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

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. Segmentation allows for the definition of these segments. Because of the increasing size of point clouds faster and more reliable segmentation methods are being sought. Various algorithms have been proposed for the segmentation of point clouds. 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. In this graph structure 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 has been tested on terrestrial lidar point clouds.

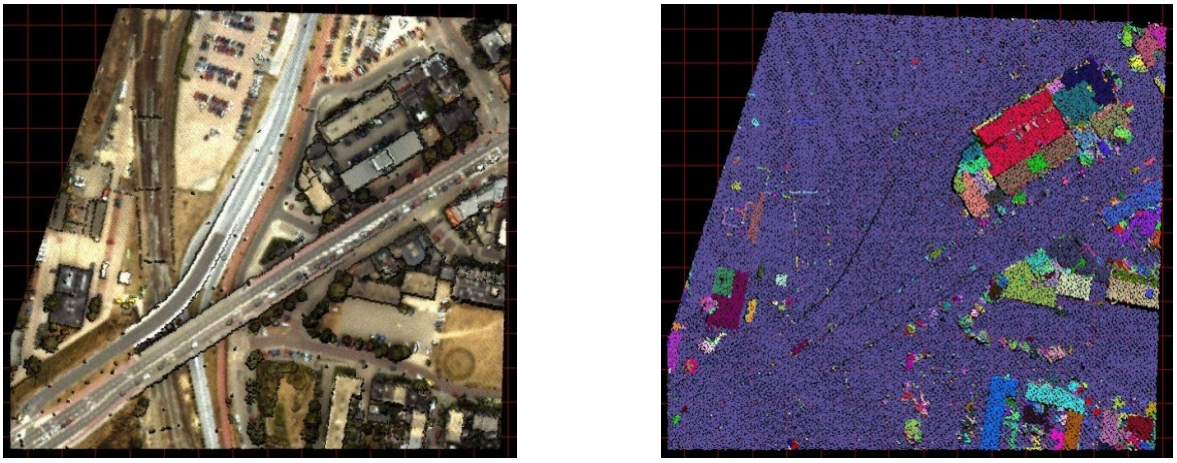
1. Introduction

The last two decades have seen an increase in the number of devices for the acquisition of 3D point clouds. In the same period the resolution, accuracy and acquisition rate of these devices have increased. This has led to a marked increase in the resolution, accuracy and size of point clouds that have to be processed. This trend is set to continue with devices becoming cheaper and more accessible to the general public. Presently the biggest challenge for processing point clouds is their size (typically numbering in the hundreds of millions of points). This challenge is overcome by a combination of data structuring, point reduction, segmentation and memory optimization strategies.

With 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. With segmentation strategies higher order features are extracted from a point cloud so that user interaction is with these features rather than the points themselves. Naturally fewer features than points leads to greater efficiency as fewer features have to be manipulated. 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. An example of segmentation is shown in figure 1. 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 (extraction of edges, surfaces, facets or objects) of the point cloud and the local variations in the point cloud. A robust segmentation algorithm has to be able to accommodate different applications and local variations.



(a) Point cloud (b) Segmentation to obtain objects

Figure 1. Example of the segmentation of an urban scene. An airborne point cloud (a) is segmented to obtain objects in the cloud (b).

The flexibility and efficiency of a segmentation algorithm is determined by 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 an extension of a segmentation algorithm developed by Sithole (2005). The original algorithm was only applied to airborne laser scanner data. The extension is necessary for the segmentation of terrestrial laser scanner data.

The paper is structured into four sections. Sections 1.1 and 1.2 present a review of current segmentation algorithms. In sections 2, 3 and 4 the proposed algorithm is discussed. Section 5 presents results from applications of the algorithm. In section 6 the paper is concluded.

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. A review of segmentation algorithms is provided by Hoover et al (1996), Wang and Shan (2009).

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

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 normals, 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 Gorte (2002), Lee et al (2002), Lukas et al, (1998) and Besyl and Jain (1988).

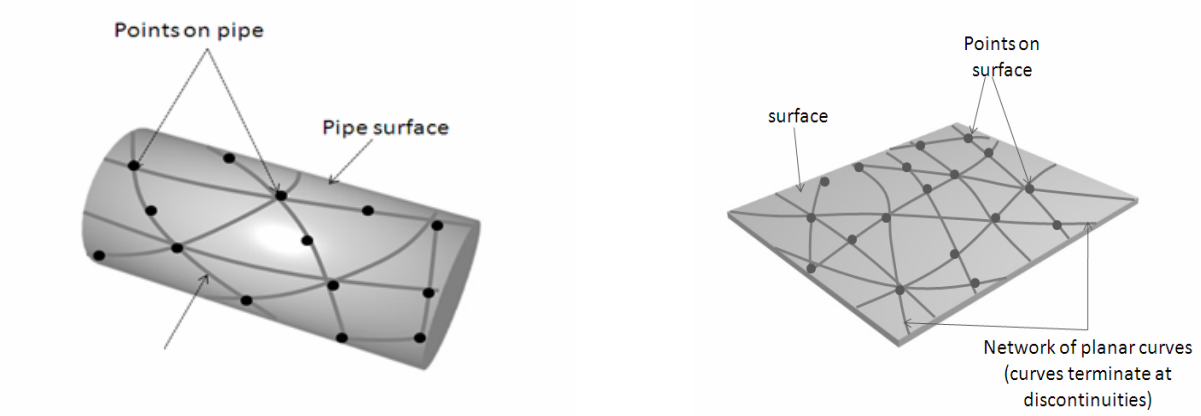
A less used algorithm is the scan line segmentation algorithm. Single raw scans obtained by devices such as terrestrial laser scanners retain their scan line geometry. Such point clouds are called range images or structured point clouds. Structured point clouds allow for a point in a scan line to be 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 et al (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. 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 include the size and organization of point clouds. 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 describe 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.



(a) (b)

Figure 2. Network of planar curves representing a surface

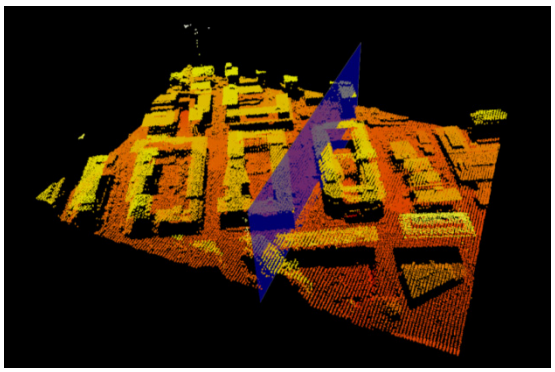
2. The Data Structure

A surface can be described by planar curves running across it. This is illustrated in figures 2a and b. Non parallel curves will intersect and the resulting intersection points can themselves be used to describe the surface. Reversing this scenario if the points on the surface are known then the curves

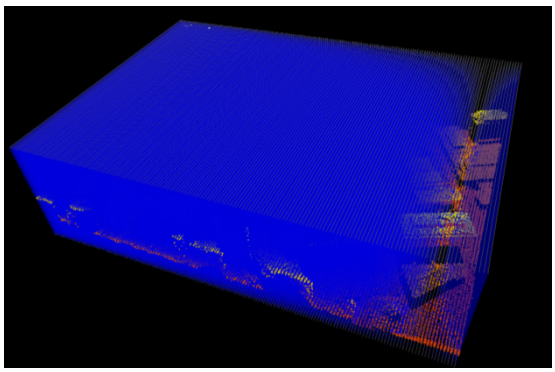
can be reconstructed and used to describe the surface. Additionally, if the surface contains discontinues, then these discontinuities can be determined from the terminators (points) at the end of the curves. The data structure devised by Sithole (2005) is based on this idea.

2.1 The Stack and Profiles

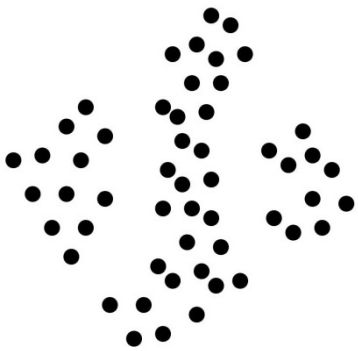
In unstructured point clouds, points are not arranged along planar curves. To overcome this problem Sithole (2005) sliced the point cloud into thin contiguous profiles. These profiles are collectively called the 'stack'. This is shown in figure 3.



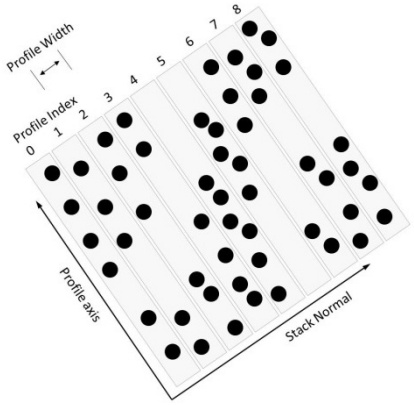
(a) A single profile in a point cloud



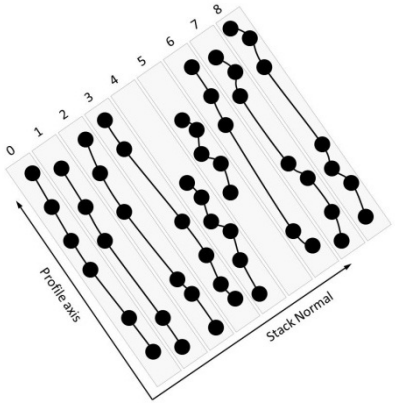
(b) Multiple contiguous profiles … the stack



(c) Point Cloud



(d) Stack 1 and its profiles



(e) Connect points

Figure 3. Profiles and stack. A 3D view of a single profile in a point cloud (a). Other continguous profiles are created to obtain the stack (b). A 2D conceptualisation is shown in (c) and (d). The points in each profile are connected (e).

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 to approximate the planar curves

within the profile. The stack can now be defined by a graph Gφ(V, Eφ) where V is the set of all the points in the point cloud and Eφ is the set of all edges in all the profiles in the stack. Naturally every profile in a stack is a sub graph of Gφ (V, Eφ). The profile graph is represented by Gφ,p(Vp, Eφ,p) where p is the index (p = 0, 1, 2, 3 … n-1, where n is the total number of profiles in a stack) of a profile in the stack, Vp is the set of points in a profile and Eφ,p is the set of edges in a profile after a profile segmentation. By necessity $Vp \in V$ and $E\varphi,p \in E\varphi$.

2.2 The Stacks

To complete the data structure other stacks are created along different stack normals. This is shown in figure 4. These stack normals are spread as far apart as possible within a hemisphere, figure 4(a). 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 Gφ 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 4(d).

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 Gφ(V, Eφ) . 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 Gφ,p(Vp, Eφ,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:

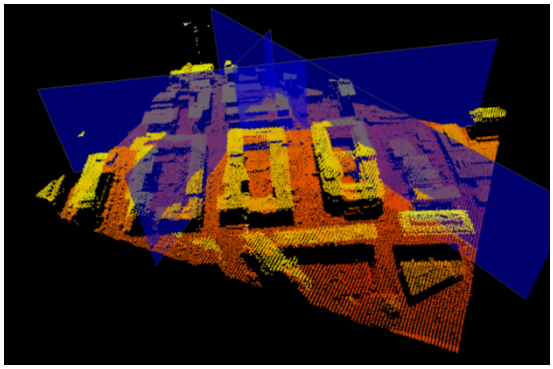
   $G\varphi,pi(Vpi, E\varphi pi) \cap G\varphi,pj(Vpj\ E\varphi,pj) = \emptyset$,

   where: pi and pj are the indices of profiles in the stack aligned along the vector φ.

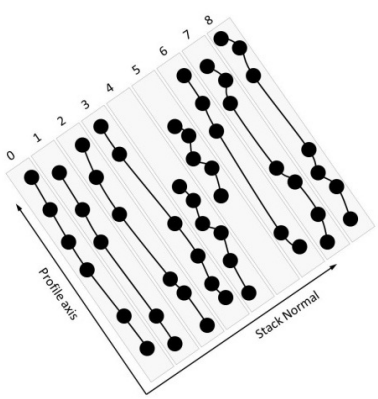
5. The intersection of profiles in two different stacks is a single point:

   $G\varphi m,pi(Vpi, E\varphi m,pi) \cap G\varphi n,pj(Vpj\ E\varphi n,pj) = v$,

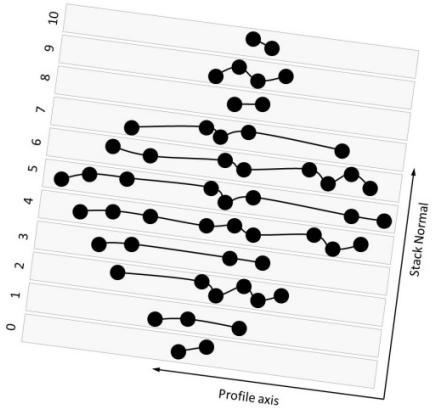
   where: φm and φn are the vector directions of two different stacks and v is a single point in V.



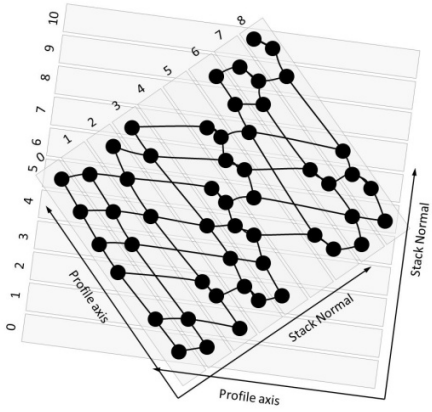
(a) Profiles from different stacks



(b) Stack 1



(c) Stack 2



(d) Overlay of stacks 1 and 2

Figure 4. Stacks and profiles defined on a point set. A 3D view of three stacks aligned in different directions (a). A 2D conceptualisation of two stacks, (b) and (c). The two stacks are overlayed resulting in a graph in which the two stacks are subgraphs.

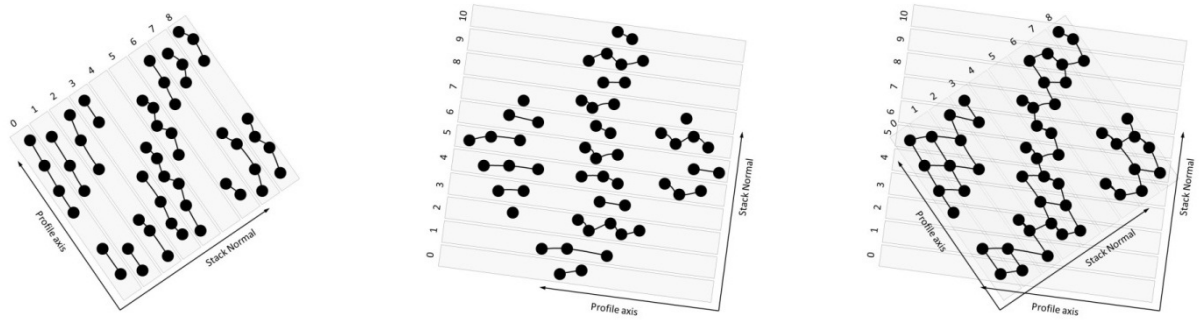
Together these properties make it possible to traverse the data structure or search point neighbourhoods in G(V, E). Importantly the data structure contains cross sectional views of the point cloud. These properties are now harnessed to segment the point cloud.

3. The Segmentation Algorithm

After creating the stacks the connected points in each profile 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 is discussed in section 4.1. After profile segmentation the overlay of the stacks now yields segments, or graphs Gs(Vs, Es) that are sub graphs of G(V, E). This is shown in figure 5. 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 the complexity of the segmentation problem is substantially reduced. This is the main advantage of the proposed segmentation approach.



(a) (b) (c)

Figure 5. Segmentation of the point cloud. Segmentation of profiles in Stack 1 (a), Segmentation of profiles in Stack 2 (b) and Overlay of stacks 1 and 2 (c).

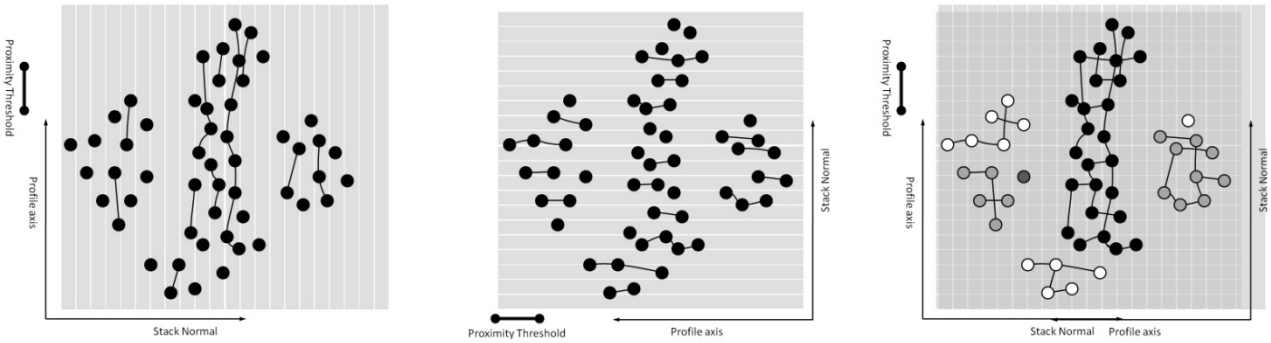
Another powerful feature of the data structure is that it allows for cross sectional analysis of the point cloud. The effect of this is that the segmentation can be tuned to yield edges, surfaces, facets or objects. A disadvantage of the data structure is that it maybe not be as compact or memory efficient 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 breadth first searches of point neighbourhoods are possible.

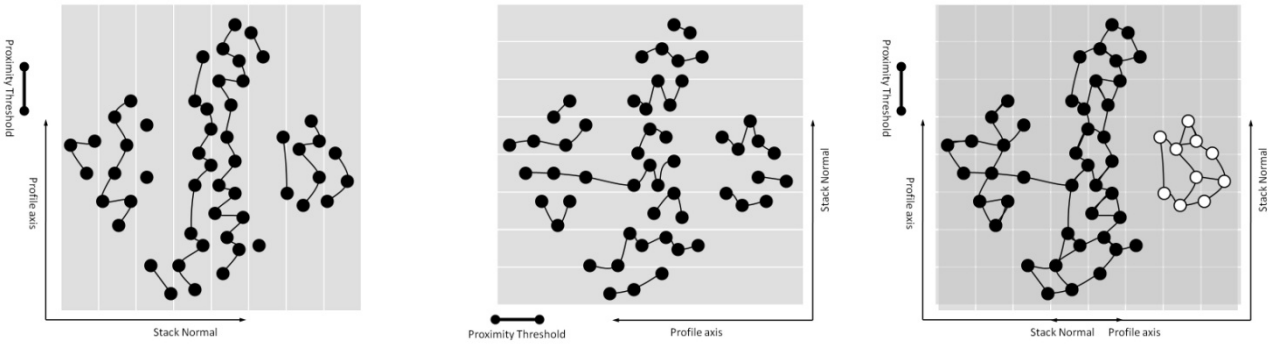
Space partitioning data structures such as kd-trees are defined by both rules and parameters. However, because of their hierarchical construction, an immediate traversal of point neighbourhoods is impossible. Discovering a point's neighbourhood always requires a traversal of the tree's hierarchy. The 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.



Stacks with a profile width 0.5 times that of the stacks in figure 5 Overlay of the stacks 1and 2

Result: Over segmentation



Stacks with a profile width 1.5 times that of the stacks in figure 5 Overlay of the stacks 1and 2

Result: Under segmentation

Figure 6. 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 5.

These two data structures demonstrate the trade-offs 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.

The space complexity of an unconstrained Delaunay triangulation is determined solely by the number of points in the cloud. In contrast the space complexity of the proposed data structure is determined by the number of stacks, the profile width and the method used in segmenting the profiles. From experiments it was found that when the number of stacks was set to three, the profile width set to the average point spacing and the profiles segmented by proximity the space complexity of the data structure approximated that of a Delaunay triangulation.

The time complexity of the data structure also depends on the number of stacks, the profile width and the method used to segment the profiles. The time complexity of creating the data structure hasn't been fully tested but it is expected that if the number of stacks is set to three and the profile width is set to the average point spacing that it will be equal to or lower than a surface determination using a Delaunay triangulation.

3.1.2 Profile Width versus 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 3, 4 and 5. However, very small and very large profile widths lead to over segmentations and under segmentations respectively. This is shown in figure 6. 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

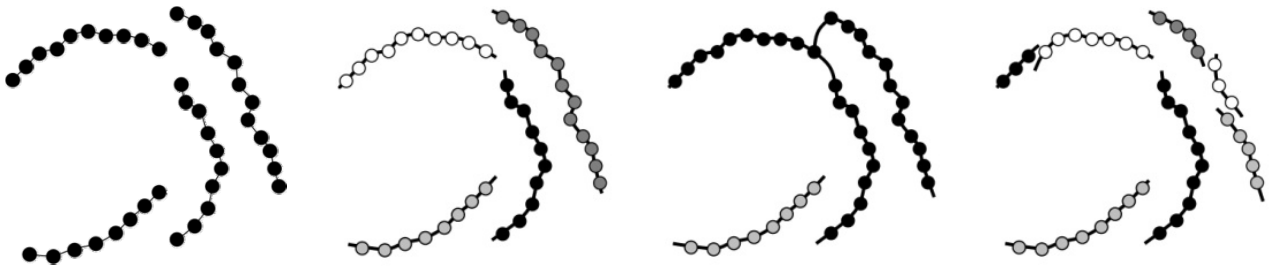
Sithole's (2005) implementation of the profile intersection algorithm allowed for only a distribution of the stack normals around a horizontal disk. This was because he was working with airborne lidar data and vertical profiles were ideal for the analysis of urban landscapes. The first contribution of this paper to Sithole's profile intersection algorithm is to spread the stack normals about a hemisphere; this is shown in figure 7. 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.

Original: Stacks distributed on a horizontally. Extension: Stacks distributed on a hemisphere.

Figure 7. Extension to the profile intersection algorithm. Instead of evenly spreading the stack normal about a semi-circle the stack normal are evenly distributed about a hemi-sphere.

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 8. 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.



(a) (b) (c) (d)

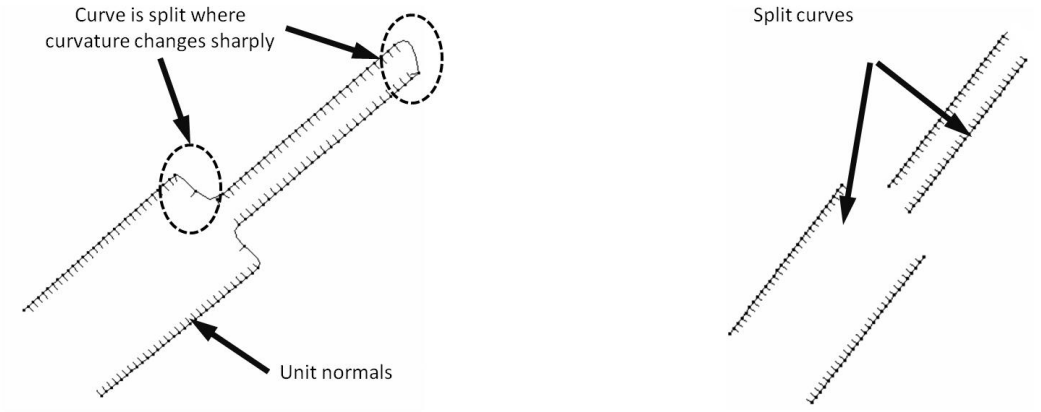
Figure 8. The choice of algorithm and parameters can yield a correct detection of the curves, a merging of the curves, or an over segmentation of the curves. (a) Points taken from a single profile in a stack. (b) ideal segmentation, (c) under segmentation, (d) over segmentation.

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.

4.1.2 Segmentation by Curve Fitting

Proximity segmentation will fail where curves are entangled. 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, is used to identify the model that best fits the points. 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 9 for 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 normal.



(a) Curve before splitting (b) Curve after splitting

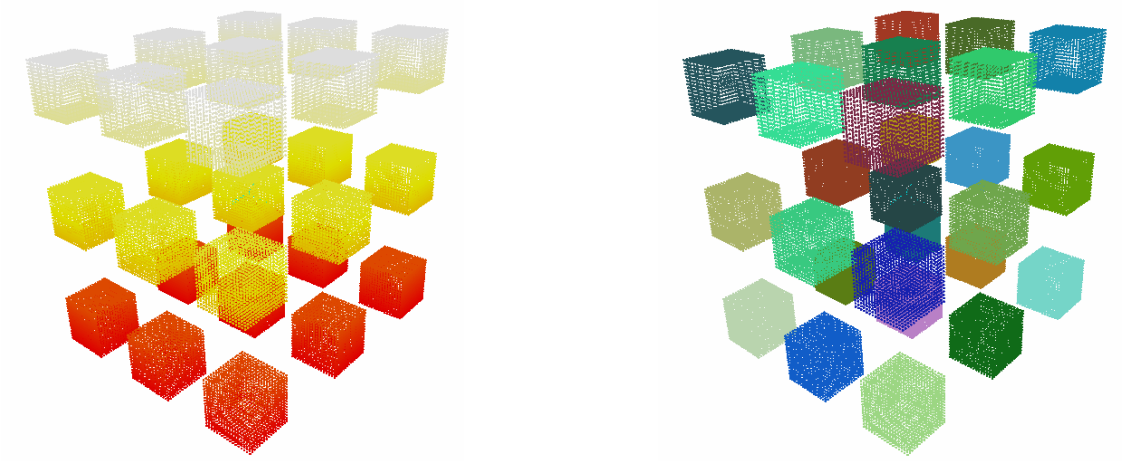
Figure 9. A single profile across a coaxial pipe. (a) The points in the profile represent four curves. (b) Discontinuities in the profile, are used to discriminate the four curves.

5. Results

To demonstrate the applicability of the concept, the segmentation algorithm was tested on various terrestrial lidar point clouds. The results of these tests are described below.

5.1 Sample 1: Cubes

This is a simulated point cloud of cubes. The purpose of this sample is to demonstrate that disjointed objects in a cloud can be separated. The results are shown in figure 10.

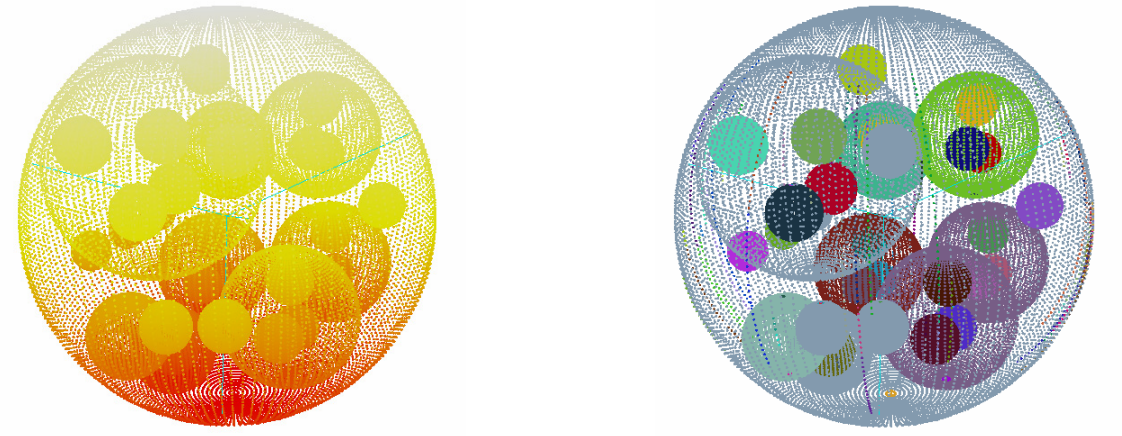


41500 points … … segmented in 9 seconds

Figure 10. A simulated point cloud (a) is segmented (b). In the segmentation the individual cubes are detected and coloured differently.

5.2 Sample 2: Spheres within Spheres

This is also a simulated point cloud of spheres contained within other spheres. The purpose of this test is to find if the algorithm is able to find objects within objects. This is shown in figure 11. The algorithm had little difficulty in doing this. Here the profiles were segmented by first fitting curves to profile points using the curve reconstruction algorithm proposed by Dey et al (2000) and then applying a proximity threshold on the curves connecting the profile points.

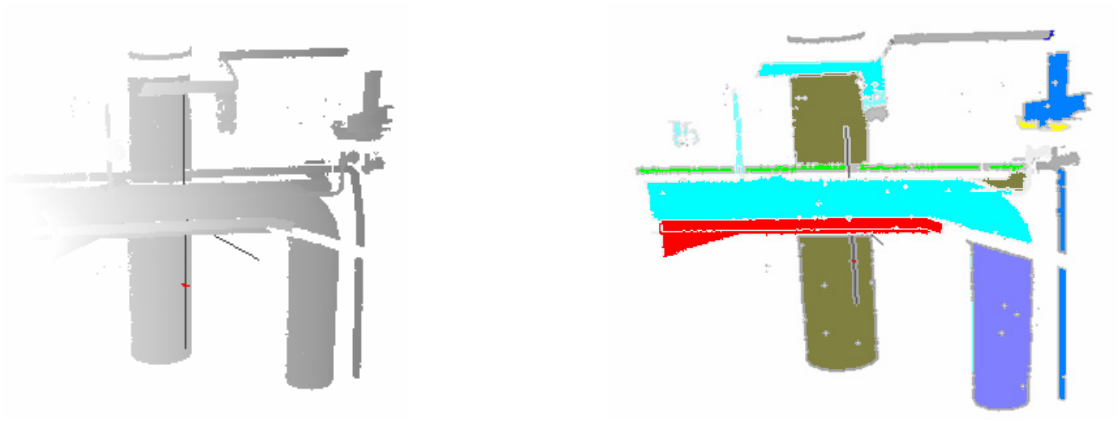


155000 points … … segmented in 35 seconds

Figure 11. A simulated point cloud (a) is segmented (b). In the segmentation the spheres of different sizes are detected and coloured differently.

5.2.1 Sample 3: Piping Installation

The segmentation is that of a section of a piping installation, figure 12. Contained in this point cloud are pipes of varying sizes. In this instance the segmentation yields objects (the individual pipes). From the data structure it is possible to obtain the axes and the deformations in the pipes, but this will not be elaborated here. The segmentation is successful and noticeably even the thin pipes are segmented.

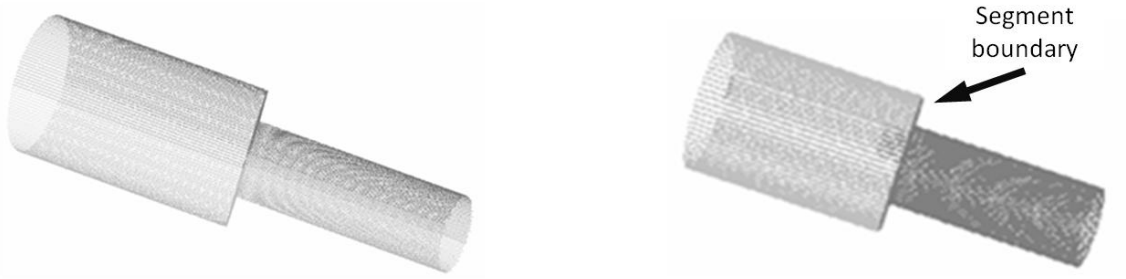


Point cloud Segmentation

Figure 12. Segmentation of a piping installation.

5.2.2 Sample 4: Coaxial Pipes

The point cloud (figure 13) is used to demonstrate the performance of the algorithm in the presence of surface discontinuities. Although the coaxial surfaces are connected, because of the curve fitting profile segmentation algorithm the discontinuity in the surfaces are detected.

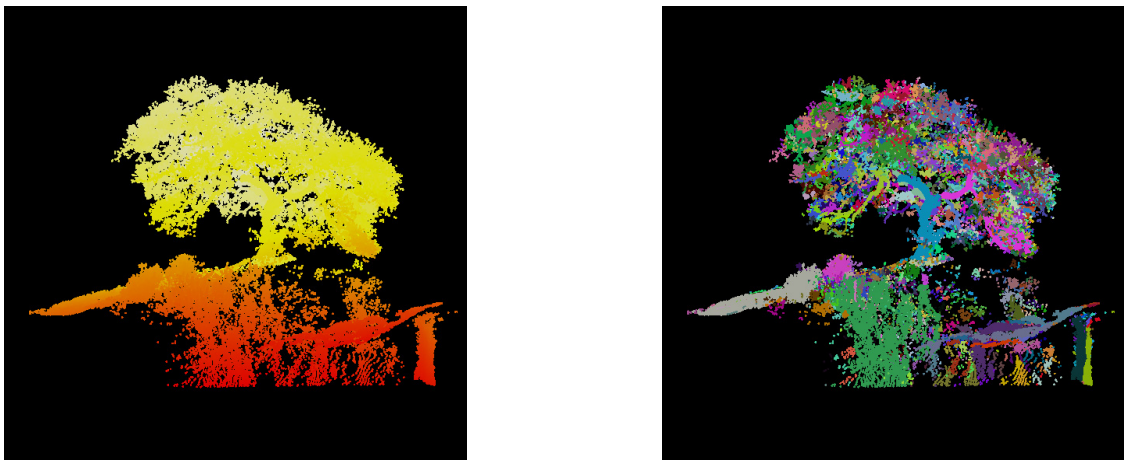


Point cloud Segmentation

Figure 13. Segmentation of a coaxial pipe.

5.2.3 Sample 4: Tree

The next sample point cloud was used to test the performance of the algorithm on point clouds of natural objects, in this case a tree, figure 14. Generally the point cloud is able to detect the trunk and branches of the tree. In general performance on natural object will require improvement.



Point cloud Segmentation

Figure 14. Segmentation of a tree. The trunk and larger branches are successfully detected.

5.3 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. In general the profile width is set to the average useable resolution of a single scan (before registration). Useable here meaning the portions of a scan that are used in modelling.

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.

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 advantage of the segmentation algorithm is that it is based on a conceptually simple data structure that can be imposed on any point cloud. Although the present implementation is not

parallel, the data structure is well suited for parallel computation. Furthermore it can simultaneously discriminate objects in a point cloud relatively fast. The disadvantage of the point cloud is that the data structure is not compact and requires greater management than other data structures (e.g., TINs). It can also become bloated if too many stacks are used.

The data structure and segmentation algorithm have a range of applications. Future work will explore performance, automatic parameter selection, optimisation and parallel computation, and identification of applications. Potential applications include edge detection, deformation and object classification.

7. Acknowledgements

The authors would like to acknowledge that this paper is based on a paper presented at AfricaGeo 2011 (Sithole and Mapurisa, 2011).

8. References

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

T. K. Dey, K. Mehlhorn and E. Ramos, 1999 Curve reconstruction: connecting dots with good reason. Comput. Geom. Theory & Appl., Vol. 15 (2000), 229-244. Also in Proc. 15th. Sympos. Computational Geometry 1999, pp. 197-206.

Gorte, Ben, 2002. Segmentation of TIN-structured laser altimetry points clouds, Symposium on Geospatial Theory, Processing and Application, 5 pages.

Hoover, A., G. Jean-Baptiste, X. Jiang, P.J. Flynn, H. Bunke, D.B. Goldgof, K. Bowyer, D.W. Eggert, A. Fitzgibbon, and R.B. Fisher, 1996. An experimental comparison of range image segmentation algorithms, IEEE Transactions on Pattern Analysis and Machine Intelligence, 18, 673 – 689.

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.

Lee, Impyeong and Toni Schenk, 2002. Perceptual organization of 3D surface points, International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, 34, 193-198.

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.

Sithole, G., Mapurisa, W., 2011: '3D Object Segmentation of point clouds using profiling techniques'. AfricaGeo Conference 2011. Cape Town, 31 May – 2 June, 2011. Pages16.

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.

Wang, J. & Shan, J., 2009. Segmentation of lidar point clouds for building extraction. In Proceedings American Society of Photogramm Remote Sensing Annual Conference. p. 9–13.

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.