Project IP-2014-09-3155

Advanced 3D Perception for Mobile Robot Manipulators

3D MESH SEGMENTATION INTO PLANAR SEGMENTS BASED ON REGION GROWING AND PLANE INTERSECTION

Technical Report

ARP3D.TR3

version 2.1

Robert Cupec

Josip Juraj Strossmayer University of Osijek

Faculty of Electrical Engineering Osijek

Osijek, 2016.

Ovaj rad je financirala/sufinancirala Hrvatska zaklada za znanost projektom IP-2014-09-3155.

Mišljenja, nalazi i zaključci ili preporuke navedene u ovom materijalu odnose se na autora i ne odražavaju nužno stajališta Hrvatske zaklade za znanost.

This work has been fully supported by/supported in part by the Croatian Science Foundation under the project number IP-2014-09-3155.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of Croatian Science Foundation.

1. Introduction

Many segmentation algorithms perform oversegmentation to superpixels or supervoxels as a preprocessing step (Gupta, Arbeláez and Malik, 2013). The introduction of a low-level preprocessing step to oversegment images into superpixels – relatively small regions whose boundaries agree with those of the semantic entities in the scene – has enabled advances in segmentation by reducing the number of elements to be labeled from hundreds of thousands, or millions, to a just few hundred (Simari, Picciau and De Floriani, 2014). Probably the most popular method for oversegmentation of 2D images into superpixels is SLIC (Achanta et al., 2012). The method is based on k-means clustering. The same idea is adapted to 3D point clouds by (Papon et al., 2013). They designed a method which represents a 3D point cloud by rectangular voxel grid and then groups neighboring voxels into supervoxels.

A drawback of the method proposed in (Papon et al., 2013) is that the size of the obtained supervoxels is defined by a user specified parameter, which has to be selected according to previously known properties of the scene or objects of interest.

We developed a method which segments a 3D point cloud into approximately planar surface segments, where a user can specify the maximum deviation of points grouped in a segment form a plane assigned to this segment. This segmentation has the same underlying principle as the segmentation method proposed in (Holz and Behnke, 2014). It is based on region growing and it requires a 3D triangular mesh as input. A seed point is randomly selected from the input mesh. The region growing procedure is initialized by creating a region consisting of the selected seed point and growing this region by adding neighboring points which satisfy the criterion that the point must lie on the plane defined by the seed point and its normal within a predefined threshold. After the region growing procedure is completed, i.e. if no more points which satisfied the planarity criterion can be added to the grown region, a new randomly selected mesh point, which currently isn't assigned to any planar segment, is used as the seed for the next region growing process. The advantage of this approach in comparison to the method proposed in (Papon et al., 2013) is that the size of the obtained segments is determined by the scene geometry. Large planar surfaces in a given scene are represented by few large segments, while highly curved surfaces are represented by multiple small segments. Another advantage in comparison to (Papon et al., 2013) is that the points of a surfel lie on a planar surface within a user specified tolerance. The main difference between our approach and the method presented in (Holz and Behnke, 2014) is that the segment boundaries obtained by our approach are fit to the intersection between the planes assigned to adjacent segments and thus represent a better approximation of a polygonal representation of the given point cloud.

2. Notation

Notation presented in this section is related to general concepts such as point positions, local surface normals, planes etc., and it is used in all technical and scientific documents representing results of ARP3D project.

Points in a 3D point cloud are represented by P. The position of the i-th point P_i in a point cloud w.r.t. to a particular reference frame (RF) is represented by vector $\mathbf{p}_i$. All algorithms developed in ARP3D project use local surface normals assigned to the points in the input point cloud. The local surface

normal assigned to the point P_i is represented by a unit vector $\mathbf{n}_i$. Most of the algorithms developed in the project are based on planar segments. 3D planes are represented pairs $(\mathbf{n}, d)$, where $\mathbf{n}$ is the plane normal represented w.r.t. a particular reference frame and d is the plane distance from the origin of the considered reference frame in direction of the plane normal. Hence, a plane is defined by equation

$$\mathbf{n}^T \cdot \mathbf{p} = d \,, \tag{1}$$

where $\mathbf{p}$ represents a point lying on the plane. A planar segment is represented by F. The plane supporting a planar segment F_j is defined by parameters $(\mathbf{n}_j, d_j)$.

The pose of a reference frame S_A relative to a reference frame S_B is commonly represented by a homogenous transform matrix ${}^B\mathbf{T}_A$ defined by

$${}^B\mathbf{T}_A = \left[\begin{array}{ccc|c} & {}^B\mathbf{R}_A & & {}^B\mathbf{t}_A \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

where ${}^B\mathbf{R}_A$ is a rotation matrix defining the orientation and ${}^B\mathbf{t}_A$ is a translation vector defining the position of S_A relative to S_B.

A point cloud segmentation process divides a point cloud into subsets, referred to in this paper as *segments*. Number of points grouped in a segment is an important feature of the segment since it indicates its salience. Number of elements of a set C is denoted by $|C|$.

Many methods developed in ARP3D project are based on graphs. A *graph* is a set of nodes connected by edges. The *neighborhood* of a node A, denoted in this paper by $\mathcal{N}(A)$ is a set of nodes connected to A by edges.

3. Method

Our segmentation approach is described by Algorithm 1. An Input to the algorithm is a triangular mesh T obtained from a given 3D point cloud. This mesh can be created by one of the available methods for creating triangular meshes from point clouds. If an organized point cloud, such as a depth image obtained by a RGB-D camera, is used as input, then a triangular mesh can be created by the mesh construction algorithm presented in (Holz and Behnke, 2012), which is included in the Point Cloud Library (PCL) (Rusu and Cousins, 2011). The presented algorithm requires local surface normals to be assigned to the points of the input mesh T. We use PCL to compute local surface normals. The output of the algorithm is a set $\mathcal{F}$ of disjoint segments F, each representing a cluster of points from the input point cloud which lie on a plane within a given tolerance σ_p. Let Pi be a point of the input mesh T whose position w.r.t. the mesh reference frame is defined by vector $\mathbf{p}_i$ and whose local surface normal is defined by unit vector $\mathbf{p}_i$. This point can be assigned to a segment F_j only if the following two criteria are satisfied

$$\left|\mathbf{n}_j^T \mathbf{p}_i - d_j\right| \le \sigma_p \,, \tag{2}$$

$$\left\|\mathbf{n}_i - \mathbf{n}_j\right\| \leq \sigma_n \ , \tag{3}$$

where $\left(\mathbf{n}_j, d_j\right)$ are the parameters of the supporting plane of F_j. Parameters σ_p and σ_n are inputs to the discussed algorithm.

Algorithm 1 *Point Cloud Segmentation into Planar Surface Segments*

Input: T, σ_p, σ_n
Output: $\mathcal{F}$

1 : $\mathcal{F} \leftarrow \varnothing$
2 : $A \leftarrow T$
3 : $j \leftarrow 1$
4 : **Repeat**
5 : Randomly select a point P_i.
6 : $F_j \leftarrow P_i$
7 : Region growing process which grows F_j by adding adjacent points satisfying criteria (2) and (3). Only the points which are not already assigned to a previously created segment are considered in this process.
8 : $F_j \leftarrow$*ExpandSegment*(F_j, T)
9 : Add F_j into $\mathcal{F}$.
10 : Remove all points grouped in F_j from A.
11 : $j \leftarrow j + 1$
12 : **until** there are no points in A.
13 : **return** $\mathcal{F}$

Procedure *ExpandSegment* in line 8 of Algorithm 1 expands segment F_j into adjacent segments in a way that the resulting boundary between two adjacent segments is closely fit to the separation line between the supporting planes of these two segments. The basic principle of this procedure is explained by Algorithm 2. The algorithm creates a new segment F, which is initially a copy of the input segment F_{in}, which is then expanded into its adjacent segments. The algorithm considers the adjacent segments one by one and performs a region growing procedure, where F is grown by adding adjacent points of a neighboring segment F' which satisfy criterion (2). Condition (3) is omitted in this expansion because normal are often corrupted by noise and, therefore, including the condition (3), which considers normals, results in noisy segment boundaries. An example of a segment F and its adjacent segment F' is given in Fig. 1(a) and the result of the region growing process, which expands F into F' is shown in Fig. 1(b), where the gray region G contains points which satisfy criteria (2) and (3) for both segments. Then, the boundary of G is examined and the segments of this boundary which connect F and $F'' = F' \setminus G$ are identified. These boundary segments are denoted in Fig. 1(c) by red and blue lines. The proposed algorithm then searches for an optimal cut which separates F from F'' starting from one of the two discussed boundary segments, which is denoted in Fig. 1(c) by a red line and finishing in the other boundary segment, denoted in Fig. 1(c) by a blue line. The proposed algorithm is designed to find a cut which mostly lies on the separation plane of the segments F and F'. For two adjacent segments F and F', *separation plane* is defined as the plane which contains the intersection line of the supporting planes of F and F' and which is oriented in such a way that the angles between the normals of these two supporting planes and the separation plane are equal. The normal $\mathbf{n}_{IS}$ and distance d_{IS} of the separation plane of segments F and F' are computed by

$$\mathbf{n}_{IS} = \frac{\mathbf{n} - \mathbf{n}'}{\left\|\mathbf{n} - \mathbf{n}'\right\|} \ , \tag{4}$$

$$d_{IS} = \frac{d - d'}{\left\| \mathbf{n} - \mathbf{n}' \right\|}. \tag{5}$$

Algorithm 2 *ExpandSegment*

Input: F_{in}, T
Output: F

$F \leftarrow F_{in}$
$Q \leftarrow$ list of segments adjacent to F
Repeat
  $F' \leftarrow$ a segment taken and removed from Q.
  Region growing process which grows F by adding adjacent points of F' satisfying criteria (2) and (3). The result is a point set G containing all added points.
  $F'' \leftarrow F' \setminus G$
  $F'' \leftarrow Connect(F'', T)$
  $(\mathbf{n}_{IS}, d_{IS}) \leftarrow$ separation plane of F and F' satisfying computed by (4) and (5)
  $\mathcal{G} = \{G_1, G_2, \dots\} \leftarrow$ connected components of G
  $\mathcal{E} \leftarrow \varnothing$
  For every $G_i \in \mathcal{G}$
    $B \leftarrow$ boundary of G_i
    $N \leftarrow$ set of adjacent points of B, which are not elements of G
    If $N \cap F \neq \varnothing$
      $\mathcal{E}' \leftarrow$ optimal cut of G_i, where the cut cost is computed using $(\mathbf{n}_{IS}, d_{IS})$
      $\mathcal{E} \leftarrow \mathcal{E} \cup \mathcal{E}'$
    end for
  end for
  Region growing process which grows F by adding adjacent points of G until reaching $\mathcal{E}$.
  $F' \leftarrow F'' \cup (G \setminus F)$
  Update Q by identifying segments adjacent to the expanded segment F.
until there are no segments in Q.
return F

Note that the separation plane between two planar segments is defined only if their supporting planes are not parallel. The separation plane is denoted in Fig. 1(c) and (d) by a green line.

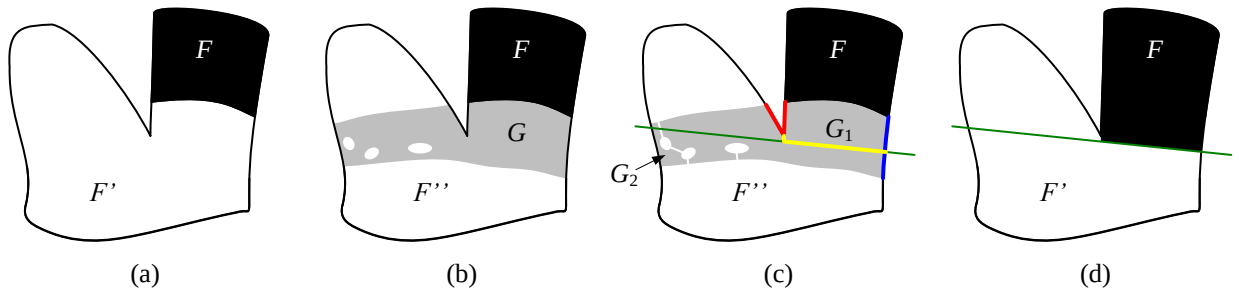

Fig. 1. Expansion of segment F into segment F'.

The cost of a cut is defined as the sum of costs of cutting an edge between adjacent points in the input triangular mesh T, where the cost of cutting an edge separating two points on different sides of the separation plane is 0, while the costs of cutting other edges are 1. The optimal cut is determined by a wave propagation algorithm analogous to the algorithms for searching for the shortest path in mobile

robot path planning. An example is shown in Fig. 1(c), where an optimal cut is depicted by a yellow line. Notice that the cut lies mostly in the separation plane. Finally, all points of set G positioned on one side of the obtained cut are assigned to F, while the points of G lying on the other side of the cut are assigned to F', as illustrated in Fig. 1(d).

It is possible that set F'' consists of several connected components, as illustrated in Fig. 1(b). In order to ensure that the resulting cut separates points of F from points of F'', the connected components of F'' are connected into a single connected component. This step is represented by procedure *Connect* in line 7 of Algorithm 2 and described by Algorithm 3. It is performed by forming a graph whose nodes are the connected components of F'' and edges connect neighboring connected components. Then a minimum spanning tree of this graph is determined, where the costs of edges are the distances between the corresponding connected components. Finally, the connected components are connected by paths corresponding to the edges of the minimum spanning tree. An example is shown in Fig. 1, where the connected components of F'' shown in Fig. 1(b) are connected into a single connected component shown in Fig. 1(c). The procedure of connecting set F'' into a single connected component.

Algorithm 3 *Connect*

Input: F_{in}, T
Output: F

1 : $F \leftarrow F_{in}$
2 : $\mathcal{C} = \{C_1, C_2, \dots\} \leftarrow$ connected components of F
3 : $\mathcal{G} \leftarrow$ graph whose nodes represent the connected components $C_i \in \mathcal{C}$ and edges E_{ij} connect the neighboring connected components C_i and C_j. Each edge E_{ij} is assigned a cost representing the length of the shortest path connecting C_i and C_j.
4 : $\mathcal{E} \leftarrow$ edges of $\mathcal{G}$ belonging to a minimum spanning tree of $\mathcal{G}$.
5 : **For** each edge $E_{ij} \in \mathcal{E}$
6 : $\quad \mathcal{P} \leftarrow$ shortest path consisting of points from T connecting C_i and C_j
7 : $\quad F \leftarrow F \cup \mathcal{P}$
8 : **end for**
9 : **return** F

Furthermore, G is not necessarily a connected set. Hence, search for optimal cut must be performed for every connected component of G which is adjacent to both F and F''. Note that there can be connected components of G which are completely surrounded by F''. An example is shown in Fig. 1(c), where connected component G_1 is adjacent to both F and F'', while G_2 is completely surrounded by F''.

4. Implementation

The proposed method is implemented as a part of the Robot Vision Library (RVL) created at J. J. Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technologies Osijek (FERIT). RVL is publically available at the Project web page http://www.etfos.unios.hr/ARP3D. The library is created in C++ programming language using Microsoft Visual Studio 2013. It requires the following third-party libraries: Visualization Toolkit (VTK), OpenCV, OpenNI, Eigen, RapidXML and PCL.

The proposed method is implemented by class *PlanarSurfelDetector* contained in library *RVLPCSegment.lib*. *PlanarSurfelDetector* is a program tool which takes a triangular mesh as input and produces a graph whose nodes are planar segments. Adjacent segments are connected to each other by edges. This graph is represented by class *SurfelGraph*.

RVL includes a demo application *RVLPCSegmentDemo.exe* whose purpose is to demonstrate functionality of the method presented in this paper as well as some other methods developed in ARP3D project. In order to configure this application for demonstration of the method presented in this paper, configuration file *RVLPCSegmentDemo_ARP3D.TR3.cfg* must be renamed to *RVLPCSegmentDemo.cfg* before running *RVLPCSegmentDemo.exe*. For details of usage of RVL, please contact the Project PI at robert.cupec@etfos.hr.

The result of applying the proposed method to a depth image from the TUW dataset used in (Aldoma et al., 2012b) is shown in Fig. 2.

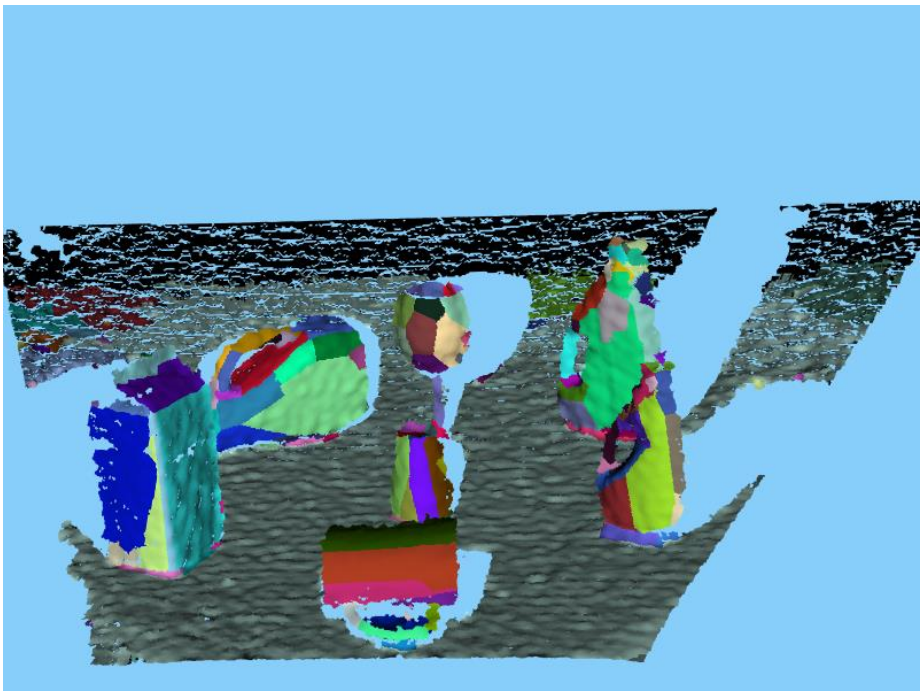

Fig. 2. A depth image segmented into planar segments depicted in different colors.

References

Achanta R, Shaji A, Smith K, Lucchi A, Fua P and Süsstrunk S (2012) SLIC Superpixels Compared to State-of-the-Art Superpixel Methods, IEEE Transactions On Pattern Analysis And Machine Intelligence, vol. 34, no. 11, pp. 2274–2281

Aldoma A, Tombari F, Di Stefano L and Vincze M (2012b) A global hypothesis verification method for 3d object recognition, *European Conference on Computer Vision (ECCV)*

Gupta S, Arbeláez P, and Malik J (2013) Perceptual Organization and Recognition of Indoor Scenes from RGB-D Images, *Computer Vision and Pattern Recognition (CVPR)*

Holz D and Behnke S (2012) Fast Range Image Segmentation and Smoothing using Approximate Surface Reconstruction and Region Growing, *International Conference on Intelligent Autonomous Systems (IAS)*, Jeju Island, Korea

Holz D and Behnke S (2014) Approximate Triangulation and Region Growing for Efficient Segmentation and Smoothing of Range Images, Robotics and Autonomous Systems, vol. 62, no. 9, pp. 1282–1293

Papon J, Abramov A, Schoeler M and Wörgötter F (2013) Voxel Cloud Connectivity Segmentation - Supervoxels for Point Clouds, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2027–2034.

Rusu RB and Cousins S (2011) 3D is here: Point Cloud Library (PCL), *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China

Simari P, Picciau G and De Floriani (2014) Fast and scalable mesh superfacets. *Computer Graphics Forum*, vol. 33, no. 7, pp. 181–190.