MOVING WINDOW SEGMENTATION FRAMEWORK FOR POINT CLOUDS

George Sithole[a], Ben Gorte[b]

[a] Geomatics Division, School of Architecture, Planning and Geomatics, University of Cape Town, Private Bag X3, Rondebosch, 7701, South Africa, Email: george.sithole@uct.ac.za
[b] Optical and Laser Remote Sensing, Delft University of Technology,
P.O.Box 5058, 2600GB Delft, the Netherlands, email: b.g.h.gorte@tudelft.nl

KEYWORDS: Algorithm, Segmentation, Lidar, Point Cloud

ABSTRACT:

As lidar point clouds become larger streamed processing becomes more attractive. This paper presents a framework for the streamed segmentation of point clouds with the intention of segmenting unstructured point clouds in real-time. The framework is composed of two main components. The first component segments points within a window shifting over the point cloud. The second component stitches the segments within the windows together. In this fashion a point cloud can be streamed through these two components in sequence, thus producing a segmentation. The algorithm has been tested on airborne lidar point cloud and some results of the performance of the framework are presented.

1 INTRODUCTION

1.1 Background

The segmentation of lidar data has traditionally been a post processing operation typically effected on an entire lidar dataset. But recent advances in lidar technology now demand streamed methods of segmentation. Two advances in particular necessitate this. Firstly, scan rates have increased considerably. Secondly, mobile scanners are typically mounted on vehicles, but will soon also be portable and this opens scanning to a wide range of real-time applications. The increased volume of scans and the future requirement for real-time segmentation will mean that static segmentations will become complex, inefficient and inadequate. This will require streamed approaches to segmentation.

1.2 Previous Work

There is an extensive of body of work on the segmentation of range data and unstructured point clouds. A general review of segmentation algorithms is provided by Hoover et al (1996) and Wang and Shan (2009). In a static segmentation segments are mostly aggregated by means of clustering, region growing and merging/dividing, or connected components frameworks. Most segmentation algorithms use one or a combination of these frameworks.

An example of a clustering segmentation approach is presented by Filin (2002). Region growing algorithms achieve segmentation by examining point neighborhoods about selected seed points. Segments are built radiating outward from these seed points until the entire point cloud is segmented. Proximal segmentation algorithms define distance measures between points in a cloud. Points that are within a given distance of each other are joined by edges to form a disconnected graph. A connected component analysis of the disconnected graph yields segments (the sub graphs).
Streamed segmentation frame works build on the above segmentation algorithms with the objective of (1) partially loading the point cloud into main memory, i.e., the segmentation is done in chunks (2) removing from main memory those chunks that are no longer needed and make room for succeeding chunks. When a chunk is removed from memory it is written to a segmentation file. This framework minimizes the memory footprint and allows the point cloud to be streamed into a segmentation processor.

1.3 Current approaches

The study of streamed processing of point clouds has been ongoing with applications devoted to meshing and surface generation. Examples of such research are presented by Bolitho et al (2007) and Isenburg and Lindstrom (2005). Many of these applications are designed to be off-line and require a restructuring of the point cloud file. The approaches below consider situations where the processing has to be real-time and a reorganization of the point-cloud file is not possible.

Most of streamed segmentation approaches are designed for range images. Klasing et al 2009 present a streamed segmentation approach in which they search for the nearest points in a sequence of range images. Another example is present by Heisele and Ritter (1999). Wang and Tseng (2011) present a volumetric streamed segmentation for unstructured point clouds. In their approach they voxelize the point cloud. For real time segmentation this presents two overheads. Firstly, voxelization requires the extent of the point cloud to be determined. Secondly the increments to the point cloud have to be voxelized. These streamed techniques are designed to take advantage of the structure of the data or create a data structure that facilitates the streamed segmentation. This complicates the application of these algorithms.

1.4 Objectives

This paper proposes a streamed segmentation framework designed to work on unstructured point clouds. The framework is based on a moving windows strategy. The

objective of the framework is to stream existing segmentation approaches. The paper is divided into four sections. Section two presents the proposed streamed segmentation framework. In section three the results of experiments are presented. Section four discusses the results of the experiments. Finally in section five the paper is concluded.

2 METHOD

2.1 Algorithm

The framework of the algorithm is shown in Figure 1. The algorithm is composed of three components arranged in the following sequence:

1. **Segmentor**: This component segments each span of the point cloud.

2. **Weaver**: This component aggregates current segment labels with previous segment labels.

3. **Extractor**: This component extracts objects from the streamed segments.

This paper only discusses the Segmentor and Weaver components.

2.1.1 Segmentor

The point cloud is partitioned into contiguous spans. The first three spans (Window 1) are loaded into memory and segmented, Figure 1 (b). The segmentation algorithm used to segment the spans is called the segmentor. Next the first two spans are unloaded from memory but the segment labels together with the point indexes are retained. After this the fourth and fifth spans are loaded into memory and segmented together with the third span (Window 2). Note here that only three spans are loaded into memory at any one moment. The segmentation algorithm used in each window depends on the application. In Figure 1 segmentation is done by proximity but it can easily be another algorithm.

2.1.2 Weaver

This component is the core of the proposed moving window segmentation algorithm. Because the segments from windows 1 and 2 overlap the segment labels from the first window can be simply carried over to the overlapping segments in the second window. The weaver algorithm aggregates segments in a fashion similar to region growing and connected components. However only a partial graph is built and segment labels are unloaded from memory when they are no longer needed.

Additionally the connected components analysis is done on the fly. To begin two lookup tables (LUT) are created. The first LUT is made of single key-value pairs and provides the segment label (value) of every point (key) in the cloud. The second LUT is made of multi key-value pairs and provides the segment label (value) for the segments in the current window against the segment labels (key) of overlapping segments in the previous window. The lookup tables are populated as in Figure *1*.

The First Segmentation

In the first segmentation all segments are labeled sequentially starting from 1. The segment labels are transferred to the points. The point-segment label pairs are added to the LUT. All segments labels are added to the Segment-Segment LUT, but because there are no overlapping segments to the right the labels are paired with a value of -1. The value of -1 is used to indicate non-existent data.

The Second Segmentation

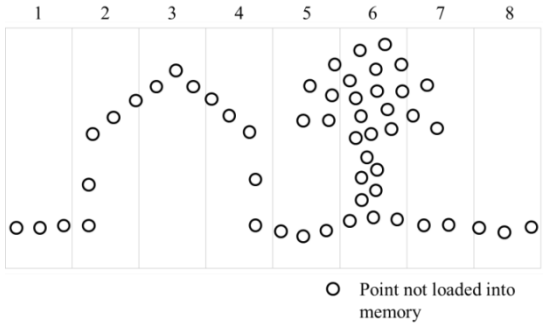
All new segments are labeled sequentially starting from the last segment label. Each point in the window is looked up in the Point-Segment LUT. If the point exists in the LUT nothing further is done for the Point-Segment LUT. However in the Segment-Segment LUT the key-value pairs are updated for those segments that overlap, i.e., now have a segment on the right side. For example in the table in Figure *1* (c) segment 1 now has segment 4 to its right. If the point doesn't exist in the LUT then the point-segment label pair is added to the LUT. All new segments labels are added to the Segment-Segment LUT, but because there are no overlapping segments to the right the labels are paired with a value of -1

Succeeding Segmentations

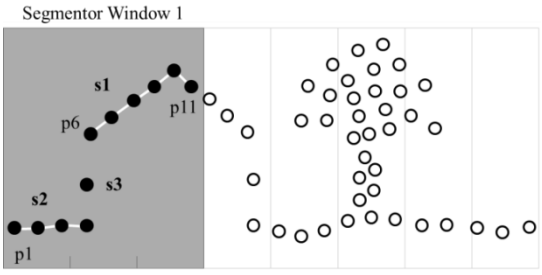
All points not inside the current or previous window are said to be out of scope. Points that are out of scope are unloaded from memory and removed from the Point-Segment LUT. This is shown in Figure 1(d). Segments go out of scope when none of their points lie inside the current or previous window. The entries in the Point-Segment and the Segment-Segment LUT for the points and segments that go out of scope are written to a Point-Segment and Segment-Segment file respectively. Unloading points and segments that are out of scope ensure that the footprint of the algorithm in memory is as small as possible and that the lookups in the LUTs are as fast as possible. Moreover it is consistent with the objective of developing a streamed segmentation. As with the first and second segmentation all new segments are labeled sequentially starting from the last segment label. The aggregation procedure now proceeds as in the second segmentation.

The Last Segmentation

The last segmentation proceeds as in the preceding segmentations. However at the end all remaining points and segment labels are written to the Point-Segment and Segment-Segment files. After this the points and segments are unloaded from memory.



(a)



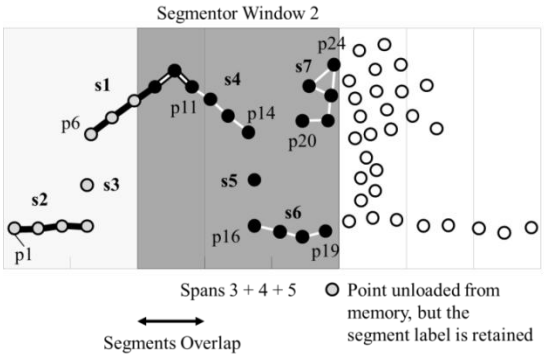
Point-Segment LUT

Point	Seg ID
1	2
2	2
:	:
11	1

Seg-Seg LUT

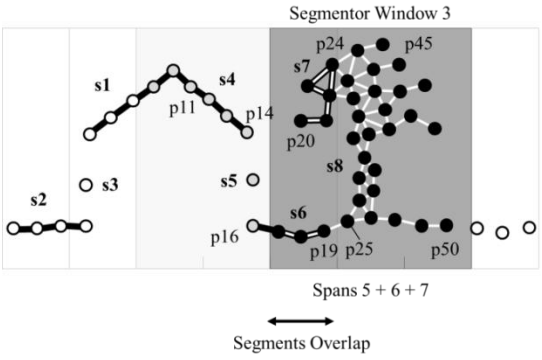
Seg ID L	Seg ID R
1	-1
2	-1
3	-1

(b)



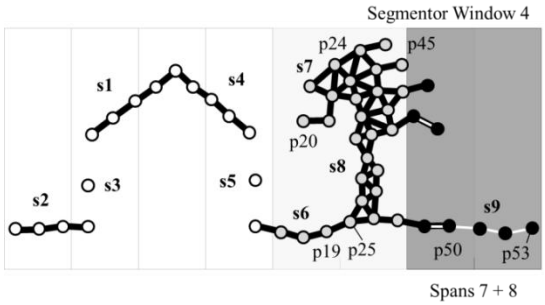
Point	Seg ID	Seg ID L	Seg ID R
6	1	1	4
:	:	4	-1
11	1	5	-1
12	4	6	-1
:	:	7	-1
16	6		
:	:		
24	7		

(c)



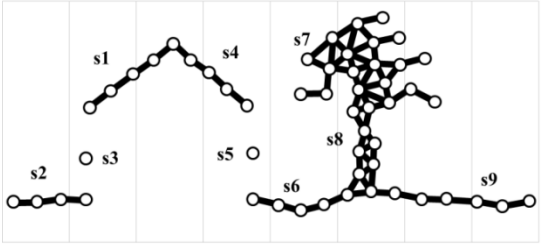
Point	Seg ID	Seg ID L	Seg ID R
16	6	6	8
:	:	7	8
19	6	8	-1
:	:		
24	7		
25	8		
:	:		
45	8		

(d)



Point	Seg ID	Seg ID L	Seg ID R
45	8	8	9
:	8	9	-1
50	8		
51	9		
52	9		
53	9		

(e)



(f)

Figure 1: The moving window segmentation of a spanned point cloud (a). Points in the first three spans (the segmentor window) are selected and segmented (b). The segmentor window is moved forward by two spans. There is an overlap of one span between

the current and previous segmentor window, therefore the segments overlap. Because of the segment overlaps the segment labels can now be carried over from the previous window into the current window (c). The process is repeated in steps (d) and (e) until the entire point cloud has been segmented (f).

2.2 Discussion

2.2.1 Advantages

General

The algorithm can be applied to any point cloud. Also as points are streamed to the algorithm knowledge of the geometric characteristics of the point cloud are not required.

Multi-purpose

The framework allows any existing segmentation algorithm to be used.

Memoryless

The segmentation only depends on adjacent windows with no knowledge required of the windows that preceded them. This allows for point clouds to be streamed through the segmentor and weaver, thus making the algorithm applicable for real-time applications such as mobile mapping.

Flexible

The size, number and overlap of spans/windows can be adapted to the geometric characteristics of the point cloud (e.g. point spacing). For the ease of explanation, the demonstration example in Figure 1 uses spans of equal size, three spans per window and windows overlap by one span. These parameters can all be changed.

2.2.2 Disadvantages

Memoryless

The order in which spans are streamed to the framework has a strong influence on the quality of the segmentation. If spans are not fed sequentially then there will be an over segmentation.

Self-Intersection

If a stream loops on itself (in space) the framework does not account for previous segmentations in the overlap. Therefore the same segment will be detected multiple times in the same location. This limits the application of the framework where a full segmentation is required. A solution to this is being investigated.

2.2.3 Segmentation History

The final segmentation is contained in the Point-Segment and Segment-Segment files. Essentially these files contain two graphs that describe the relationship between points and segments, and overlapping segments. The files provide a history of the segmentation. However, if a history of the segmentation is not desired, then the segment labels can be easily carried over on the fly where segments overlap. Naturally the Segment-Segment LUT will now always contain -1 on the right hand side.

3 EXPERIMENTS AND RESULTS

3.1 Data source

The data source used in the test is of a residential district of Stuttgart, see Figure 2. The data is a small subset (190000 points) of the OEEPE project on laser scanning (Petzold and Axelsson, 2001). The point cloud has a point spacing of about 1.0 m. The landscape is not particularly difficult to segment but for the purpose of the test it was considered adequate.

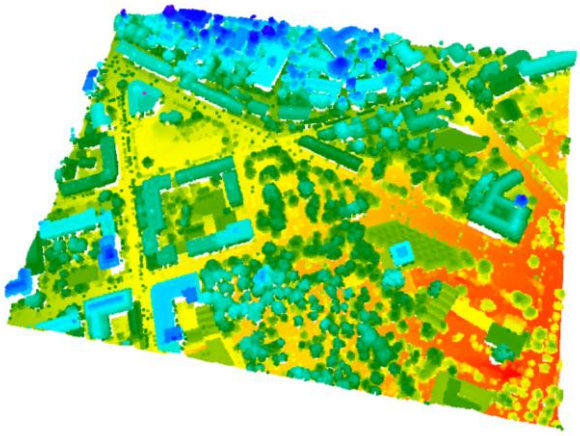


Figure 2 Test data used in the experiments

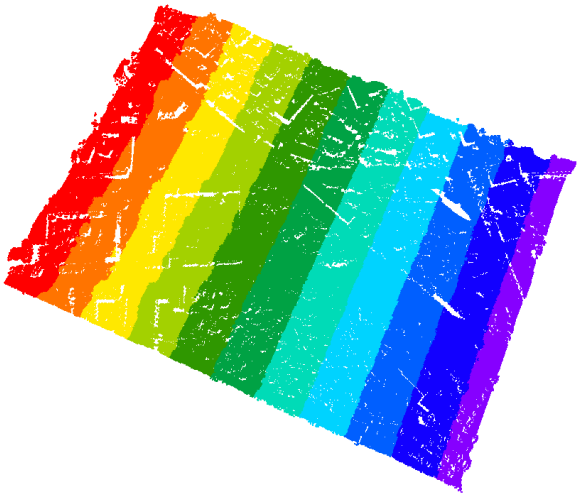


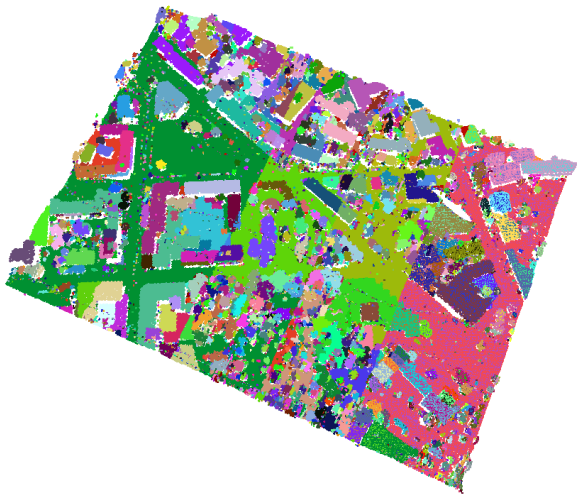
Figure 3 The spanned point cloud. Here for visual clarity spans of 50 m are used. Ideally smaller spans are desirable. The spans are loaded into memory when needed.

3.2 Segmentation

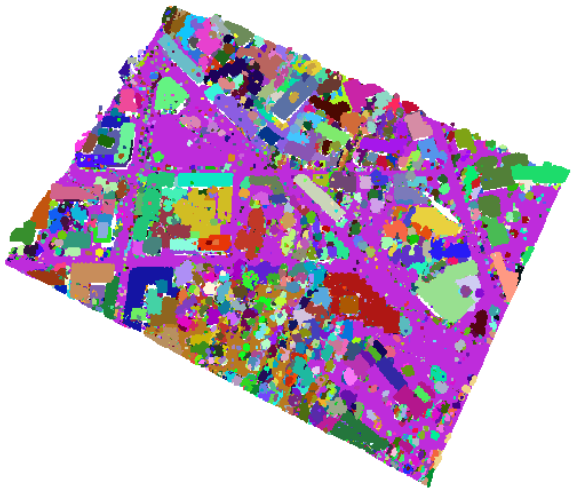
The method of segmentation used in the segmentor is proximity segmentation. A Delaunay triangulation was used to generate the edges between the points. The proximity threshold was set at 2.5 m. Proximity segmentations are not robust, but again for the purposes of this test they were considered sufficient.

3.3 Demonstration

Figure 3 and Figure *4* show how the algorithm works on a real point cloud. The point cloud is spanned and then groups of adjacent spans (the segmentor window) are segmented. The window is then repeatedly moved and segmented until the entire point cloud is segmented.
A visual comparison of the moving window segmentation against a static segmentation showed visually no difference. This is mostly due to the proximity segmentation which is expected to always perform the same along span edges.



(a)



(b)

Figure 4 The segmentation of each segmentor window. Unmerged segments (a) and merged segments (b).

3.4 Performance

It's desired to know how the moving window segmentation performs at different span widths. This is shown in Figure 5. As expected the static segmentation is faster than the moving window segmentation. At the lower smaller span widths the triangulation is faster because of the fewer in-circle tests. As the span width is increased this advantage is lost. Further increases in the span width see the processing time decreasing. This is because the number of triangulations are becoming fewer and offsetting the cost of in-circle tests.

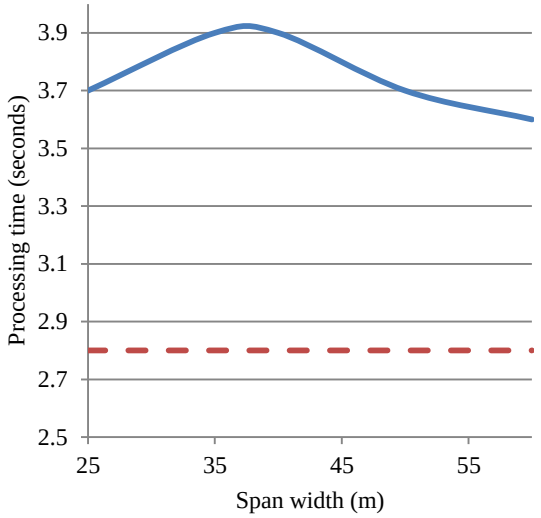


Figure 5 Span width versus processing time. The solid line show the processing times for segmentation with varying span width. The dotted line shows the processing time for a static segmentation.

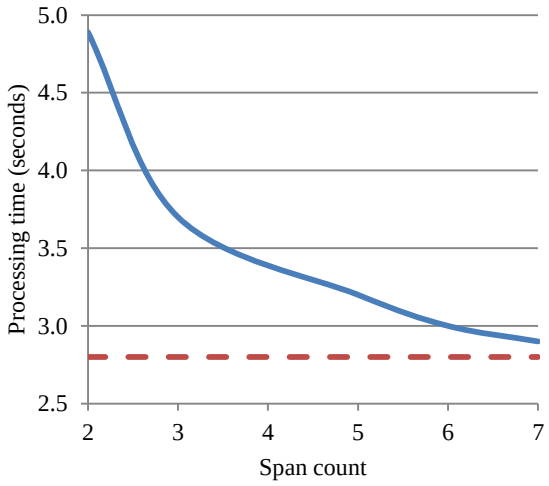


Figure 6 Count of spans in a segmentor window versus processing time. The solid line shows the processing times for segmentation with varying span counts. The dotted line shows the processing time for a static segmentation.

The maximum represents the point at which the balance of the width of spans versus the number of spans is most unfavorable. Further tests on are needed to confirm this.

In the above tests only three spans were used in a segmentor window. Figure 6 shows the effect of varying the number of spans in a segmentor window. The fewer spans in the segmentor window the greater the processing time. Again this is because of the in-circle tests in the triangulation. As the number of spans increase the triangulations become fewer and the cost of in-circle tests is reduced.

Isotropy

From the explanation provided in section 2.1 it may be concluded that the framework is isotropic. This is only because a proximity segmentation algorithm has been used to explain the segmentor. Proximity segmentation algorithms are generally isotropic particularly if the tested proximal neighborhood is small. Segmentation algorithms such as scan line algorithms are strongly direction dependent. Because of this the framework is not guaranteed to be isotropic.

Edge effects

As can be seen in Figure 1 the success of the framework depends on consistent segment generation in the window overlaps. Segmentation algorithms that are based on surface analysis, such as region growing algorithms, may yield different segmentations at the edges of the windows. Because of this consistent segments are not guaranteed in the window overlaps. The framework may behave unpredictably in such situations. The solution to this problem is being investigated.

4 DISCUSSION

The test in Figure 5 and Figure 6 suggest that there exists a relationship between processing time, the width of spans, the number of spans in the point cloud and the number of spans in a segmentor window. Determining this relationship requires further testing. From the tests it can be said that where resources are available static segmentation should be used. The moving window segmentation should be used when resources (e.g., memory) are insufficient or where a real time application is needed.

5 CONCLUSION

A frame work has been presented for the streamed segmentation of point clouds with the intention of segmenting unstructured 3D point clouds in real-time. The frame work has been tested on an airborne lidar point cloud. However the framework should work with any 3D point cloud as long as overlaps between segmentor windows can be established in the point cloud. This means that the framework is ideal for use with point clouds produced by 2D mobile scanners.

Presently the performance of the framework is being studied more thoroughly, particularly to understand the stability of the framework in the overlaps of windows. Also being studied is the extension of the frame work to 2D and 3D, with the intention of serving off-line segmentation of very large point clouds.

A future publication is also planned to present the extractor component.

6 REFERENCES

Bolitho, M., Kazhdan, M., Burns, R., Hoppe, H., 2007. Multilevel streaming for out-of-core surface reconstruction. Proceedings of the fifth Eurographics Symposium on Geometry Processing, Aire-la-Ville, Switzerland, 2007. pp. 69 -78.

Filin, S., 2002. Surface clustering from airborne laser scanning data, The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. XXXIV, 3A, Graz, pp 117-124.

Heisele, B., Ritter, W., 1999, Segmentation of range and intensity image sequences by clustering. Proceedings 1999 International Conference on Information Intelligence and Systems

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, pp 673 – 689

Isenburg, M., Lindstrom, P., 2005. Streaming meshes. Proceedings of IEEE Visualization 2005, Minneapolis, U.S.A.. pp 231-238.

Klasing, K., Wollherr, D., and Buss, M., 2009, Realtime segmentation of range data using continuous nearest neighbors. Technische Universität München in Proc. ICRA, 6 pages

Petzold, B., Axelsson, P., 2001. Results of the OEEPE working group on laser data acquisition. Proceedings of OEEPE workshop on airborne laser scanning and interferometric SAR for detailed digital elevation models. March 1-3, Stockholm, Sweden. Official Publication No. 40. CD-ROM., 6 pages.

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

Wang, M., Tseng, Y., 2011. The Photogrammetric Record 26(133): pp 32–57