A brief guide to iterative two-stage segmentation

by Jaudiel S. Vélez Robles, ERDC-TEC, Alexandria, VA

1. System requirements:

- Operating system: Ubuntu 10.10 or higher (either native or virtualized)
- Libraries: PCL 1.1 or higher, and its dependencies
- CMake 2.8 or newer

2. Overview:

Iterative two-stage segmentation (I2SS) is a two-pass algorithm that, given a point-cloud of arbitrary size and geometry, first tries to fit as many planes as possible using a sample consensus method and subtracts them from the original cloud. Then, it takes the remaining points in the cloud and groups them all into clusters according to how close they are to each other and extracts them as well. Each extracted object is saved into separate point-cloud data (.pcd) files for easier simultaneous visualization.

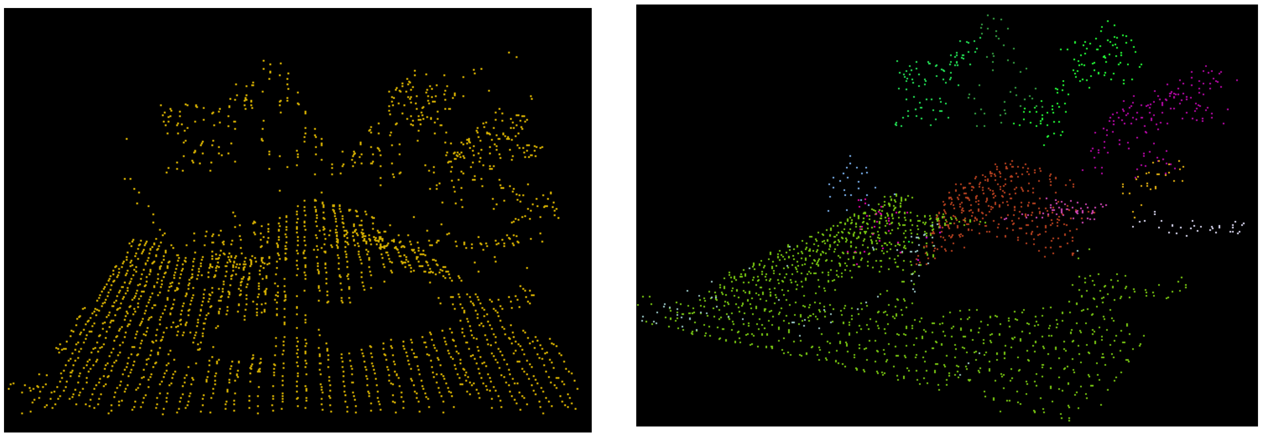


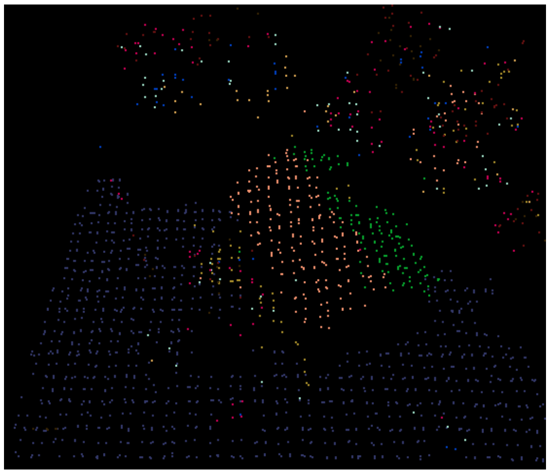
Fig. 1: A point-cloud representation (left) of a roughly 15x15m area containing a warehouse and surrounding trees, taken from a LiDAR scan at 1m resolution above Fort Leonard Wood, Missouri. The image on the right shows the individual segments after applying the I2SS algorithm.

The algorithm itself is structured as follows:

```
I2SS(P,method,model,delta,k,sigma,ctol,cmin,cmax,nwt):
      remove duplicate points from the point-cloud P
      if model is a normal plane:
            estimate point normals for P
            while a sufficient number of points from P still remain:
                  segment the largest component whose normals consistently
                    point in the same direction
                  write the segmented plane to disk
                  subtract the segmented points and respective normals from P
            endw
      else:
            while a sufficient number of points from P still remain:
                  segment the largest planar component from P
                  remove outliers from the extracted segment
                  write the segmented plane to disk
            endw
      endif
      extract points close enough to each other into individual clusters
      for each extracted cluster in P:
            write all the points in that cluster to a separate cloud
            write the cloud to disk
      endfor
end I2SS
```

To ensure an accurate, topologically consistent segmentation, there are several parameters one must consider and adjust according to the problem:

- The choice of sample consensus estimator is an important factor to evaluate. For data-sets where it is easy to visualize which points belong to what objects, RANSAC and MLESAC perform fairly similarly. However, in cases where the distinction between objects is not as apparent, MLESAC is often the better choice.
- δ, the distance threshold used by the selected sample consensus method. As this value increases, points farther apart will be considered in the calculation of the fitted plane. A high δ means you'll end up fitting solids instead. On the other hand, too low a value will probably confuse the sample consensus estimator into fitting what might actually be a single plane into multiple 'layers' with sparse point density.
- *k*, the number of nearest neighboring points to take into account for statistical outlier removal. Higher values mean more points are searched, but increases computing cost.
- σ, the minimum distance between a point and its k-nearest neighbors for it to not be considered an outlier, in multiples of the standard deviation of the average distance between all k-neighboring points. A low enough value increases the chance that points will be considered outliers.
- *ctol*, the distance threshold for a point to be considered as part of a cluster. Similar to δ, higher values are more prone to overlapping another cluster or object.
- *cmin/cmax*: the minimum and maximum number of points in a group for it to be considered a cluster. Broader ranges mean more clusters will be detected.
- *nwt*, the angular distance weight when estimating a plane through point normal vectors expressed as a floating-point number between 0 and 1 (inclusive). A value of 1 represents an angular threshold of $\pi/2$.



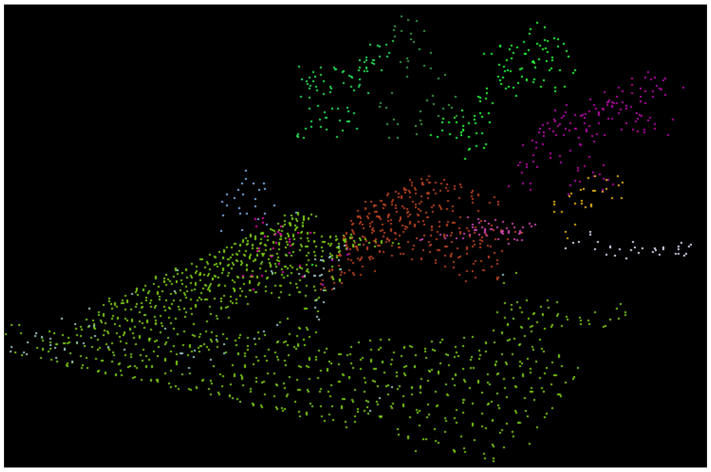


Fig. 2: Two different segmentations of the warehouse area achieved by using RANSAC (left) and MLESAC (right). Notice how RANSAC fit multiple planes comprised of sparse points from completely different objects, while the MLESAC attempt achieved near-perfect segmentation.

Again, note that the term segment refers to a group of points that fits reasonably well to some specified mathematical model, whether a plane, a line, a circle, or any other arbitrary geometric or polygonal surface. On the other hand, a cluster denotes a group of points packed together within a certain threshold distance of each other that can potentially, but is not necessarily guaranteed to represent an arbitrary surface or object of interest; in layman's terms, they are what's left after all segments in a point-cloud have been extracted.

3. Usage instructions:

First, open a terminal and navigate to the folder containing the executable, then type the following:

```
./cluster_subtraction <input_cloud>.pcd delta
Options: --sor k sigma                  (default: 50 1.0)
         --cluster ctol cmin cmax       (default: 0.02 250 25000)
         --method [ransac|mlesac]       (default: ransac)
         --model [plane|normal_plane]   (default: plane)
         --normal_weight nwt            (default: 0.1)

         *Note: --sor is unnecessary when estimating normal planes.
          Likewise, --normal_weight is unnecessary during regular
          estimations.
```

Then, to visualize your results:

```
pcd_viewer <method-type>*.pcd
```

A window should appear showing each of the output components in different colors. You can rotate the viewpoint by simply clicking and dragging inside the window. The middle scroll wheel controls the camera's zoom. You can also press J to take a snapshot, and G to show the scale grid. Pressing H brings up the help menu.

Depending on which version of PCL is installed on your computer, the viewer may show a black screen. If this happens, simply press the R key to re-center the camera, and everything should be fine. Also, even though all output segments and clusters are technically rendered in different colors, sometimes the viewer renders two objects in close proximity with similar shades of the same color, making it difficult to distinguish which one is which. In this case, press 1 to cycle the colors at random until you achieve the desired palette.