The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XL-5, 2014
ISPRS Technical Commission V Symposium, 23 – 25 June 2014, Riva del Garda, Italy

A NEW FRAMEWORK FOR INTERACTIVE SEGMENTATION OF POINT CLOUDS

Kun Liu* Jan Boehm

Dept of Civil, Environ & Geomatic Eng, University College London
{kun.liu, j.boehm}@ucl.ac.uk

KEY WORDS: Point Cloud, Segmentation, User, Visualization, Algorithms

ABSTRACT:

Point cloud segmentation is a fundamental problem in point processing. Segmenting a point cloud fully automatically is very challenging due to the property of point cloud as well as different requirements of distinct users. In this paper, an interactive segmentation method for point clouds is proposed. Only two strokes need to be drawn intuitively to indicate the target object and the background respectively. The draw strokes are sparse and don't necessarily cover the whole object. Given the strokes, a weighted graph is built and the segmentation is formulated as a minimization problem. The problem is solved efficiently by using the Max Flow Min Cut algorithm. In the experiments, the mobile mapping data of a city area is utilized. The resulting segmentations demonstrate the efficiency of the method that can be potentially applied for general point clouds.

1. INTRODUCTION

Due to the rapid development of surveying and reality capture technology, such as mobile mapping vehicles, e.g., Google street view car and consumer range cameras, e.g., Microsoft's Kinect, the acquisition of point clouds continues to become easier and faster and incurs lower costs. Point cloud data is ubiquitous nowadays in many areas such as geomatics, civil engineering and computer vision. However before point cloud can be used in any of these application areas, the data needs to be processed in a series of operations. Among of these operations, segmentation of point clouds is an important one. For instance, object localization and classification [Kim et al., 2012] need segmentation to partition point clouds into different clusters.

Segmentation of point clouds is a fundamental problem and a large number of approaches were proposed in last two decades. However, it remains to be one of the most challenging problems in point cloud processing. Most segmentation methods for point clouds are inspired from image segmentation algorithms [Golovinskiy and Funkhouser, 2009,Ioannou et al., 2012]. Image segmentation and point cloud segmentation, both need to compute neighbors of a pixel or a point. Due to the grid structure of images, the neighbors of a pixel can be obtained effortlessly. However, for point clouds, a neighbor searching algorithm is necessary, e.g., k-nearest neighbors (KNN) [Muja and Lowe, 2009]. Segmentation aims to find boundaries between distinct parts. To determine the boundaries, it is essential to choose a proper descriptor to characterize pixels or points. Usually pixels are described by color models such as RGB, while points are defined by coordinates. The color is a descriptor which can easily identify differences of pixels, whereas coordinates are not intrinsic and can vary in different frames. Therefore, other descriptors for points are usually utilized, e.g., normals and curvatures. Moreover, an image is a flat 2D grid, whereas a point cloud can exhibit complicated geometry in 3D space. Due to different acquisition systems used, point clouds might have distinct qualities possibly containing noise and outliers. On account of these issues, segmentation of point clouds is more difficult than segmentation of images.

Segmentation of point clouds is difficult but user interactions could help to address this challenging problem. In this paper, we present a new framework which only needs a few simple user interactions to segment point clouds efficiently. The user interactions are inspired by Lazy Snapping [Li et al., 2004] for image segmentation. As shown in Figure 6, the user interactions are simple and sparse, and only two strokes are required to accomplish the segmentation. The method can be employed in many other applications such as labeling point clouds. In our proposed method, the technique of graph cuts is used as [Li et al., 2004, Liu et al., 2009] for image segmentation. As shown in Figure 1, graph cuts aims to find the minimum cut partitioning the graph into disjoint parts, and hence graph cuts is suitable for segmentation problems. First, each point in the point cloud is connected with nearby points using nearest neighbor algorithms to construct a graph. With user interactions, the points selected by the drawn stroke are connected to the virtual added nodes. If the weights are determined properly, the minimum cut coincides with the boundary of the target object. The weights are usually defined by using various descriptors of points such as coordinate and normal. For some point clouds with colors, the color information could be used as well. Therefore, the proposed method is a framework which is flexible and can be customized. In summary, this paper makes the following two contributions:

1. Propose a framework to segment point clouds interactively
2. Develop an efficient solution for partitioning

In the experiments, the mobile mapping laser scanning data are utilized to test efficiency and robustness of the method.

2. RELATED WORK

Segmentation is a well studied problem in image processing, and many approaches were proposed in the last decade that also effect the related work in point cloud area. Most methods for point cloud segmentation are inspired by methods for image segmentation.

In image segmentation, a group of approaches using interactive tools is very attractive. Based on user interactions these methods can output results efficiently. Among of them, Lazy Snapping [Li et al., 2004] is a representative work. In Lazy Snapping, lines are

*Corresponding author.

drawn to indicate the target object and the background. The lines are far away from the true boundary of the object, and are not necessary to cover the whole object. With these simple interactions, a cut along the contour of the object is automatically computed and hence the object is easily segmented out from the image. Our method proposed in this paper is inspired by Lazy Snapping and uses a similar user interaction. In addition, GrabCut [Rother et al., 2004] and Paint Selection [Liu et al., 2009] are two more efficient methods for image segmentation. In GrabCut, users drag a rectangle loosely surrounding the target object and then a segmentation is computed. The Paint Selection method provides a progressive paint tool. Users paint the target object progressively using a brush and check the results instantly until the satisfied segmentation is obtained. Noticeably, Lazy Snapping, GrabCut and Paint Selection all use the technique of graph cuts which is related to the Max-Flow Min-Cut problem in graph theory. Graph cuts is an optimization process which computes the minimum cut for a weighted graph. It is very suitable for solving segmentation problems. The previous methods for image segmentation also demonstrate its efficiency. In our proposed method, graph cuts is opted for point cloud segmentation as well.

In point cloud segmentation, graph cuts is applied in the previous work [Golovinskiy and Funkhouser, 2009]. In the method presented in [Golovinskiy and Funkhouser, 2009], the location of the target object needs to be specified. Based on the given location, a weighted graph is built and then graph cuts is applied to compute the boundary of the object. In the method, constraints indicating the object or the background can be added interactively in necessary to obtain a satisfied segmentation. In [Ioannou et al., 2012] the Difference of Normals (DoN) operator is introduced for segmentation of point clouds. The operator is inspired by the Difference of Gaussian (DoG) which is a filter for edge detection in image processing. Since the normal computation of point cloud is based on the local neighborhood of points, the Difference of Normals characterizes the distinctions of normals computed using different sizes of the local neighborhood. For example, this distinctions is significant on the areas with the sharp edge. The boundaries between different objects are assumed to be along sharp edges in this method. By detecting the sharp edges, a segmentation of the point cloud is computed. However, the two discussed methods for point cloud segmentation both have limitations. The former method needs to specify locations and add constraints that are tedious and not easy to manipulate if a point cloud is geometrically complicated. The assumption in the latter one is so strong that prevents widely applying for general point clouds.

3. THE INTERACTIVE SEGMENTATION METHOD

In this section, an interactive segmentation method based on graph cuts is proposed to partition point clouds. The related theory is reviewed first. Then the segmentation algorithm is presented in detail.

3.1 Graph cuts

In computer vision and computer graphics, graph cuts is widely applied in various applications such as image segmentation [Li et al., 2004] and surface reconstruction [Lafarge and Alliez, 2013]. In these applications, the problems are formulated as energy minimization which can be solved by using graph cuts.

As shown in Figure 1, a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is displayed with a set of nodes $\mathcal{V}$ and a set of edge $\mathcal{E}$. For instance, in image segmentation pixels are considered as the nodes and edges are built by connecting each pixel and its neighbors. In the graph, two special nodes are called source and sink which are colored as the green and the yellow dots respectively in the figure. In addition, a weight w_e is associated with each edge e in $\mathcal{E}$. Graph cuts aims to compute the minimum cut $\mathcal{C}$ which partitions the graph into two disjoint components as illustrated in Figure 1. Mathematically, the minimum cut is the solution of the minimization problem

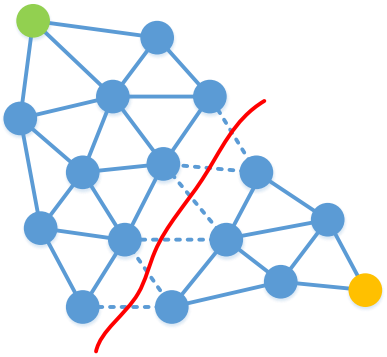


Figure 1: The colored nodes and edges constitute a graph and each edge is associated with a weight. The green and the yellow nodes are two special ones, named as source and sink respectively. The red cut separates the graph into two disjoint subsets and each of them contains only source or sink node. The removed edges are dashed in the figure. The minimum cut is a cut whose removed edges have the smallest sum of weights.

$$\underset{\mathcal{C}}{\text{minimize}} \sum_{e \in \mathcal{C}} w_e. \tag{1}$$

This problem coincides with the Min-Cut problem in graph theory [Bondy and Murty, 1976]. In practise, the Min-Cut problem is converted to its dual equivalent problem, i.e., Max-Flow, and solved efficiently [Boykov and Kolmogorov, 2004]. See the book [Bondy and Murty, 1976] for further detail about the Max-Flow Min-Cut theorem.

3.2 Algorithm

In this paper, an interactive approach is proposed to address the segmentation problem for point cloud. The proposed user interactions are similar to Lazy Snapping [Li et al., 2004] for image segmentation. It allows users to draw sparse strokes and segment target objects out from point clouds as illustrated in Figure 6a. Generally, only two strokes are required to specify target the object and the background respectively. Although the strokes are sparse, they are sufficient for graph cuts to compute a satisfied segmentation as show in Figure 6b.

3.2.1 Segmentation via graph cuts To use graph cuts to segment a point cloud, a graph needs to be built first. Figure 2 gives an example illustrating a built graph of a point cloud. Given a point cloud, each point is considered as a node in the graph. To construct edges, more computation is necessary. This is due to the fact that point cloud is a type of geometrical representation lacking connectivities between points. It is distinct from images which have grid structures. In our method, to construct edges the k-nearest neighbors algorithm (KNN) [Muja and Lowe, 2009] is applied to compute neighbors of all points, and edges are created by connecting each point and its neighbors.

As stated in Section 3.1, two special nodes, i.e., source and sink, need to be defined. In our method, two virtual nodes are introduced as the source and the sink (see the blue and the orange dots in Figure 2). More edges are built between the sink (orange)

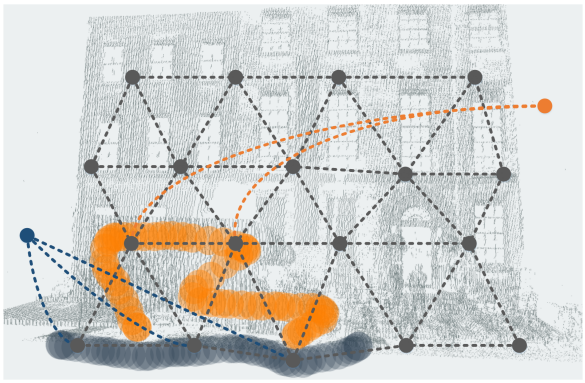


Figure 2: The orange and gray strokes are drawn by users to specify the object (a van) and the background. Given the two strokes, a graph of the point cloud is built, the dots represent the nodes in the graph and the dashed lines are the edges. Noticeably, the blue and the orange dots are two added nodes as source and sink (see Figure 1), and the edges connecting them and common nodes (black) are also colored as orange and blue lines respectively. Note that the graph displayed in the figure is just an illustration, because in reality every point is one node in the built graph.

and the nodes specified as the object part. Similarly, the source (blue) and nodes specified as the background part are connected as illustrated in Figure 2.

The topology of the graph is determined and edge's weight needs to be defined as well. Given a weighted graph, graph cuts is able to compute the minimum cut as shown in Figure 1. Intuitively, the minimum cut is generally along edges with smaller weights. Therefore, to define a proper weight between two nodes, a smaller value is set if the two nodes more likely belong different objects in the point cloud, and vice versa.

In our experiments, two kinds of weight definitions are used. The first one is using the Euclidean distance between two nodes. Specifically, for an edge $e = (a, b)$ where a and b represent two different nodes, the weight w_e is defined as

$$w_e = e^{-\frac{||x_a - x_b||^2}{\sigma^2}}. \quad (2)$$

x_a and x_b are the coordinate vectors of a and b respectively, $||\cdot||$ is the l^2-norm and σ is a user specified parameter. The function $f(x) = e^{-\frac{x^2}{\sigma^2}}$ used in Equation 2 is a Gaussian function which is widely applied in many applications such as data denoising and regression. Figure 3 depicts the shapes of the function with different σ values. As shown in the figure, the $f(x)$ value decreases slower when x is larger than σ. Therefore, σ can be considered as a parameter to scale distances between nodes in a nonlinear way. The other way to define edge weights is based on *normals*. In a point cloud, each point can be associated with a covariance matrix determined locally [Demantké et al., 2011]. In our paper, the normal of a point is defined as the least significant eigenvector of the associated covariance matrix, i.e., the eigenvector corresponding the smallest eigenvalue. To compute the normal filed of a point cloud, the approach presented in [Demantké et al., 2011] is used. The weight w_e of the edge $e = (a, b)$ is defined as

$$w_e = |n_a^T n_b|. \quad (3)$$

n_a and n_b are the normals of a and b respectively and $|\cdot|$ is the absolute value function. The normals are 3-dimension column unit vector and hence $n_a^T n_b$ is the dot product of vectors. This weight characterizes normal similarity of two nodes. For example, the

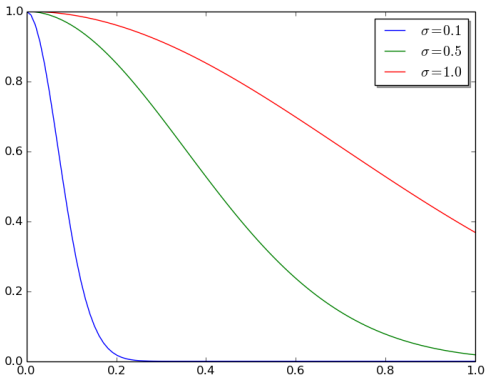
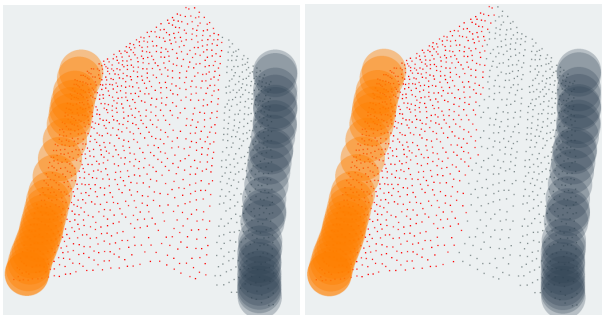


Figure 3: The shapes of the Gaussian function with different σ values are displayed.

value is close to the maximum value 1 when the two normals are nearly parallel. Furthermore, for the edges connecting virtual nodes and common nodes, weights are always defined as infinity.



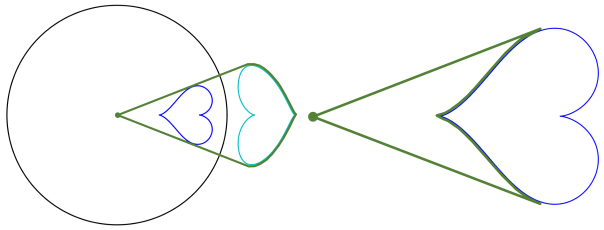
(a) (b)

Figure 4: An example shows the difference of the two definitions of edge weights: (a) weights defined using Euclidean distance (see Equation 2); (b) weights based on normals (see Equation 3). The point cloud represents a L-shape model with a long gap in one face. The user interactions are exactly same.

As shown in Figure 4, the two definitions of edge weights exhibit different properties. A point cloud of L-shape are segmented twice by our method with same user interactions but using different weight definitions. In Figure 4a the definition in Equation 2 is used. Since the weights are determined by using the Euclidean distance, the segmented boundary is along the long gap where distances between points and their neighbors are larger than other areas. On the other hand, the computed boundary in Figure 4b is along the sharp edge of the L-shape due to using the definition in Equation 3 based on normals. In our method, users can switch the definitions depending on different segmentation tasks. Specifically, to segment an object locating distantly from others the definition in Equation 2 is preferred, while the definition in Equation 3 is more suitable to partition objects with boundaries along sharp edges. Noticeably, the proposed method is so flexible that users can adopt other definitions of edge weights such as combining color information. From this point of view, this work also proposes a framework for point cloud segmentation that can be highly customized.

3.2.2 Solving graph cuts In Section 3.2.1, the segmentation of a point cloud is formulated as a minimization problem same to Equation 1. The problem is well studied in graph theory and can be addressed efficiently using the Max-Flow Min-Cut algo-

rithms [Bondy and Murty, 1976]. In our experiments, the Max Flow algorithm proposed in [Boykov and Kolmogorov, 2004] is opted for solving, and the public available C++ library [Kolmogorov, 2010] is used in our implementation.



(a) (b)

Figure 5: A 2D example is given to illustrate the algorithm in [Katz et al., 2007] for computing visibility of point cloud. The blue heart represents a point cloud and the view point is in the center of the circle. The blue heart is transformed to the cyan distorted one as Equation 4. A convex hull of the transformed heart and the view point is computed, and the visible part of the blue heart exactly corresponds the hull boundary as shown in (b). In reality the circle is much larger than the displayed one.

3.2.3 Visibility of point cloud In our method, a point cloud is segmented by drawing strokes such as in Figure 6a. However, the orange stroke in the figure not only covers a few points from the van but also some points from the building. This issue can be solved by computing the visibility of the point cloud. By using visibility information, the part of the building occluded by the van can be excluded before drawing strokes.

Point cloud is a type of discrete representation and geometry information is missing between points. Therefore, computing visibility for a point cloud is not straightforward. In our implementation, the algorithm in [Katz et al., 2007] is used to obtain the visibility. The idea of the algorithm is illustrated in Figure 5. Mathematically, the given point cloud is transformed as the formulation

$$x = x + 2 \cdot \frac{R - ||x||}{||x||} \cdot x, \tag{4}$$

where x is the coordinate vector, R is the circle radius and $||x||$ is the l^2-norm of x. Then the convex hull of the transformed point cloud and the view point is computed. The visible points are obtained by reversely transforming the boundary of the hull as shown in Figure 5b.

4. EXPERIMENTAL RESULTS

In this section, segmentation results are shown to demonstrate the efficiency of the method proposed in Section 3.. Point clouds used in the experiments are from the mobile mapping data of the Bloomsbury area (London). The point data is acquired by the vehicle with mounted laser scanners.

Figure 6 displays a complicated scene with van, building, bicycles and etc. The van is the target object and aims to be segmented out from the point cloud. However, several challenges exist in this example. Parts of building are missing. The distribution of points is not uniform and exhibits strip-like patterns which could potentially harm the resulting segmentation. Furthermore the van is closely connected to the ground. All these issues increase the difficulty to get a satisfied results. As shown in Figure 6a, the orange stroke is drawn to specify the target object and the gray one is used to indicate the background. The gray one is on the ground region which is close to the van, and hence a correct boundary of the van is guaranteed. Noticeable, these strokes are sparse and not along the boundary of the object. The resulting segmentation is computed by using graph cuts as discussed in Section 3.2.1. As shown in Figure 6b, the result is correct without selecting points on the building due to the visibility check as discussed in Section 3.2.3.

In Figure 7, three segmentation results are shown using the proposed interactive method. The left column illustrates the interactions where the oranges strokes specify the target objects and the gray strokes indicate the backgrounds. The right column shows the results of segmenting person, tree and bench respectively. In the tree's example, it shows clearly that the drawn strokes are very few and sparse comparing the large region of the target object. In summary, the interactions is easy for users, only intuitive drawing is required. The resulting segmentations of the examples demonstrate the efficiency of the method that can be applied for general point clouds.

(a)

(b)

Figure 6: The two strokes shown in (a) are drawn by users. The orange one specifies the target object, i.e., a van in the point cloud, and the gray one specifies the background. The strokes are sparse and don't necessarily cover a large part. Using graph cuts, the selection of the orange stroke can propagate over the whole object and hence the van is segmented out from the point cloud as displayed in (b).

5. CONCLUSION

In this paper, an interactive method based on graph cuts is proposed for point cloud segmentation. The interactions are simple and only a few strokes are drawn. The experiments are conducted and prove the efficiency of the method which can be potentially applied for general point clouds.

Two immediate directions can be explored to improve the work in future. One direction is optimizing the implementation and accelerating the algorithm to segment a larger point cloud, e.g., with more than ten million points. Hierarchical strategy of solving and subsampling can be used. The subsampling is also able to improve the quality of point distribution. The other direction is studying descriptors besides coordinate and normal that can benefit other applications as well such as classification and recognization.

ACKNOWLEDGEMENTS

The authors would like to thank ABA Surveying LTD to provide the mobile mapping data. They also would like to thank Simon Julier for the helpful discussions and suggestions. Part of this research work is supported by EU grant FP7-ICT-2011-318787 (IQmulus).

REFERENCES

Bondy, J. A. and Murty, U. S. R., 1976. Graph theory with applications. Vol. 6, Macmillan London.

Boykov, Y. and Kolmogorov, V., 2004. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. Pattern Analysis and Machine Intelligence, IEEE Transactions on 26(9), pp. 1124–1137.

Demantké, J., Mallet, C., David, N. and Vallet, B., 2011. Dimensionality based scale selection in 3D lidar point clouds. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, Laser Scanning.

Golovinskiy, A. and Funkhouser, T., 2009. Min-cut based segmentation of point clouds. In: Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on, IEEE, pp. 39–46.

Ioannou, Y., Taati, B., Harrap, R. and Greenspan, M., 2012. Difference of normals as a multi-scale operator in unorganized point clouds. In: 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on, IEEE, pp. 501–508.

Katz, S., Tal, A. and Basri, R., 2007. Direct visibility of point sets. In: ACM Transactions on Graphics (TOG), Vol. 26number 3, ACM, p. 24.

Kim, Y. M., Mitra, N. J., Yan, D.-M. and Guibas, L., 2012. Acquiring 3D indoor environments with variability and repetition. ACM Transactions on Graphics (TOG) 31(6), pp. 138.

Kolmogorov, V., 2010. maxflow-v3.01, `http://vision.csd.uwo.ca/code/maxflow-v3.01.zip`.

Lafarge, F. and Alliez, P., 2013. Surface reconstruction through point set structuring. Computer Graphics Forum 32(2pt2), pp. 225–234.

Li, Y., Sun, J., Tang, C.-K. and Shum, H.-Y., 2004. Lazy snapping. ACM Trans. Graph. 23(3), pp. 303–308.

Liu, J., Sun, J. and Shum, H.-Y., 2009. Paint selection. In: ACM Transactions on Graphics (ToG), Vol. 28number 3, ACM, p. 69.

Muja, M. and Lowe, D. G., 2009. Fast approximate nearest neighbors with automatic algorithm configuration. In: VISAPP (1), pp. 331–340.

Rother, C., Kolmogorov, V. and Blake, A., 2004. Grabcut: Interactive foreground extraction using iterated graph cuts. In: ACM Transactions on Graphics (TOG), Vol. 23number 3, ACM, pp. 309–314.

Figure 7: An illustration of segmenting person, tree and bench out from point clouds respectively: (a), (c), (e) in the left column show the user interactions and (b), (d), (f) in the right column display the corresponding results.