3D OBJECT SEGMENTATION OF POINT CLOUDS USING PROFILING TECHNIQUES

George Sithole
University of Cape Town
33 PUBLICATIONS **2,257** CITATIONS
SEE PROFILE

Willard Mapurisa
Luxcarta
10 PUBLICATIONS **30** CITATIONS
SEE PROFILE

3D OBJECT SEGMENTATION OF POINT CLOUDS USING PROFILING TECHNIQUES

G. Sithole [1], [*]W.T. Mapurisa [2]

[1]Geomatics Division, University of Cape Town, South Africa, George.Sithole@uct.ac.za

[2]Research and Development, ComputaMaps, South Africa, wmapurisa@computamaps.com

[*]Research undertaken at the University of Cape Town

Abstract

In the automatic processing of point clouds, higher level information in the form of point segments is required for classification and object detection purposes. Point cloud segmentation allows for the definition of these segments. Various algorithms have been proposed for the segmentation of point clouds. The advancement of Lidar capabilities has resulted in the increase in volumes of data captured by Laser Scanners, thus impacting negatively on existing algorithms in relation to computational cost. Consequently, fast and reliable segmentation methods are being sought. In this paper, an extension of a segmentation approach based on intersecting profiles is proposed. In the presented method, surfaces are considered as a graph of intersecting planar curves as opposed to surface patches. In this graph structure the curves intersect at common points and terminate at surface discontinuities. This property of the curves makes it possible to determine point segments by connected components.

A method for the detection of curves in the profiles is presented. The algorithm is tested on three terrestrial lidar point clouds. Experiments were done to test the effectiveness of the algorithm.

1. Introduction

The last two decades have seen an increase in the number of devices and spatial data acquisition techniques that produce 3D point clouds. In the same period the resolution, accuracy, acquisition rate and size of these devices have increased. In tandem this has led to a marked increase in the resolution, accuracy and size of point clouds. This trend is set to continue with devices and techniques becoming cheaper and more accessible to the general public. Presently, the biggest challenge associated with 3D point clouds is their size. Ten years ago a large lidar project contained tens of millions of points and occupied a few gigabytes of memory. Today a similar project will contain billions of points and occupy terabytes of memory. This size problem is overcome by a combination of data structuring, point reduction, segmentation and memory optimization strategies.

In data structuring the point cloud is arranged in a fashion that makes it easier to navigate. Data structuring is typically achieved by means of tree structures such as octrees and kd-trees.

Point reduction aims to remove redundant points in a point cloud thus reducing the computational load at the source. In the segmentation strategy higher order features are extracted from a point cloud so that user interaction is with these features rather than the points themselves. Naturally there will be fewer features than points and greater efficiency is earned by having to manipulate these fewer features.

Memory optimization attempts to physically store the point cloud so that storage, retrieval and visualisation of the point cloud are as efficient as possible. This typically involves the use of out-of-core algorithms and pushing the point cloud onto the graphical processing unit (GPU) of a computer.

Segmentation is the partitioning a point cloud into regions or segments with homogeneous geometric or radiometric characteristics. Segmentation is a non-trivial problem because an absolute definition of a ‘good’ segment does not exist and very often within the same point cloud features are best represented by segments with differing geometric or radiometric characteristics. In effect the appropriateness of segments in segmentation is determined by the application of the point cloud and the local variations in the point cloud. To appreciate this it must be firstly borne in mind that applications operate on features such as edges, surfaces, facets or objects in a point cloud. All of these features are implicitly present in a point cloud. Presently, segmentation algorithms differ in their ability to extract these features. A segmentation algorithm that extracts surfaces is generally unsuited for the extraction of objects. Secondly, the information density in a point cloud is non uniform. The result of this is that identical objects in different locales in a point cloud will be represented differently. For example in a point cloud obtained with a terrestrial scanner, an object near the scanner will enjoy a greater fidelity than an object far away because the point resolution recedes in direct proportion with the distance from the scanner. A robust segmentation algorithm has to be able to accommodate such local variations.

The flexibility and efficiency of a segmentation algorithm is founded on the data structure used in containing and navigating the point cloud. Presently, space-partitioning data structures (octrees, kd-trees, etc.,) and Triangular Irregular Networks (TIN) are the preferred data structure for segmenting point clouds.

This paper presents the extension of a segmentation algorithm developed by Sithole (2005). The paper is structured into four sections. The first section presents a review of current segmentation algorithms. In the second section of the paper the proposed algorithm is discussed. The third section of the paper presents results from applications of the algorithm. Finally a conclusion on the research is presented.

1.1 Previous work

Segmentation algorithms yield segments in the form of edges, surfaces, facets or objects. Of these edge and surface based algorithms are the most common.

1.1.1 Edge based algorithms

Edge based segmentation algorithms seek out edges in a point cloud and then identify segments by further seeking for edge loops (cyclic edges). Those points that exist within the same edge loop are then determined to belong to the same segment. Edge based algorithms also make used of TINs. Edges are discriminated using geometric properties such as principal curvature, facet normal and gradients (Yang and Lee, 1999; Woo et al, 2001; Fan et al, 1987).

1.1.2 Surface based algorithms

Surface based segmentation techniques typically begin by triangulating the point cloud. Next, seed triangles are randomly chosen. Triangles in the neighbourhood of these seed triangles are tested. Triangles that meet given similarity criteria are aggregated with the seed triangles. This aggregation forms the initial segment. This process of testing neighbouring triangles is repeated with the previously identified triangles being used as seed points. In this manner, called region growing, the entire point cloud is segmented. The similarity criteria are based on changes in surface geometric or radiometric (or a combination of both) properties. Geometric properties include curvature and facet normal’s, while radiometric properties include colour and shading.

The success of region growing depends on the selection of the initial seed points and the choice of similarity criteria. An example of region growing is presented by Rabbani et al. (2006), where segmentation is carried out using a smoothness constraint. Points are fit to planar surfaces and then merged based on residuals from the fit and point normals. Other examples are Lukas et al, (1998) and Besyl and Jain (1988).

1.1.3 Scan line algorithms

A less used algorithm is the scan line segmentation algorithm. Single raw scans obtained by devices such as terrestrial laser scanners retain their scan line geometry. Such point clouds are called range images or structured point clouds. Structured point clouds allow for a point in a scan line to be compared against other points in the same scan line and points in neighbouring scan lines. If the comparisons meet given geometric criteria (such as proximity) then the points are aggregated into the same segment. Typical examples are presented by Jiang et al, (1994) and Khalifa, Moussa and Mohamed, (2003).

When scans are combined, the scan line geometries are lost and scan line segmentation is no longer possible. To overcome this limitation Sithole (2005) proposed a data structure that creates artificial scan lines for an unstructured point cloud. He then devised a segmentation algorithm that employs the proposed data structure. This data structure and segmentation algorithm is discussed in more detail in section 2 and 3.

1.2 Analysis

All segmentation algorithms have limitations in their design. Primarily these limitations arise from the complexity of objects in a point cloud, the geometric and radiometric properties of the point cloud, and the representation of surfaces in the algorithms. Secondary factors are the size and organization of point clouds, and the computational overheads. Some of these limitations are discussed below.

A major problem with surface based segmentation is the presence of noise in point clouds. In a scan surfaces are supposed to be infinitely thin, but noise has the effect of making surfaces thick. This affects the accurate estimation of surface properties in region growing tests (Leonardis, 1993) often resulting in over segmentation. In edge based segmentation edges become more difficult to detect and this leads to under segmentation. To resolve the noise problem, hybrid methods which combine edge detection and surface based segmentation algorithms have been proposed (Yokoya and Levine, 1997; Checchin et al, 1997).

Surface based segmentation algorithms described surfaces using explicit surface functions (cylinders, planes, surface patches). This complicates the segmentation of implicit surfaces. Because of this algorithms are designed for specific scenes, for example, Rabbani et al, 2006, for industrial installations. As a result, there are very few generic segmentation algorithms.

Many segmentation algorithms perform neighbourhood searches. In the absence of a space partitioning scheme this can lead to unacceptable computational overheads, particularly for very large point clouds.

2. The data structure

A surface can be described by planar curves running across it. This is illustrated in figures 1a and b. If the curves are not parallel they will intersect. The resulting intersection points can themselves be used to describe the surface. Now, reversing this scenario, if the points on the surface are known then the curves can be reconstructed and used to describe the surface. Additionally, if the surface contains discontinues, then these discontinuities can be determined from the terminators (points) at the end of the curves. The data structure devised by Sithole (2005) is based on this idea.

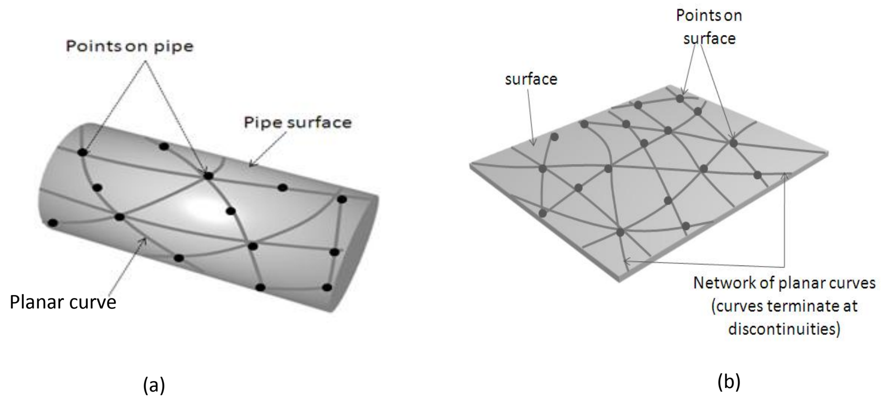


Figure 1. Network of planar curves representing a surface

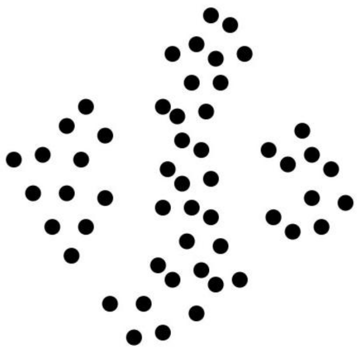
2.1 The stack and profiles

In unstructured point clouds, points are not arranged along planar curves. To overcome this problem the point cloud is sliced into thin contiguous profiles. This is shown in figure 2. These profiles are collectively called the 'stack'. The thin profiles are approximations of planar curves and the points that lie inside a profile are assumed to belong to the planar curve represented by the profile. The (profiles in a) stack is aligned in space along a vector called the 'stack normal' represented by the symbol φ.

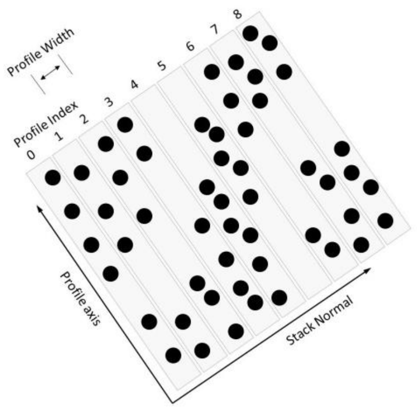
Once the stack has been created the next step is to connect the points in each profile so as to approximate the planar curves within the profile. The stack can now be defined by a graph $G_\varphi(V, E_\varphi)$ where V is the set of all the points in the point cloud and E_φ is the set of all edges in all the profiles in the stack. Naturally every profile in a stack is a sub graph of $G_\varphi(V, E_\varphi)$. The profile graph is represented by $G_{\varphi,p}(V_p, E_{\varphi,p})$ where p is the index ($p = 0, 1, 2, 3 \ldots n-1$, where n is the total number of profiles in a stack) of a profile in the stack, V_p is the set of points in a profile and $E_{\varphi,p}$ is the set of edges in a profile after a profile segmentation. By necessity $V_p \in V$ and $E_{\varphi,p} \in E_\varphi$.

2.2 The stacks

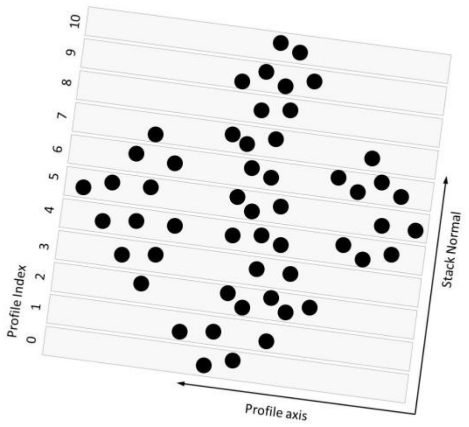
The previous section described the creation of a single stack. To complete the data structure other stacks are created along different stack normal's. These stack normal's are spread as far apart as possible within a hemi-sphere. Together the stacks can now be described by a graph G(V, E) where V is the set of all the points in the point cloud and E is the set of all the edges in all the stacks. Because of this $E_\varphi \in E$ and as a consequence G_φ is now a sub graph of G. The effect of this is that by necessity profiles in the different stacks will intersect. Put differently, every point will be contained in a single profile in every stack. The curve and surface reconstruction concept described in the first paragraph of section 2 is now realised. This is shown in the overlay of stacks in figure 2.



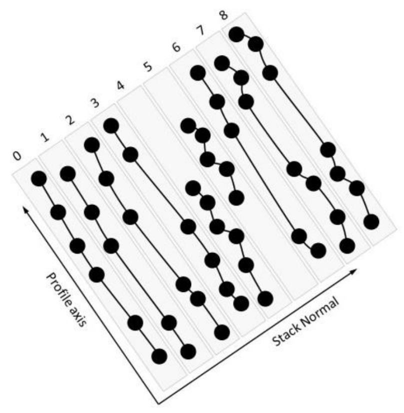
Point Cloud



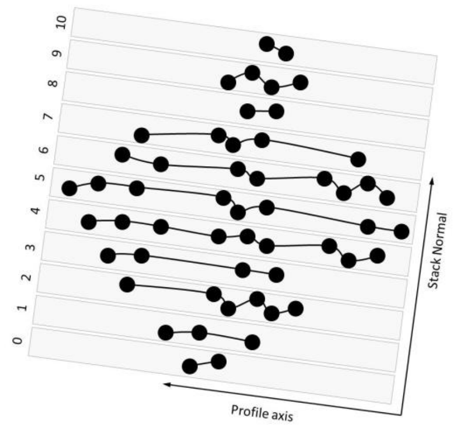
Stack 1 and its profiles 0 to 8



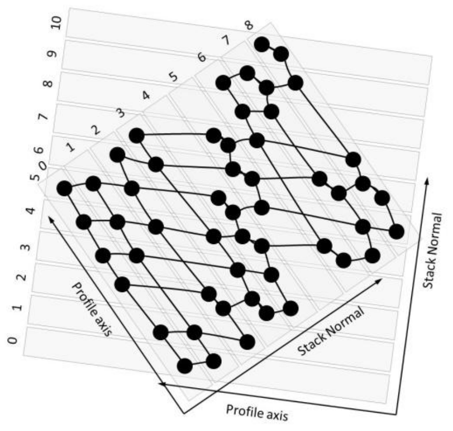
Stack 2 and its profiles 0 to 10



Connect points in the profiles



Connect points in the profiles



Overlay of stacks 1 and 2

Figure 2. Stacks and profiles defined on a point set.

2.3 Summary

In summary the data structure has the following properties:

1. The data structure is represented by a graph G(V, E)
2. The graph G(V,E) contains sub graphs $G_{\varphi}(V, E_{\varphi})$. These sub graphs are called the stacks. These stacks are aligned along a vector φ called the stack normal.
3. The stack contains sub graphs $G_{\varphi,p}(V_p, E_{\varphi,p})$. These sub graphs are called the profiles.
4. No two profiles in a stack share common points or common edges. Therefore the intersection of two profiles is an empty set.

   $G_{\varphi,pi}(V_{pi}, E_{\varphi pi}) \cap G_{\varphi,pj}(V_{pj}\ E_{\varphi,pj}) = \emptyset$

   Where:

   pi and pj are the indices of profiles in the stack aligned along the vector φ.
5. The intersection of profiles in two different stacks is a single point.

   $G_{\varphi m,pi}(V_{pi}, E_{\varphi m,pi}) \cap G_{\varphi n,pj}(V_{pj}\ E_{\varphi n,pj}) = v$

   Where:

   φm and φn are the vector directions of two different stacks and v is a single point in V

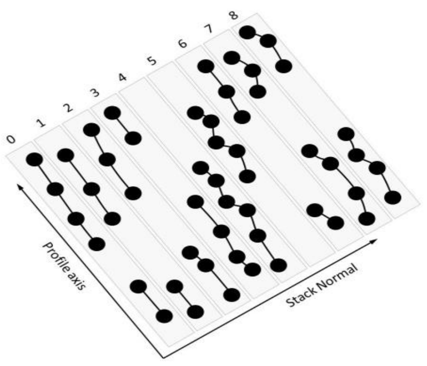
Together these properties make it possible to traverse the data structure or search point neighbourhoods in G(V, E). Importantly the data structure contains cross sectional views of the point cloud. These properties are now harnessed to segment the point cloud.

3. The segmentation algorithm

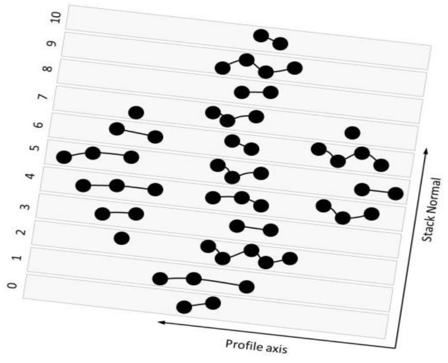
After creating the stacks the connected points in the profiles are segmented. The choice of segmentation is user defined. Examples of profile segmentation strategies are segmentation by point proximity, segmentation of minimum spanning trees and segmentation by curve fitting. Profile segmentation algorithm is discussed in section 4.1. After profile segmentation the overlay of the stacks now yields segments, or graphs $G_S(V_S, E_S)$ that are sub graphs G(V, E). This is shown in figure 3. Extracting the segments is achieved by performing a connected components analysis on G.

3.1 Discussion

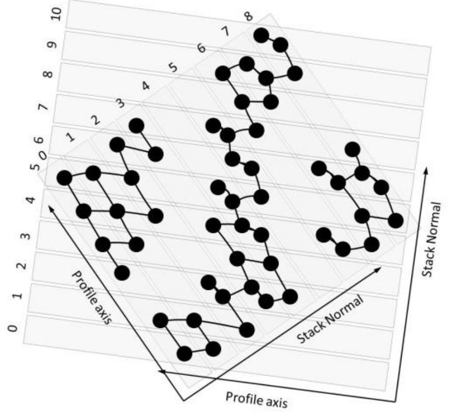
Classical approaches to segmentation first create surface meshes (or skins) on the point cloud. This is a non-trivial problem because it typically requires a nearest neighbourhood analysis, a Delaunay triangulation and finally a culling of triangles to obtain the skin mesh. It is the skin mesh that is then segmented. The segmentation described in section 2.3 is a 2D solution to a 3D problem. By reducing the problem from 3D to 2D complexity of the segmentation problem is substantially reduced. This is the main advantage of the segmentation approach.



Segmentation of profiles in Stack 1



Segmentation of profiles in Stack 2



Overlay of stacks 1 and 2

Figure 3. Segmentation of the point cloud

Another powerful feature of the data structure is that it allows for cross sectional analysis of the point cloud. The effect of this is that the segmentation can be tuned to yield edges, surfaces, facets or objects. A disadvantage of the data structure is that it maybe not be as compact or memory efficient as other data structures such as TINs. The storage requirements increase with the number of stacks created. Ideally three stacks should be sufficient, but in practice four or five are used.

3.1.1 Comparison to other common data structures

Data structures such as TINs are constructed based on rules and are typically parameter less. For example a Delaunay triangulation requires that a hyper sphere defined by three points not contain any other points. These data structures impose a graph on the point cloud, thus permitting an immediate traversal of point neighbourhoods. The effect of this is that bread first searches of point neighbourhoods are possible.

Space partitioning data structures such as kd-trees are defined by both rules and parameters. However, because of their hierarchical construction, an immediate traversal of point neighbourhoods is impossible. Discovering a point’s neighbourhood always requires a traversal of the tree’s hierarchy. The overheads associated with this traversal of the hierarchy are determined by the parameter of the tree’s construction, e.g., the depth of a tree determines the number of branches that have to be searched.

These two data structures demonstrate the competition between the need for a compact and progressive representation of a point cloud. TINs are more compact then they are progressive. Trees are more progressive then they are compact. The profile intersection algorithm attempts to balance these two needs. The data structure is compact in that the graph G(V,E) permits an immediate traversal of point neighbourhoods. The stack-profile organisation of the data structure lends it its progressive character (although this still requires further development). The efficiency of these two needs is determined by the profile width and the profile intersection.

3.1.2 Profile width vs. resolution

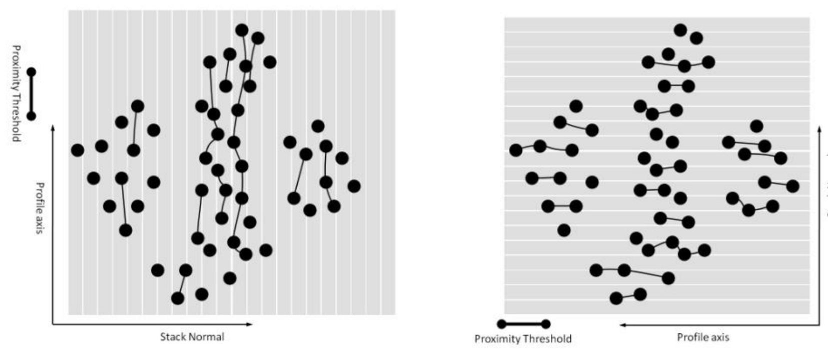
The choice of profile width depends on the spacing of points in the cloud. Ideally the profile width should be set such that the profile contains sufficient points to approximate the correct planar curve. This is shown in figures 2 and 3. However, very small and very large profile widths lead to over segmentations and under segmentations respectively. This is shown in figure 4. The minimum acceptable profile width is the average point spacing.

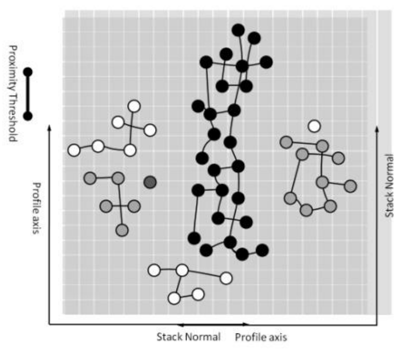
3.1.3 Profile segmentation

Profiles have to be segmented to isolate the points that belong to the same curve. If the profile segmentation is too generous then the profile will be under segmented. If the profile segmentation is too strict then the profile will be over segmented. Profile segmentation is further discussed in 4.1.

4. Extensions to the profile intersection algorithm

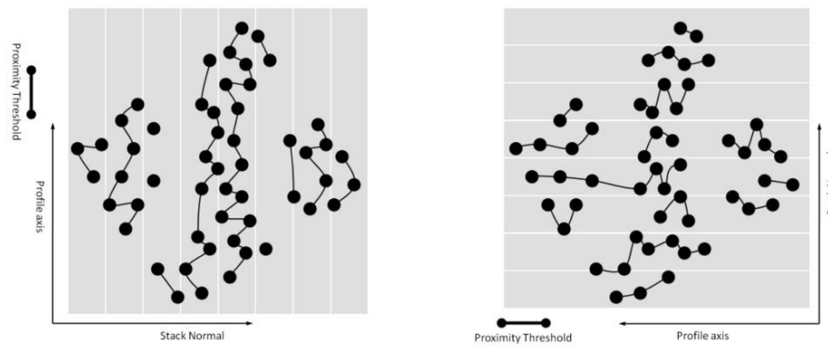
Sithole’s (2005) implementation of the profile intersection algorithm only allowed for a distribution of the stack normal’s around a horizontal disk. This was because he was working with airborne lidar data and vertical profiles were ideal for the analysis of urban landscapes. The first contribution of this paper to Sithole’s profile intersection algorithm is to spread the stack normal’s about a hemisphere. This extended distribution of the stacks is necessary for the analysis of point clouds such as terrestrial lidar scans. The second contribution of this paper is to segment the profiles using curve fitting algorithms. The paper’s third contribution to the profile intersection algorithm is the application of the algorithm to terrestrial scans.

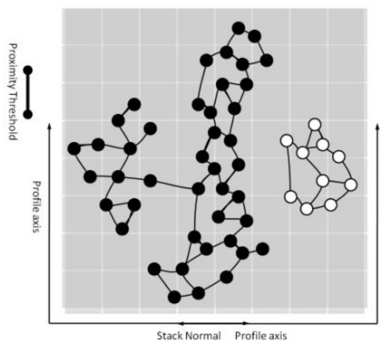




Stacks with a profile width 0.5 times that of the stacks in figure 3 Overlay of the stacks 1and 2

Result: Over segmentation





Stacks with a profile width 1.5 times that of the stacks in figure 3 Overlay of the stacks 1and 2

Result: Under segmentation

Figure 4. The effect of profile width on a point set. If profile width is made too small the profiles contain fewer points and the proximity threshold is exceeded and this results in an over segmentation. Conversely large profile widths lead to an under segmentation. Compare to figure 3.

4.1 Profile segmentation

As discussed in section 3.1.3 the profile segmentation yields chains of connected points representing the curves in a profile. This is shown in figure 5. The success of detecting the curves in a profile will determine the success of the segmentation. The manner in which points are chained together will also determine whether the final segmentation will yield edges, facets or objects. The following section will describe various profile segmentation algorithms.

4.1.1 Segmentation by proximity

In this segmentation algorithm points are first chained together by their proximity. Closest points are linked together. After the chaining, those links that are greater than a user defined threshold are removed thus yielding segmented profile points. A simple implementation of the algorithm is achieved by imposing a Delaunay triangulation on the points, followed by a minimum spanning tree of the triangular network, and finally a removal of the edges in the minimum spanning tree whose length exceeds a given threshold. This algorithm works well where there is a high density of points.

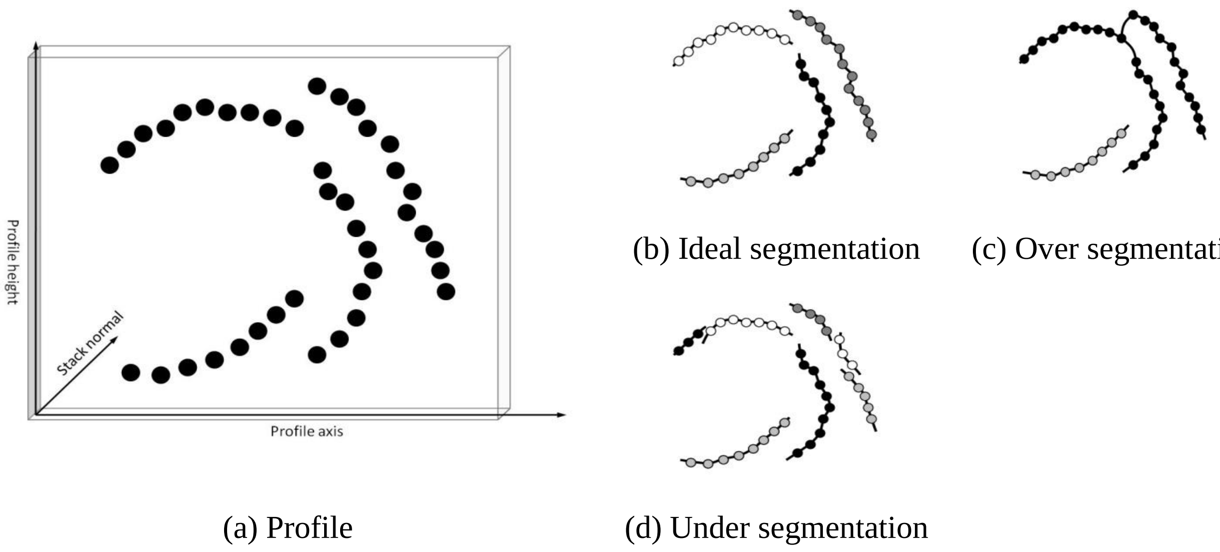
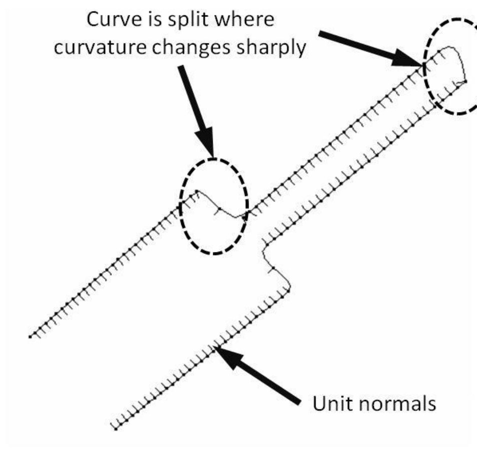


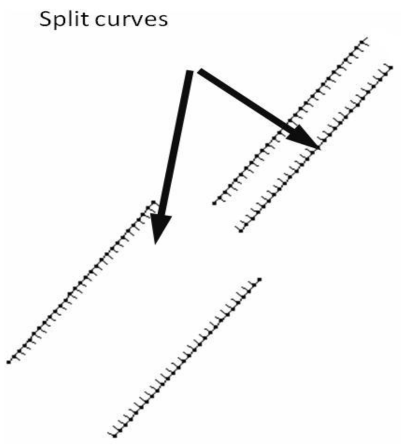
Figure 5. The choice of algorithm and parameters can yield a correct detection of the curves, a merging of the curves, or an over segmentation of the curves. (a) A single profile taken from a stack. Although the points are distributed along the width of the profile, they are treated as existing on a single plane defined by the profile axis (x axis) and the profile height (y axis), (b) ideal segmentation, (c) under segmentation, (d) over segmentation.

4.1.2 Segmentation by curve fitting

Proximity segmentation will fail where curves are entangled. It may also fail when the point density is low or when the noise in the point cloud is high. In these situations model fitting and model selection strategies have to be employed. The work in this paper uses such a strategy to detect the curves in a profile. A family of polynomials, the models, is selected. The points in the profile are fit to these models. A model selection criterion, in this case the Akaike Information criterion (AIC), is then used to identify the model that best fits the points. Literature on AIC is presented by Burnham et al. (2004). The selected fit is then tested to determine if it contains discontinues, noticeable by sharp changes in curvature. Where the curvature changes sharply the points in the profiles are divided into separate sets. The model fitting and selection is repeated until no new curves are detected. An example of this is shown in figure 6. Shown is a profile across a coaxial pipe. A curve is fit to the points. Next, the curve is tested for sharp changes in discontinuity along its length. Here the change of curvature is estimated by changes in the direction of the unit normals.



(a) Curve before splitting



(b) Curve after splitting

Figure 6. Shown here is a single profile across a coaxial pipe. (a) The points in the profile sufficiently represent four curves. (b) Discontinuities in the profile, evident by sharp changes in curvature, are used to discriminate the four curves.

5. Results

The segmentation algorithm was tested on three terrestrial lidar point clouds. The results of these tests are described below.

5.1 Sample 1: Building façade

This is a point cloud of the front of the Jameson Hall at the University of Cape Town. The point cloud (figure 7) contains large marble columns with a wall 3 m behind them. Placed into the wall are doors and windows. The average point spacing is of the sample 0.01m. The profile width is set to double the average point spacing. The aim on this data set was to separate the walls from the marble columns thus proximity segmentation is used. The wall segments are successfully segmented as shown in figure 7. Windows and doors are not segmented since proximity segmentation is used. A further segmentation by curve fitting is required in order to retrieve doors and windows as these are attached to the walls. The properties of the point cloud and the segmentation are summarised in table 1.

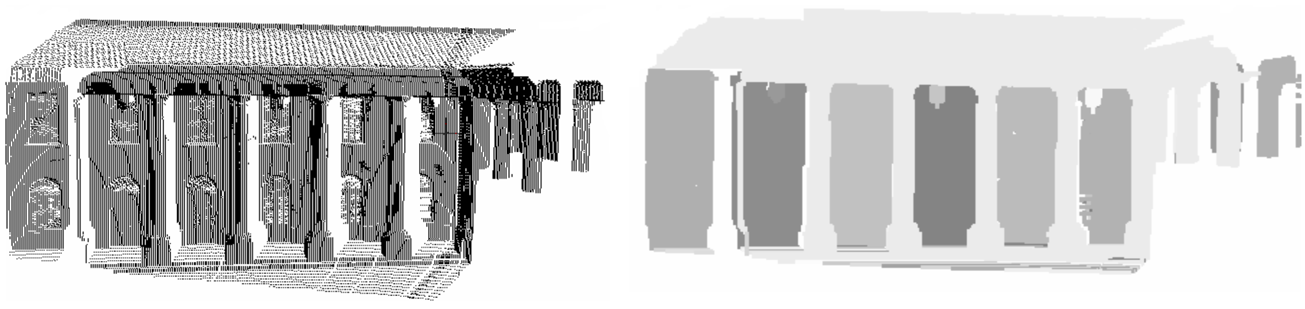


Figure 7. Sample 1 - Segmentation of a building façade

5.1.1 Sample 2: Piping installation

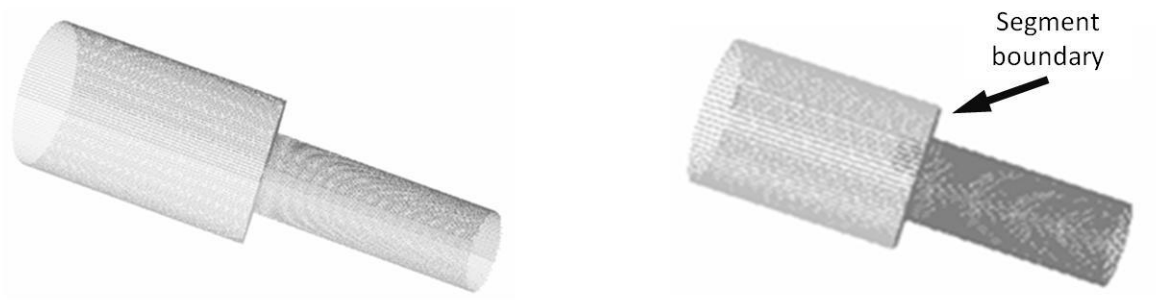
The second point cloud (figure 8) is that of a piping installation. Contained in this point cloud are pipes of varying sizes. In this instance the segmentation yields objects, in this case being the individual pipes. From the data structure it is possible to obtain the axes, radii values and the deformations in the pipes, but this will not be elaborated here. The average point spacing for this sample is 0.1m. The profile width was set to 0.2m. Both proximity and curve segmentation are employed in the segmentation. Figure 8 shows the set of pipe segments identified by the algorithm each of different radius. Erroneous points (noise) are detected as separate segments made up of small point clusters or a single point. The properties of the point cloud and the segmentation are summarised in table 1.

Point cloud
Segmentation

Figure 8. Sample 2 – Segmentation of a piping installation.

5.1.2 Sample 3: Coaxial pipes

The third point cloud (figure 9) is used to demonstrate the performance of the algorithm in the presence of surface discontinuities. Although the coaxial surfaces are connected, because of the profile segmentation algorithm the discontinuity in the surfaces is detected (see figure 6). The point spacing for this sample is 0.002m. Proximity and curve segmentation are used for this sample. The profile width is set to 0.009m. This width ensures sufficient points are captured is a single profile so as to identify boundaries correctly. The two separate sections of the pipe are identified and the segment boundary is shown in figure 9. Each segment is a pipe of constant radius. The properties of the point cloud and the segmentation are summarised in table 1.



Point cloud Segmentation

Figure 9. Sample 3 – Segmentation of a coaxial pipe

5.2 Discussion

For all the samples the profile width was set to a value of least one and a half times larger than the average point spacing. This avoids sparse profiles that lead to over segmentation. The distance threshold for proximity segmentation in all samples was set to a distance slightly larger than the average point spacing. A stack count of at least three was found to work well on all samples. However, the pipe point clouds contained a lot of noise and because of this a stack count of 4 was used. For the same reason profile segmentation by curve fitting was required for samples 2 and 3.

Table 1 Segmentation parameters

	Sample 1	Sample 2	Sample 3
Average point spacing (m)	< 0.01	0.1	0.002
Stack count	3	4	4
Profile width (m)	0.02	0.2	0.009
Proximity threshold (m)	0.01	0.15	0.004
Profile segmentation	Proximity	Proximity and curve fitting	Proximity and curve fitting

Curvature threshold (Degrees)	N/A	5	5

6. Conclusion

A segmentation method for point clouds using profiling techniques has been presented. The segmentation algorithm treats surfaces as a series of interconnected curves which terminate at discontinuities or surface boundaries. The algorithm identifies connected points on continuous curves in profiles using proximity and changes in curvature. The identified connected points are then overlaid to extract segments. The segmentation algorithm has the advantage of simultaneously discriminating objects in a point cloud, and relatively fast.

The data structure and segmentation algorithm have a range of applications. Future work will explore applications and improvements to the algorithm (such as the profile segmentation). Potential applications include edge detection, deformation and object classification.

7. References

Checchin, P., Trassoudaine, L., Alizon, J., 1997. Segmentation of range images into planar regions, *Proceedings of IEEE 3D Digital Imaging and Modeling, IEEE,* Ottawa, Canada, 1997, pp. 156–163.

Fan et al, 1987 – Segmented descriptions of 3-d surfaces, *IEEE J. Robot. Automation 3 (6)*, 527–538.

Jiang, X. and Bunke, H., 1994. Fast segmentation of range images into planar regions by scanline grouping, *Machine vision and applications*, vol. 7, no. 2, June, pp. 115-122.

Khalifa, I., Moussa, M. and Mohamed, K., 2003. Range image segmentation using local approximation of scan lines with application, *Machine Vision and Applications (2003) 13: 263–274*, vol. 13, 263–274.

Rabbanni, T., van den Heuvel, F. and Vosselman, G., 2006. Segmentation of point clouds using smoothness constraint', *ISPRS Commission V Symposium 'Image Engineering and Vision Metrology, September 25-27, Dresden, Germany*, pp. 248-253.

Sithole, G., 2005. Segmentation and classification of airborne Laser Scanner Data, *Phd Thesis, TU Delft*, 65-92.

Leonardis, A., 1993. Image analysis using parametric models: model-recovery and model selection paradigm, PhD dissertation, University of Ljubljana, Faculty of Electrical Engineering and Computer Science, May, pp. 672, 672.

Yang, M., Lee, E., 1999, Segmentation of measured point data using a parametric quadric surface approximation, Computer-Aided Design 31, pp. 449–457.

Woo, H., Kang, E., Wang, S. and Lee, K. H. 2001. A new segmentation method for point cloud data, *International Journal of Machine Tools & Manufacture,* vol. 42, pp. 167–178

Yokoya, N., Levine, M.D., 1997. Range image segmentation based on Differential geometry: a hybrid approach, *IEEE Transactions on Pattern Analysis and Machine Intelligence 11* (6) (1997) 643–649.