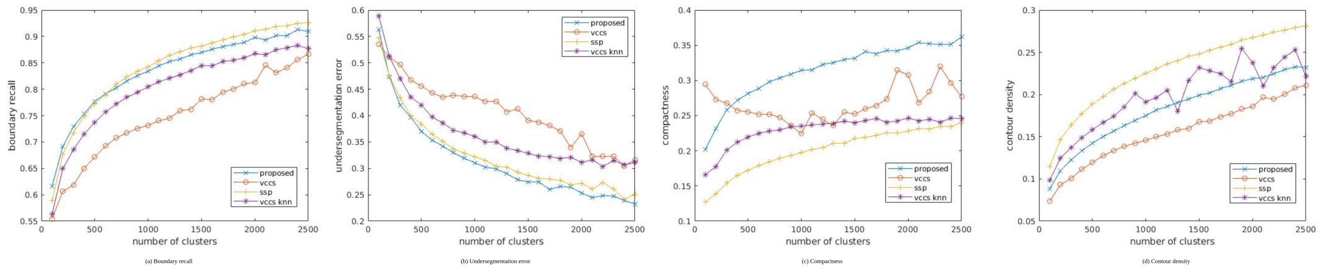

FIGURE 4. Quantitative analysis of the four methods on NYU Depth V2 dataset.

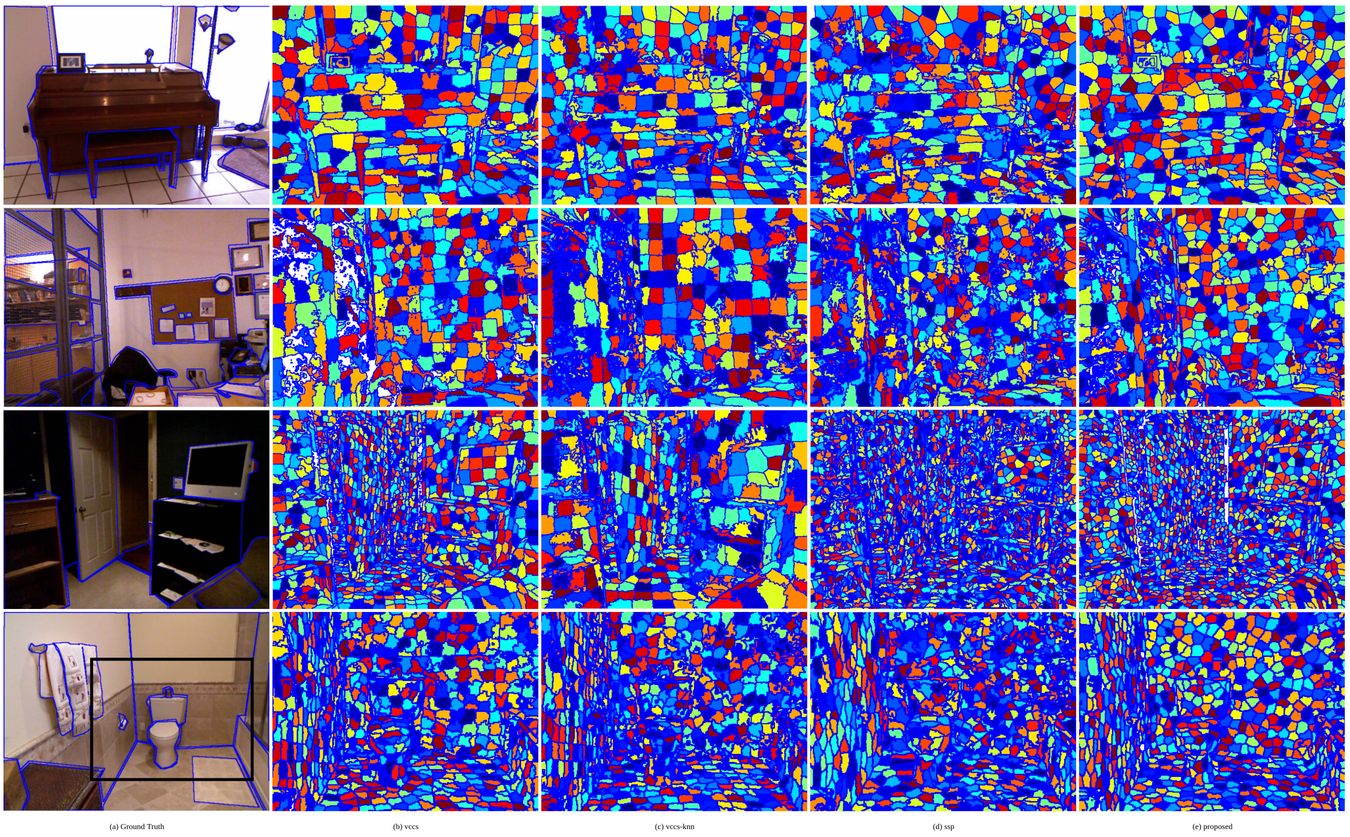

FIGURE 5. Visual representation of segmentation results on NYU Depth V2 dataset.

produces compact supervoxels with clear and tight boundaries that addhere well to the scene, for a range of supervoxel resolutions (100 supervoxels to over 2000 supervoxels) with some exceptions as shown for the point cloud in the second row of 5.

A comparison of the total running times by vccs, ssp and our method are shown in Fig. 6. In terms computational time, the vccs method proves its simplicity and robustness as it performs segmentation in about $1s$. While all methods have constant complexity, our method, with a complexity of $\mathcal{O}(N)$, requires significantly longer computational time when compared against the other methods. This is expected as our algorithm involves the assignment step for all the points in every iteration. The step of collecting unassigned points and initializing and/or removing cluster centers adds additional computational overhead. However, our proposed initialization procedure based on Fibonacci binning produces accurate initial seeds. The algorithm typically converged in $6 - 8$ iterations in all cases. While the overall complexity of our algorithm seems big when compared to other methods, the gained accuracy and quality of segmentation may be beneficial to applications that do not have serious limitations on computational time but, which may benefit from accurate and compact supervoxel segmentation. Performing the assignment step only on the points at supervoxel boundaries or maintaining a database of initially unassigned points and working only on that set of points iteratively, may benefit the algorithm in terms of speed.

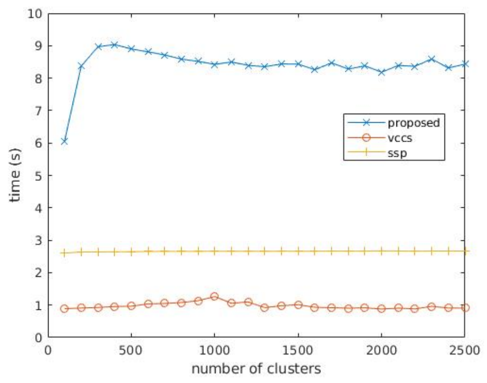

FIGURE 6. Time taken for segmentation by the vccs, ssp and our proposed methods on NYU Depth V2 dataset.

V. CONCLUSION AND FUTURE WORK

We proposed a new supervoxel segmentation method with dynamic cluster seed initialization. Our method inherits the advantages of the k-means algorithm. Coupled with a novel cluster seed initialization strategy based on Fibonacci binning of surface normals, the method achieves superior boundary adherence when compared against existing state-of-the-art methods. As the algorithm is parameter-free, it gives equal importance to the spatial location, color, and surface normals of all points to produce regularly shaped compact supervoxels with tight boundaries. Quantitative analysis using four metrics on the publicly available NYU Depth V2 dataset shows that our method performs equally good as the ssp and vccs methods in the boundary recall and contour density metrics, respectively. Our proposed method shows significantly better performance than all other methods in the undersegmentation error and compactness metrics. Visual representations of segmentation results on some of the point clouds show that our method produces visually appealing supervoxels with a high degree of compactness that adhere well to object boundaries.

However, the added accuracy of our proposed algorithm comes at the cost of increased complexity. For future work, we propose to explore the possibility of reducing the overall computational time of our algorithm. Calculating the distance metric in the assignment step for only the points at supervoxel boundaries and maintaining a database of unassigned points and working iteratively only on that set of points can be approaches to improve the time complexity of our algorithm. Another improvement to decrease the number of calculations per iteration could be to stop the assignment step after meeting a certain criterion. An example criterion could be to track the change in cluster seeds based on the $\mathcal{L}^1$-norm between cluster centers at current and previous iterations and stop the assignment step when the norm becomes lower than threshold.

REFERENCES

[1] Y. Xu, Z. Ye, W. Yao, R. Huang, X. Tong, L. Hoegner, and U. Stilla, "Classification of lidar point clouds using supervoxel-based detrended feature and perception-weighted graphical model," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 13, pp. 72–88, 2019.

[2] H. Luo, K. Khoshelham, L. Fang, and C. Chen, "Unsupervised scene adaptation for semantic segmentation of urban mobile laser scanning point clouds," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 169, pp. 253–267, 2020.

[3] S.-S. Huang, Z.-Y. Ma, T.-J. Mu, H. Fu, and S.-M. Hu, "Supervoxel convolution for online 3d semantic segmentation," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 3, pp. 1–15, 2021.

[4] J. Huo, J. Wu, J. Cao, and G. Wang, "Supervoxel based method for multi-atlas segmentation of brain mr images," *NeuroImage*, vol. 175, pp. 201–214, 2018.

[5] S. Hansen, S. Kuttner, M. Kampffmeyer, T.-V. Markussen, R. Sundset, S. K. Øen, L. Eikenes, and R. Jenssen, "Unsupervised supervoxel-based lung tumor segmentation across patient scans in hybrid pet/mri," *Expert Systems with Applications*, vol. 167, p. 114244, 2021.

[6] H. Guan, Y. Yu, J. Li, and P. Liu, "Pole-like road object detection in mobile lidar data via supervoxel and bag-of-contextual-visual-words representation," *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 4, pp. 520–524, 2016.

[7] J.-S. Yun and J.-Y. Sim, "Supervoxel-based saliency detection for large-scale colored 3d point clouds," in *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2016, pp. 4062–4066.

[8] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

[9] P. K. Nathan Silberman, Derek Hoiem and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *ECCV*, 2012.

[10] V. A. Puligandla and S. Lončarić, "A multiresolution approach for large real-world camera placement optimization problems," *IEEE Access*, vol. 10, pp. 61 601–61 616, 2022.

[11] V. Machairas, M. Faessel, D. Cárdenas-Peña, T. Chabardes, T. Walter, and E. Decenciere, "Waterpixels," *IEEE Transactions on Image Processing*, vol. 24, no. 11, pp. 3707–3716, 2015.

[12] A. Levinshtein, A. Stere, K. N. Kutulakos, D. J. Fleet, S. J. Dickinson, and K. Siddiqi, "Turbopixels: Fast superpixels using geometric flows," *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 12, pp. 2290–2297, 2009.

[13] M. V. d. Bergh, X. Boix, G. Roig, B. d. Capitani, and L. V. Gool, "Seeds: Superpixels extracted via energy-driven sampling," in *European conference on computer vision*. Springer, 2012, pp. 13–26.

[14] A. P. Moore, S. J. Prince, J. Warrell, U. Mohammed, and G. Jones, "Superpixel lattices," in *2008 IEEE conference on computer vision and pattern recognition*. IEEE, 2008, pp. 1–8.

[15] C. Xu and J. J. Corso, "Evaluation of super-voxel methods for early video processing," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 1202–1209.

[16] O. Veksler, Y. Boykov, and P. Mehrani, "Superpixels and supervoxels in an energy optimization framework," in *European conference on Computer vision*. Springer, 2010, pp. 211–224.

[17] D. Weikersdorfer, A. Schick, and D. Cremers, "Depth-adaptive supervoxels for rgb-d video segmentation," in *2013 IEEE International Conference on Image Processing*. IEEE, 2013, pp. 2708–2712.

[18] G. Gao, M. Lauri, J. Zhang, and S. Frintrop, "Saliency-guided adaptive seeding for supervoxel segmentation," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 4938–4943.

[19] P. Xu, J. Li, J. Yue, and X. Yuan, "Scale adaptive supervoxel segmentation of rgb-d image," in *2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2016, pp. 1303–1308.

[20] Y. Zhou, L. Ju, and S. Wang, "Multiscale superpixels and supervoxels based on hierarchical edge-weighted centroidal voronoi tessellation," *IEEE Transactions on Image Processing*, vol. 24, no. 11, pp. 3834–3845, 2015.

[21] J. Papon, A. Abramov, M. Schoeler, and F. Worgotter, "Voxel cloud connectivity segmentation-supervoxels for point clouds," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 2027–2034.

[22] Y. Lin, C. Wang, D. Zhai, W. Li, and J. Li, "Toward better boundary preserved supervoxel segmentation for 3d point clouds," *ISPRS journal of photogrammetry and remote sensing*, vol. 143, pp. 39–47, 2018.

[23] Z. Sha, Q. Zhu, Y. Chen, C. Wang, A. Nurunnabi, and J. Li, "A boundary-enhanced supervoxel method for 3d point clouds," in *IGARSS 2020-2020 IEEE International Geoscience and Remote Sensing Symposium*. IEEE, 2020, pp. 2771–2774.

[24] X. Dong, Y. Xiao, Z. Chen, J. Yao, and X. Guo, "Gpu-based supervoxel segmentation for 3d point clouds," *Computer Aided Geometric Design*, vol. 93, p. 102080, 2022.

[25] H. Ni and X. Niu, "Svla: A compact supervoxel segmentation method based on local allocation," *Isprs journal of photogrammetry and remote sensing*, vol. 163, pp. 300–311, 2020.

[26] L. Landrieu and M. Boussaha, "Point cloud oversegmentation with graph-structured deep metric learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7440–7449.

[27] L. Hui, J. Yuan, M. Cheng, J. Xie, X. Zhang, and J. Yang, "Superpoint network for point cloud oversegmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5510–5519.

[28] R. L. Larkins, M. J. Cree, and A. A. Dorrington, "Analysis of binning of normals for spherical harmonic cross-correlation," in *Three-Dimensional Image Processing (3DIP) and Applications II*, vol. 8290. International Society for Optics and Photonics, 2012, p. 82900L.

[29] T. Riemersma, "color metric," 2019. [Online]. Available: https://www.compuphase.com/cmetric.htm

[30] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 1–4.

[31] F. Puente León, *Forum Bildverarbeitung 2012*. KIT Scientific Publishing, 2012.

[32] A. Schick, M. Fischer, and R. Stiefelhagen, "Measuring and evaluating the compactness of superpixels," in *Proceedings of the 21st international conference on pattern recognition (ICPR2012)*. IEEE, 2012, pp. 930–934.

V. ANIRUDH PULIGANDLA was born in Hyderabad, Telangana, India in 1992. He received B.Tech degree in electronics and communications engineering from Amity University, Jaipur, Rajasthan, India in 2014 and B.Sc. degree in computer vision and robotics from University of Burgundy, France in 2016. He received M.Sc. degree in computer vision and robotics from University of Burgundy, France, University of Girona, Spain and Heriot Watt university, Scotland in 2018 as part of an Erasmus Mundus Joint Masters degree program. He is currently pursuing a Ph.D. degree program at University of Zagreb, Zagreb, Croatia under a Marie-Curie Actions ITN fellowship. His research interests include discrete and continuous optimization, signal and image processing, 3D reconstruction from multiple camera systems using multi-view stereo.

DR. SVEN LONČARIĆ is a professor of electrical engineering and computer science at the Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia. As a Fulbright scholar, he received a Ph.D. degree in electrical engineering from the University of Cincinnati, OH in 1994. From 2001-2003, he was an assistant professor at the New Jersey Institute of Technology, USA. His areas of research interest are image processing and computer vision. He was the principal investigator on a number of R&D projects. Prof. Lončarić co-authored more than 250 publications in scientific journals and conferences. He is the director of the Center for Computer Vision at the University of Zagreb and the head of the Image Processing Laboratory. He is a co-director of the Center of Excellence in Data Science and Cooperative Systems. He is a member of the Croatian Academy of Sciences and Arts and a senior member of IEEE. Prof. Lončarić received several awards for his scientific and professional work.