Voxel Cloud Connectivity Segmentation - Supervoxels for Point Clouds

Jeremie Papon   Alexey Abramov   Markus Schoeler
Florentin Wörgötter
Bernstein Center for Computational Neuroscience (BCCN)
III Physikalisches Institut - Biophysik, Georg-August University of Göttingen
{jpapon,abramov,mschoeler,worgott}@physik3.gwdg.de

Abstract

Unsupervised over-segmentation of an image into regions of perceptually similar pixels, known as superpixels, is a widely used preprocessing step in segmentation algorithms. Superpixel methods reduce the number of regions that must be considered later by more computationally expensive algorithms, with a minimal loss of information. Nevertheless, as some information is inevitably lost, it is vital that superpixels not cross object boundaries, as such errors will propagate through later steps. Existing methods make use of projected color or depth information, but do not consider three dimensional geometric relationships between observed data points which can be used to prevent superpixels from crossing regions of empty space. We propose a novel over-segmentation algorithm which uses voxel relationships to produce over-segmentations which are fully consistent with the spatial geometry of the scene in three dimensional, rather than projective, space. Enforcing the constraint that segmented regions must have spatial connectivity prevents label flow across semantic object boundaries which might otherwise be violated. Additionally, as the algorithm works directly in 3D space, observations from several calibrated RGB+D cameras can be segmented jointly. Experiments on a large data set of human annotated RGB+D images demonstrate a significant reduction in occurrence of clusters crossing object boundaries, while maintaining speeds comparable to state-of-the-art 2D methods.

1. Introduction

Segmentation algorithms aim to group pixels in images into perceptually meaningful regions which conform to object boundaries. While they initially only considered low-level information from the image, recent semantic segmentation methods take advantage of high-level object knowledge to help disambiguate object borders. Graph-based approaches, such as Markov Random Field (MRF) and Condi-

tional Random Field (CRF), have become popular, as they merge relational low-level context within the image with object level class knowledge. While the use of such techniques have met with significant success, they have the drawback that the computational cost of inference on these graphs generally rises sharply with increasing number of nodes. This means that solving graphs with a node for every pixel quickly becomes intractable, which has limited their use in applications which require real-time segmentation.

The cost of solving pixel-level graphs led to the development of mid-level inference schemes which do not use pixels directly, but rather use groupings of pixels, known as superpixels, as the base level for nodes [9]. Superpixels are formed by over-segmenting the image into small regions based on local low-level features, reducing the number of nodes which must be considered for inference. While this scheme has been successfully used in many state-of-the-art algorithms [4, 15], it suffers from one significant disadvantage; mistakes in the over-segmentation which creates the superpixels generally cannot be recovered from and will propagate to later steps in the vision pipeline.

Due to their strong impact on the quality of the eventual segmentation [5], it is important that superpixels have certain characteristics. Of these, avoiding violating object boundaries is the most vital, as failing to do so will decrease the accuracy of classifiers used later - since they will be forced to consider pixels which belong to more than one class. Additionally, even if the classifier does manage a correct output, the final pixel level segmentation will necessarily contain errors. Another useful quality is regular distribution over the area being segmented, as this will produce a simpler graph for later steps.

In this paper, we present a novel method, Voxel Cloud Connectivity Segmentation (VCCS), which takes advantage of 3D geometry provided by RGB+D cameras to generate superpixels which conform to object boundaries better than existing methods, and which are evenly distributed in the actual observed space, rather than the projected image plane. This is accomplished using a seeding method-

ology based in 3D space and a flow-constrained local iterative clustering which uses color and geometric features. In addition to providing superpixels which conform to real geometric relationships, the method also can be used directly on point clouds created by combining several calibrated RGB+D cameras, providing a full 3D supervoxel (the 3D analogue of superpixels) graph at speeds sufficient for robotic applications. Additionally, the method source code is freely distributed as part of the Point Cloud Library [11] (PCL) [1].

The organization of the paper is as follows: first, in Section 2 we give an overview of existing methods. In Section 3 we present the 3D supervoxel segmentation algorithm. In Section 4 we present a qualitative evaluation of the method segmenting 3D point clouds created by merging several cameras. In Section 5 we use standard quantitative measures on results from a large RGB+D semantic segmentation dataset to demonstrate that our algorithm conforms to real object boundaries better than other state-of-the-art methods. Additionally, we present run-time performance results to substantiate the claim that our method is able to offer performance equivalent to the fastest 2D methods. Finally, in Section 6 we discuss the results and conclude.

2. Related Work

There are many existing methods for over-segmenting images into superpixels. These can be generally classified into two subsets - graph-based and gradient ascent methods. In this section, we shall briefly review recent top-performing methods.

Graph-based superpixel methods, similar to graph-based full segmentation methods, consider each pixel as a node in a graph, with edges connecting to neighboring pixels. Edge weights are used to characterize similarity between pixels, and superpixel labels are solved for by minimizing a cost function over the graph. Moore *et al.* [8] produce superpixels which conform to a regular lattice structure by seeking optimal paths horizontally and vertically across a boundary image. This is done using either a graph cuts or dynamic programming method which seeks to minimize the cost of edges and nodes in the paths. While this method does have the advantage of producing superpixels in a regular grid, it sacrifices boundary adherence to so, and furthermore, is heavily dependent on the quality of the pre-computed boundary image.

The Turbopixels [7] method of Levinshtein *et al.* uses a geometric flow-based algorithm based on level-set, and enforces a compactness constraint to ensure that superpixels have regular shape. Unfortunately, it is too slow for use in many applications; while the authors claim complexity linear in image size, in practice we experienced run times

---

over 10 seconds for VGA-sized images. Veksler *et al.* [13], inspired by Turbopixels, use an energy minimization framework to stitch together image patches, using graph-cuts to optimize an explicit energy function. Their method (referred to here as GCb10) is considerably faster than Turbopixels, but still requires several seconds even for small images.

Recently, a significantly faster class of superpixel methods has emerged - Simple Linear Iterative Clustering[1] (SLIC). This is an iterative gradient ascent algorithm which uses a local k-means clustering approach to efficiently find superpixels, clustering pixels in the five dimensional space of color and pixel location. Depth-Adaptive Superpixels[14] recently extended this idea to use depth images, expanding the clustering space with the added dimensions of depth and point normal angles. While DASP is efficient and gives promising results, it does not take full advantage of RGB+D data, remaining in the class of 2.5D methods, as it does not explicitly consider 3D connectivity or geometric flow.

For the sake of clarity, we should emphasize that our method is not related to existing "supervoxel" methods [1, 8, 13], which are simple extensions of 2D algorithms to 3D volumes. In such methods, video frames are stacked to produce a structured, regular, and solid volume with time as the depth dimension. In contrast, our method is intended to segment actual volumes in space, and makes heavy use of the fact that such volumes are not regular or solid (most of the volume is empty space) to aid segmentation. Existing "supervoxel" methods cannot work in such a space, as they generally only function on a structured lattice.

3. Geometrically Constrained Supervoxels

In this Section we present Voxel Cloud Connectivity Segmentation (VCCS), a new method for generating superpixels and supervoxels from 3D point cloud data. The supervoxels produced by VCCS adhere to object boundaries better than state-of-the-art methods while the method remains efficient enough to use in online applications. VCCS uses a variant of k-means clustering for generating its labeling of points, with two important constraints:

1. The seeding of supervoxel clusters is done by partitioning 3D space, rather than the projected image plane. This ensures that supervoxels are evenly distributed according to the geometry of the scene.

2. The iterative clustering algorithm enforces strict spatial connectivity of occupied voxels when considering points for clusters. This means that supervoxels strictly cannot flow across boundaries which are disjoint in 3D space, even though they are connected in the projected plane.

First, in 3.1 we shall describe how neighbor voxels are calculated efficiently, then in 3.2 how seeds are generated and filtered, in 3.3 the features and distance measure used

for clustering, and finally in 3.4 how the iterative clustering algorithm enforces spatial connectivity. Unless otherwise noted, all processing is being performed in the 3D point-cloud space constructed from one or more RGB+D cameras (or any other source of point-cloud data). Furthermore, because we work exclusively in a voxel-cloud space (rather than the continuous point-cloud space), we shall adopt the following notation to refer to voxel at index i within voxel-cloud V of voxel resolution r:

$$V_r(i) = \mathbf{F}_{1..n}, \tag{1}$$

where $\mathbf{F}$ specifies a feature vector which contains n point features (*e.g.* color, location, normals).

3.1. Adjacency Graph

Adjacency is a key element of the proposed method, as it ensures that supervoxels do not flow across object boundaries which are disconnected in space. There are three definitions of adjacency in a voxelized 3D space; 6-,18-, or 26-adjacent. These share a face, faces or edges, and faces, edges, or vertices, respectively. In this work, whenever we refer to adjacent voxels, we are speaking of 26-adjacency.

As a preliminary step, we construct the adjacency graph for the voxel-cloud. This can be done efficiently by searching the voxel kd-tree, as for a given voxel, the centers of all 26-adjacent voxels are contained within $\sqrt{3} * R_{voxel}$. R_{voxel} specifies the voxel resolution which will be used for the segmentation (for clarity, we shall simply refer to discrete elements at this resolution as voxels). The adjacency graph thus constructed is used extensively throughout the rest of the algorithm.

3.2. Spatial Seeding

The algorithm begins by selecting a number of seed points which will be used to initialize the supervoxels. In order to do this, we first divide the space into a voxelized grid with a chosen resolution R_{seed}, which is significantly higher than R_{voxel}. The effect of increasing the seed resolution R_{seed} can be seen in Figure 2. Initial candidates for seeding are chosen by selecting the voxel in the cloud nearest to the center of each occupied seeding voxel.

Once we have candidates for seeding, we must filter out seeds caused by noise in the depth image. This means that we must remove seeds which are points isolated in space (which are likely due to noise), while leaving those which exist on surfaces. To do this, we establish a small search radius R_{search} around each seed, and delete seeds which do not have at least as many voxels as would be occupied by a planar surface intersecting with half of the search volume (this is shown by the green plane in Figure 1). Once filtered, we shift the remaining seeds to the connected voxel within the search volume which has the smallest gradient in the
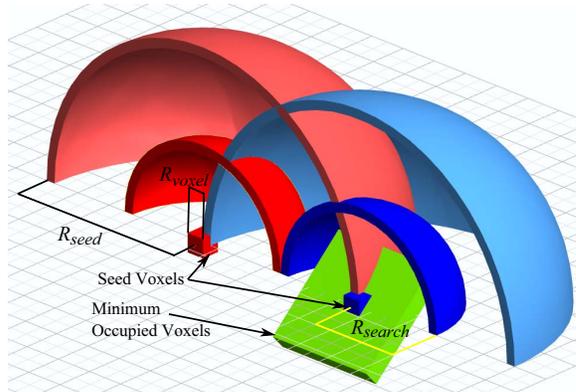

Figure 1. Seeding parameters and filtering criteria. R_{seed} determines the distance between supervoxels, while R_{voxel} determines the resolution to which the cloud is quantized. R_{search} is used to determine if there are a sufficient number of occupied voxels to necessitate a seed.

search volume. Gradient is computed as

$$G(i) = \sum_{k \in V_{adj}} \frac{\| V(i) - V(k) \|_{CIELab}}{N_{adj}}; \tag{2}$$

we use sum of distances in CIELAB space from neighboring voxels, requiring us to normalize the gradient measure by number of connected adjacent voxels N_{adj}. Figure 1 gives an overview of the different distances and parameters involved in seeding.

Once the seed voxels have been selected, we initialize the supervoxel feature vector by finding the center (in feature space) of the seed voxel and connected neighbors within 2 voxels.

3.3. Features and Distance Measure

VCCS supervoxels are clusters in a 39 dimensional space, given as

$$\mathbf{F} = [x, y, z, L, a, b, \text{FPFH}_{1..33}], \tag{3}$$

where x, y, z are spatial coordinates, L, a, b are color in CIELab space, and $\text{FPFH}_{1..33}$ are the 33 elements of Fast Point Feature Histograms (FPFH), a local geometrical feature proposed by Rusu *et al.* [10]. FPFH are pose-invariant features which describe the local surface model properties of points using combinations of their *k* nearest neighbors. They are an extension of the older Point Feature Histograms optimized for speed, and have a computational complexity of $O(n \cdot k)$.

In order to calculate distances in this space, we must first normalize the spatial component, as distances, and thus their relative importance, will vary depending on the seed resolution R_{seed}. Similar to the work of Achanta *et al.*, [1] we have limited the search space for each cluster so that it

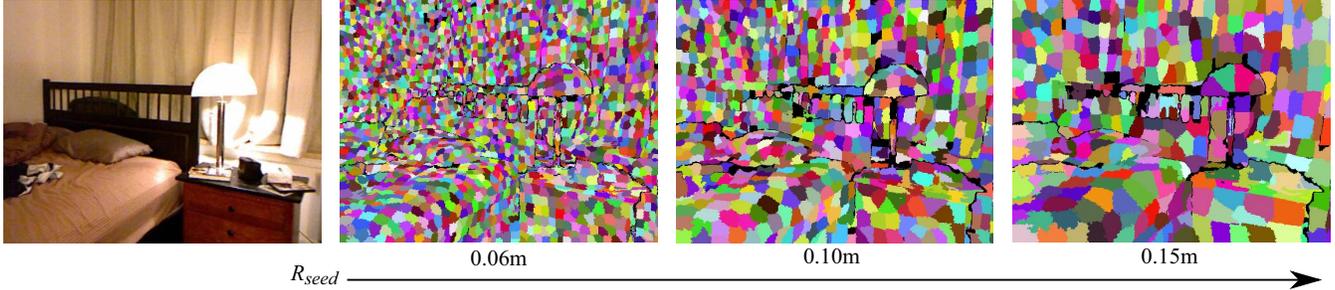R_{seed} ————————————————————→
0.06m 0.10m 0.15m

Figure 2. Image segmented using VCCS with seed resolutions of 0.1, 0.15 and 0.2 meters.

ends at the neighboring cluster centers. This means that we can normalize our spatial distance D_s using the maximally distant point considered for clustering, which will lie at a distance of $\sqrt{3}R_{seed}$. Color distance D_c, is the euclidean distance in CIELab space, normalized by a constant m. Distance in FPFH space, D_f, is calculated using the Histogram Intersection Kernel [2]. This leads us to a equation for normalized distance D:

$$D = \sqrt{\frac{\lambda D_c^2}{m^2} + \frac{\mu D_s^2}{3R_{seed}^2} + \epsilon D_{HiK}^2}, \qquad (4)$$

where λ, μ, and ϵ control the influence of color, spatial distance, and geometric similarity, respectively, in the clustering. In practice we keep the spatial distance constant relative to the other two so that supervoxels occupy a relatively spherical space, but this is not strictly necessary. For the experiments in this paper we have color weighted equally with geometric similarity.

3.4. Flow Constrained Clustering

Assigning voxels to supervoxels is done iteratively, using a local k-means clustering related to [1, 14], with the significant difference that we consider connectivity and flow when assigning pixels to a cluster. The general process is as follows: beginning at the voxel nearest the cluster center, we flow outward to adjacent voxels and compute the distance from each of these to the supervoxel center using Equation 4. If the distance is the smallest this voxel has seen, its label is set, and using the adjacency graph, we add its neighbors which are further from the center to our search queue for this label. We then proceed to the next supervoxel, so that each level outwards from the center is considered at the same time for all supervoxels. We proceed iteratively outwards until we have reached the edge of the search volume for each supervoxel (or have no more neighbors to check).

This amounts to a breadth-first search of the adjacency graph, where we check the same level for all supervoxels before we proceed down the graphs in depth. Importantly, we avoid edges to adjacent voxels which we have already checked this iteration. The search concludes for a super-

voxel when we have reached all the leaf nodes of its adjacency graph or none of the nodes searched in the current level were set to its label. This search procedure, illustrated in Figure 3, has two important advantages over existing methods:

1. Supervoxel labels cannot cross over object boundaries that are not actually touching in 3D space, since we only consider adjacent voxels, and

2. Supervoxel labels will tend to be continuous in 3D space, since labels flow outward from the center of each supervoxel, expanding in space at the same rate.

Once the search of all supervoxel adjacency graphs has concluded, we update the centers of each supervoxel cluster by taking the mean of all its constituents. This is done iteratively; either until the cluster centers stabilize, or for a fixed number of iterations. For this work we found that the supervoxels were stable within a few iterations, and so have simply used five iterations for all presented results.

4. Three Dimensional Voxel Segments

The proposed method works directly on voxelized point clouds, which has advantages over existing methods which operate in the projected image plane. The most important of these is the ability to segment clouds coming from many sensor observations - either using multiple cameras [3] or accumulated clouds from one [6]. Computationally, this is advantageous, as the speed of our method is dependent on the number of occupied voxels in the scene[2], and not the number of observed pixels. As observations will have significant overlap, this means that it is cheaper to segment the overall voxel cloud than the individual 2D observations. For instance, the scene in Figure 5 comes from 180 Kinect observations (640x480), and yet the final voxel cloud (with $R_{voxel} = 0.01m$) only contains 450k voxels.

Additionally, while VCCS will become more accurate as cloud information is filled in by additional observations, 2D methods must necessarily segment them independently and therefore cannot make use of the added information. Most

---
[2] We should note that while the initial voxelization of the cloud does take more time with a larger cloud, it remains insignificant overall
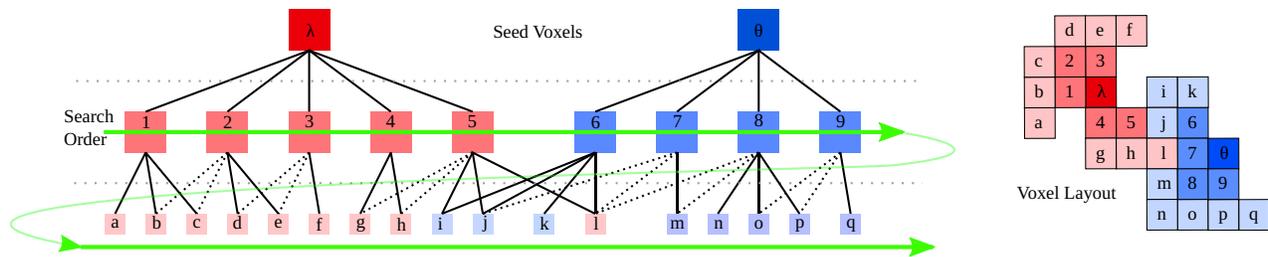
Figure 3. Search order for the flow constrained clustering algorithm (shown in 2D for clarity). Dotted edges in the adjacency graph are not searched, as the nodes have already been added to the search queue.

importantly, even with methods that use depth information, such as that of Weikersdorfer *et al*. [14], it is not clear how one would combine the multiple segmented 2d images, as superpixels from sequential observations will have no relation to each other and will have conflicting partitionings of space in the merged cloud.

5. Experimental Evaluation

In order to evaluate the quality of supervoxels generated by VCCS, we performed a quantitative comparison with three state-of-the-art superpixel methods using publicly available source code. We selected the two 2D techniques with the highest published performance from a recent review [1]: a graph based method, GCb10 [13][3], and a gradient ascent local clustering method, SLIC [1][4]. Additionally, we selected another method which uses depth images, DASP[14][5]. Examples of over-segmentations produced by the methods are given in Figure 6.

5.1. Dataset

For testing, we used the recently created NYU Depth Dataset V2 semantic segmentation dataset of Silberman *et al*. [12][6]. This contains 1449 pairs of aligned RGB and depth images, with human annotated densely labeled ground truth. The images were captured in diverse cluttered indoor scenes, and present many difficulties for segmentation algorithms such as varied illumination and many small similarly colored objects. Examples of typical scenes are shown in Figure 6.

5.2. Returning to the Projected Plane

RGB+D sensors produce what is known as an organized point cloud- a cloud where every point corresponds to a pixel in the original RGB and depth images. When such a

Figure 4. Example of hole-filling for images after returning from voxel-cloud to the projected image plane. Depth data, shown in the top left, has holes in it, shown as dark blue areas (here, due to the lamp interfering with the Kinect). The resulting supervoxels do not cover these holes as shown in the bottom left, since the cloud has no points in them. To generate a complete 2D segmentation, we fill these holes in using the SLIC algorithm, resulting in a complete segmentation, seen in the top right. The bottom right shows human annotated ground truth for the scene.

cloud is voxelized, it necessarily loses this correspondence, and becomes an unstructured cloud which no longer has any direct relationship back to the 2D projected plane. As such, in order to compare results with existing 2D methods we were forced to devise a scheme to apply supervoxel labels to the original image.

To do this, we take every point in the original organized cloud and search for the nearest voxel in the voxelized representation. Unfortunately, since there are blank areas in the original depth image due to such factors as reflective surfaces, noise, and limited sensor range, this leaves us with some blank areas in the output labeled images. To overcome this, we fill in any large unlabeled areas using the SLIC algorithm. This is not a significant drawback, as the purpose of the algorithm is to form supervoxels in 3D space, not superpixels in the projected plane, and this hole-filling is only needed for comparison purposes. Additionally, the hole fill-
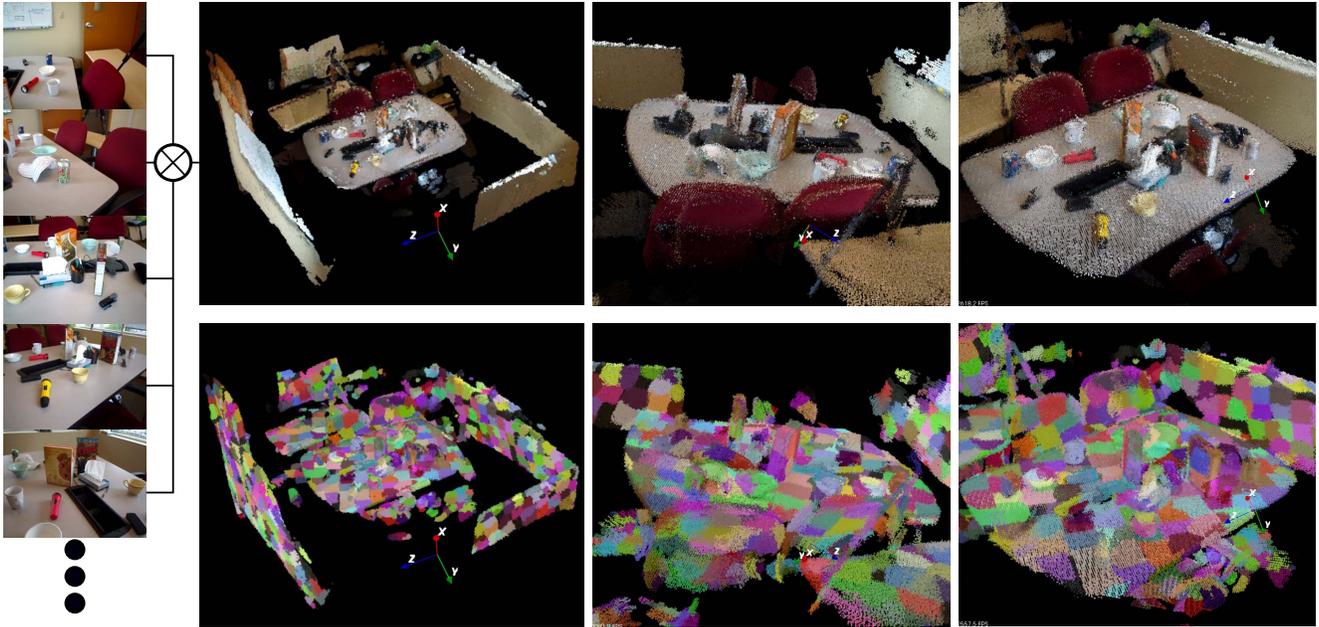
Figure 5. Over-segmentation of a cloud from the RGB-D scenes dataset[6]. The cloud is created by aligning 180 kinect frames, examples of which are seen on the left side. The resulting cloud has over 3 million points, which reduces to 450k points at $R_{voxel} = 0.01m$ and 100k points with $R_{voxel} = 0.02m$. Over-segmentation of these take 6 and 1.5 seconds, respectively (including voxelization).
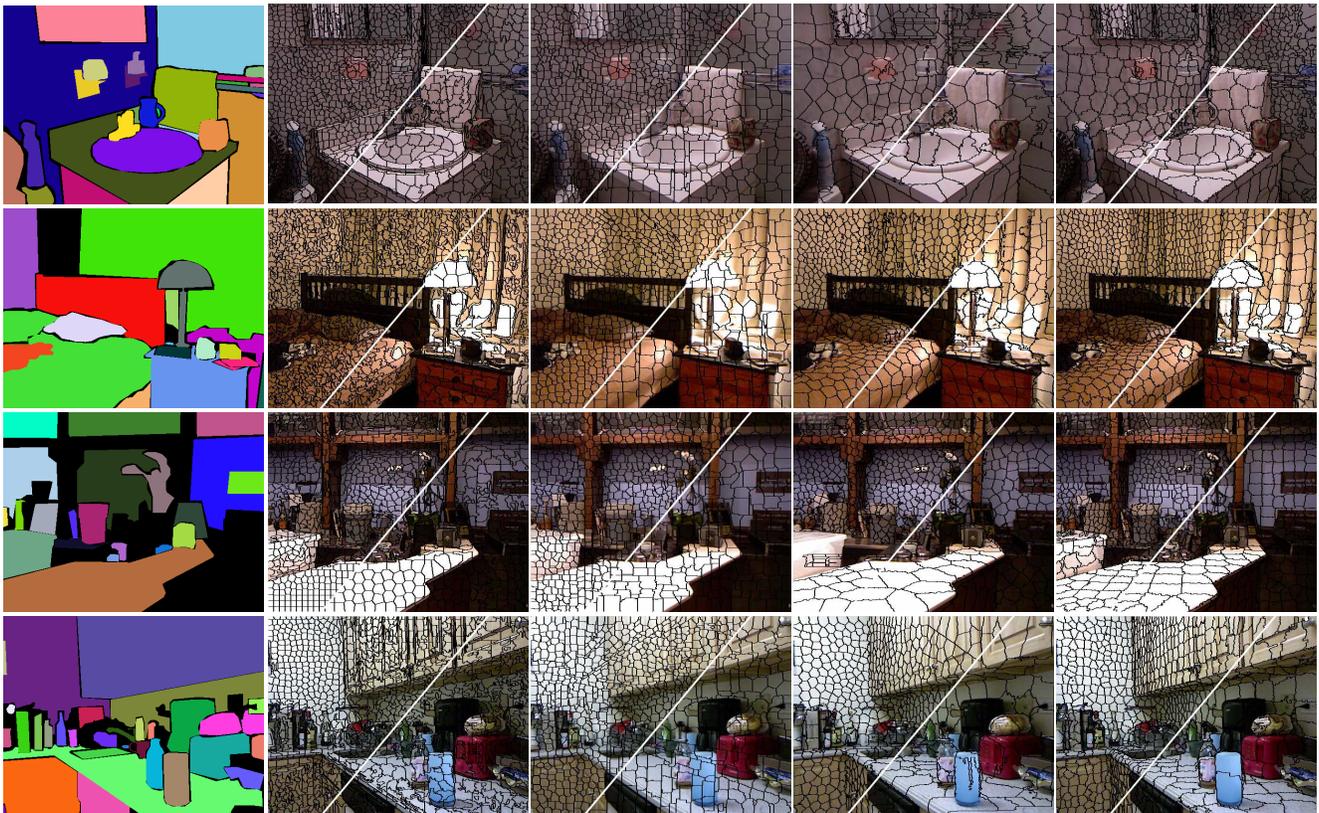

Figure 6. Examples of under-segmentation output. From left to right- ground truth annotation, SLIC, GCb10, DASP, and VCCS. Each is shown with two different superpixel densities.
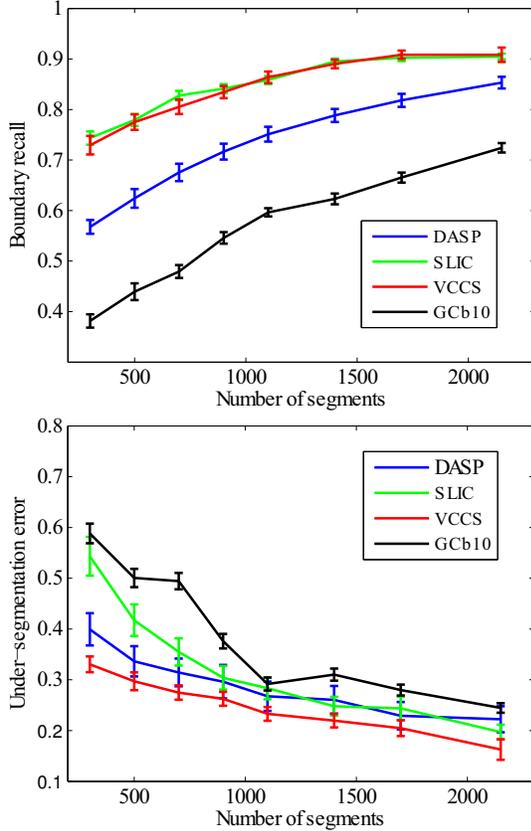
Figure 7. Boundary recall and under-segmentation error for SLIC, GCb10, DASP, and VCCS.

ing actually makes our results worse, since it does not consider depth, and therefore tends to bleed over some object boundaries that were correctly maintained in the supervoxel representation. An example of what the resulting segments look like before and after this procedure are shown in Figure 4.

5.3. Evaluation Metrics

The most important property for superpixels is the ability to adhere to, and not cross, object boundaries. To measure this quantitatively, we have used two standard metrics for boundary adherence- boundary recall and under-segmentation error[7, 13]. Boundary recall measures what fraction of the ground truth edges fall within at least two pixels of a superpixel boundary. High boundary recall indicates that the superpixels properly follow the edges of objects in the ground truth labeling. The results for boundary recall are given in Figure 7. As can be seen, VCCS and SLIC have the best boundary recall performance, giving similar results as the number of superpixels in the segmentation varies.

Under-segmentation error measures the amount of leak-

age across object boundaries. For a ground truth segmentation with regions $g_1, ..., g_M$, and the set of superpixels from an over-segmentation, $s_1, ...s_K$, under-segmentation error is defined as

$$E_{useg} = \frac{1}{N} \left[\sum_{i=1}^{M} \left(\sum_{s_j | s_j \cap g_i} |s_j| \right) - N \right], \qquad (5)$$

where $s_j \mid s_j \cap g_i$ is the set of superpixels required to cover a ground truth label g_i, and N is the number of labeled ground truth pixels. A lower value means that less superpixels violated ground truth borders by crossing over them. Figure 7 compares the four algorithms, giving under-segmentation error for increasing superpixel counts. VCCS outperforms existing methods for all superpixel densities.

5.4. Time Performance

As superpixels are used as a preprocessing step to reduce the complexity of segmentation, they should be computationally efficient so that they do not negatively impact overall performance. To quantify segmentation speed, we measured the time required for the methods on images of increasing size (for the 2D methods) and increasing number of voxels (for VCCS). All measurements were recorded on an Intel Core i7 3.2Ghz processor, and are shown in Figure 8. VCCS shows performance competitive with SLIC and DASP (the two fastest superpixel methods in the literature) for voxel clouds of sizes which are typical for Kinect data at $R_{voxel} = 0.008m$ (20-40k voxels). It should be noted that only VCCS takes advantage of multi-threading (for octree, kd-tree, and FPFH computation), as there are no publicly available multi-threaded implementations of the other algorithms.

6. Discussion and Conclusions

We have presented VCCS, a novel over-segmentation algorithm for point-clouds. In contrast to existing approaches, it works on a voxelized cloud, using spatial connectivity and geometric features to help superpixels conform better to object boundaries. Results demonstrated that VCCS produces over-segmentations which perform significantly better than the state-of-the-art in terms of under-segmentation error, and equal to the top performing method in boundary recall. This is fortunate, as we consider under-segmentation error to be the more important of the two measures, as boundary recall does not penalize for crossing ground truth boundaries- meaning that even with a high boundary recall score, superpixels might perform poorly in actual segmentation. We have also presented timing results which show that VCCS has run time comparable to the fastest existing methods, and is fast enough for use as a pre-processing step in online semantic segmentation applications such as robotics.
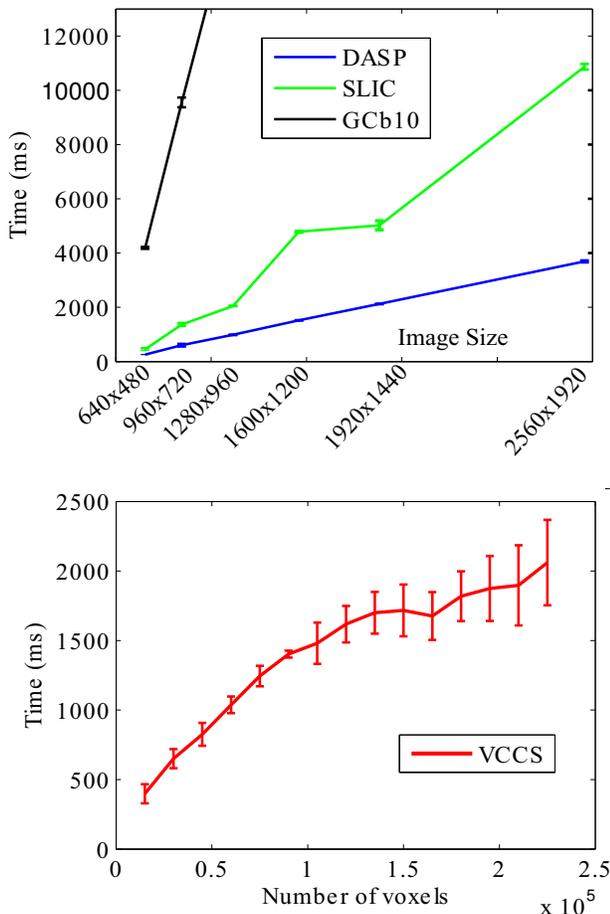
Figure 8. Speed of segmentation for increasing image size and number of voxels. Use of GCb10 rapidly becomes unfeasible for larger image sizes, and so we do not adjust the axes to show its run-time. The variation seen in VCCS run-time is due to dependence on other factors, such as R_{seed} and overall amount of connectivity in the adjacency graphs.

We have made the code publicly available as part of the popular Point Cloud Library, and intend for VCCS to become an important step in future graph-based 3D semantic segmentation methods.

Acknowledgements

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Trans. Pattern Anal. Machine Intell.*, 34(11):2274 –2282, nov. 2012. 2, 3, 4, 5

[2] A. Barla, F. Odone, and A. Verri. Histogram intersection kernel for image classification. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, pages III – 513–16 vol.2, sept. 2003. 4

[3] D. A. Butler, S. Izadi, O. Hilliges, D. Molyneaux, S. Hodges, and D. Kim. Shake'n'sense: reducing interference for overlapping structured light depth cameras. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, CHI '12, pages 1933–1936, New York, USA, 2012. 4

[4] B. Fulkerson, A. Vedaldi, and S. Soatto. Class segmentation and object localization with superpixel neighborhoods. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 670 –677, oct. 2009. 1

[5] A. Hanbury. How do superpixels affect image segmentation? In *Progress in Pattern Recognition, Image Analysis and Applications*, Lecture Notes in Computer Science, pages 178–186. 2008. 1

[6] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1817 –1824, may 2011. 4, 6

[7] A. Levinshtein, A. Stere, K. Kutulakos, D. Fleet, S. Dickinson, and K. Siddiqi. Turbopixels: Fast superpixels using geometric flows. *IEEE Trans. Pattern Anal. Machine Intell.*, 31(12):2290 –2297, dec. 2009. 2, 7

[8] A. Moore, S. Prince, J. Warrell, U. Mohammed, and G. Jones. Superpixel lattices. In *Computer Vision and Pattern Recognition, 2008 (CVPR). IEEE Conference on*, pages 1 –8, june 2008. 2

[9] X. Ren and J. Malik. Learning a classification model for segmentation. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 10 –17 vol.1, oct. 2003. 1

[10] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Robotics and Automation, 2009 (ICRA). IEEE International Conference on*, pages 3212 –3217, may 2009. 3

[11] R. B. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). In *Robotics and Automation, 2011 (ICRA). IEEE International Conference on*, Shanghai, China, may 2011. 2

[12] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *Computer Vision ECCV 2012*, volume 7576 of *Lecture Notes in Computer Science*, pages 746–760. 2012. 5

[13] O. Veksler, Y. Boykov, and P. Mehrani. Superpixels and supervoxels in an energy optimization framework. In *Computer Vision ECCV 2010*, volume 6315, pages 211–224. 2010. 2, 5, 7

[14] D. Weikersdorfer, D. Gossow, and M. Beetz. Depth-adaptive superpixels. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 2087–2090, 2012. 2, 4, 5

[15] Y. Yang, S. Hallman, D. Ramanan, and C. Fowlkes. Layered object detection for multi-class segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3113 –3120, june 2010. 1