PROCEEDINGS, IQMULUS 1ST WORKSHOP ON PROCESSING LARGE GEOSPATIAL DATA

Deliverable D8.6.1

Circulation:	PU - Public
Lead partner:	TU Delft
Contributing partners:	CNR-IMATI-GE, UCL, SINTEF
Authors:	Roderik Lindenbergh (TUDelft) , Michela Spagnuologo (Imati), Jan Boehm (UCL), Ewald Quak (Sintef)
Quality Controller:	Tor Dokken (Sintef)
Version:	1.0
Date:	17.10.2014

Consisting of

SINTEF	STIFTELSEN SINTEF, Department of Applied Mathematics, Oslo, Norway
Fraunhofer	Fraunhofer Institute for Computer Graphics Research, Darmstadt, Germany
CNR-IMATI-GE	Institute for Applied Mathematics and Information Technologies of the National Research Council (CNR-IMATI), Genova, Italy
MOSS	M.O.S.S. Computer Grafik Systeme GmbH (MOSS), Munich, Germany
HRW	HR Wallingford Ltd (HRW), Wallingford, UK
FOMI	Hungarian National Mapping and Cadastral Agency (FÖMI), Institute of Geodesy, Cartography and Remote Sensing, Budapest, Hungary
UCL	University College London (UCL), Research centre for Photogrammetry, 3D Imaging and Metrology, London, UK
TUDelft	Delft University of Technology (TUDelft), Department of Geoscience and Remote Sensing & Man-Machine Interaction Group, Delft, The Netherlands
IGN	Institut National de l'Information Géographique et Forestière (IGN), Paris, France
UBO	Université de Bretagne Occidentale (UBO), European Institute for Marine Studies, Brest, France
Ifremer	L'Institut Francais de Recherche pour l´Exploitation de la Mer (Ifremer), Brest, France
Liguria	Regione Liguria, Genova, Italy

DOCUMENT HISTORY

Version[1]	Issue Date	Stage	Content and Changes
1.0	Oct 27 2014		Final

[1] Integers correspond to submitted versions

EXECUTIVE SUMMARY

New emerging data acquisition techniques provide fast and efficient means for multidimensional spatial data collection. Systems and techniques like airborne LIDAR, SAR satellites, stereo-photogrammetryand mobile mapping are increasingly used for digitally sampling ourenvironment. All these systems provide point clouds and images, often enriched with other sensor data, yielding high volumes of raw data. The IQmulus workshop was organized to stimulate researchers from different fields such as Computer Vision, Computer Graphics, Geomatics, Remote Sensing, working on the common goal of processing 3D data, to present state-of-the-art work in the field. The workshop took place on July 8th, 2014 in Cardiff (UK), in conjunction with SGP'14. The Symposium on Geometry Processing 2014, a Eurographics symposium in cooperation with ACM SIGGRAPH, is one of the premier venues for the geometry processing community.

The IQmulus workshop consisted of two keynotes, three presentations discussing a so-called IQmulus contest track and three technical sessions, containing a total of ten technical presentations that were selected based on a submitted extended abstract. The two keynotes presented possible solutions on how to store and respectively process big spatial data. Peter van Oosterom, from Delft University of Technology, gave a keynote presentation entitled *Point Cloud Data Manangement*, discussing methodology for storing and querying a huge airborne laser data archive. In his keynote, Mohamed Mokbel, from the University of Minnesota gave an overview entitled *SpatialHadoop: A MapReduce Framework for Spatial Data*, of his solution to process big spatial data by explicitly using the spatial locations of the data.

Within the IQmulus contest, research groups organize tracks in which different methods are used to solve a given problem on a given data set on a given computer. As also ground truth is available to the organizing group, a fair comparison of the performance of the different methods can be given. At the workhop, three such tracks were presented, one on the urban modelling of a high density airborne LIDAR data set sampling Dublin, one on the classification of a Laser Mobile Mapping Data set sampling Paris, and a third one on the approximation of rainfal data over Liguria.

What is notably presented in this proceedings are the accepted extended abstracts as submitted to the workshop. This is not the final word on this workshop however. On behalve of the workshop, a special issue on Processing of Large Geospatial Data for Elseviers journal of *Computers & Graphics* is organized. At the time of this writing, the review of the submitted manuscripts is still ongoing however.

October 2014

Genua, Michela Spagnuolo

London, Jan Böhm

Delft, Roderik Lindenbergh

FOREWORD

IQmulus is a 4-year Integrating Project (IP) partially funded by the European Commission under the Grant Agreement FP7-ICT-2011-318787. It is positioned in the area of Intelligent Information Management within the ICT 2011.4.4 Challenge 4: Technologies for Digital Content and Languages. IQmulus started on November 1, 2012, and will finish on October 31, 2016.

The goal of IQmulus, which stands for A High-volume Fusion and Analysis Platform for Geospatial Point Clouds, Coverages and Volumetric Data Sets, is to develop a platform that provides the needed functionalities to integrate latest research results in data processing and visualization to tackle important real-life challenges in geospatial applications. Given the wide choice of different available sensors and the massive amounts of data thus obtained, combined with the intent to provide useful knowledge in an appropriate period of time, the platform thus has to be scalable in processing and storage, and capable of handling the four aspects of variety, volume, velocity and analytics that are commonly associated with the term Big Data.

The IQmulus consortium is made up of 12 partners from 7 European countries, representing university teams for basic research in geo-spatial information processing, applied research institutes, an SME from the GIS industry as well as national and regional organizations such as mapping agencies. You are cordially invited to find out more about the project at its website www.iqmulus.eu.

As part of the R&D efforts concerning the IQmulus platform, the consortium members are strongly committed to the dissemination of the project results through publications and presentations. This includes the organization of an annual scientific IQmulus Workshop, which incorporates as one specific session the presentation of the results of the annual IQmulus Processing Contest. The contest is meant to provide task-specific benchmarking of software which is of interest for the application domain and which has been developed by the scientific community, and not only by the project.

In the following, you will find the proceedings of the first such workshop, including also contributions related to the processing contests. Please note that the workshop has led to a special issue on Processing of Large Geospatial Data of the journal Computers & Graphics, which is currently in preparation.

Finally, as a member of the coordinating team of IQmulus, I take this opportunity to thank all participants for their valuable contributions, and especially the three workshop organizers for IQmulus, namely M. Spagnuolo, J. Boehm and R. Lindenbergh, for making this event and the upcoming special issue happen.

Special thanks also go to the two organizers of the Eurographics Symposium on Geometry Processing 2014, Y.-K. Lai and R.R. Martin, for their cooperation in synchronizing the IQmulus Workshop with SGP 2014 and its graduate school and the practical arrangements that this format necessitated. Last but not least, our thanks go to Prof Joaquim Jorge, the Editor-in-Chief of Computers & Graphics who has approved the special session on Processing of Large Geospatial Data.

Oslo, Ewald Quak

Contents

Recent Advances on LoD for Procedural Urban Models 1
Gonzalo Besuievsky and Gustavo Patow
1 Introduction 1
2 Level-of-Details Analysis 2
3 LoD in Procedural Modeling 2
3.1 LoD by Configurable Rules 2
3.2 Automatic LoD 4
4 Challenges and Future Work 7
References 8

Locally refined spline surfaces for shape representation, and feature detection in large GIS data sets 11
Oliver Barrowclough, Tor Dokken and Vibeke Skytt
1 Introduction 11
2 LR B-spline surface representations 11
3 Approximation methods for LR B-spline surfaces 14
3.1 Least squares approximation 14
3.2 Locally refined multilevel B-spline approximation (LR-MBA) . . 15
3.3 Properties 15
4 Examples for GIS data sets 15
5 Detection of features in point sets 16
References 17

Automatic buildig reconstruction from digital surface models using 3D active shape models 19
Beril Sirmacek and Roderik Lindenbergh
1 Introduction 19
2 Method and initial results 20
2.1 Extracting approximate building footprint segments 20
2.2 Assigning seed points 20
2.3 Fitting 3D active shape models 21
3 Conclusions and future work 22
References 23

Describing Paris: Automated 3D Scene Analysis via Distinctive Low-Level Geometric Features 25
Martin Weinmann, Boris Jutzi and Clément Mallet
1 Introduction 25
2 Methodology 26
2.1 Selection of Optimal Neighborhoods 27

2.2 Extraction of Distinctive Low-Level Geometric Features 27
2.3 Random Forest based Classification 27
3 Adaptation for Large-Scale Urban Point Cloud Classification 28
4 Datasets 28
5 Conclusion 28
References 29

Towards Medial Axis-based point cloud simplification for LiDAR point clouds . . 33
Ravi Peters
1 Introduction 33
2 The simplification operator in practice 34
3 Medial Axis-based simplification of point clouds 34
4 Handling noise 35
References 36

Voxel based scalable registration of laser scanned point cloud data by neighbourhood voting 39
Jinhu Wang, Roderik Lindenbergh and Massimo Menenti
1 Introduction 39
2 Methodology 40
3 Initial Results 41
4 Conclusion 42
References 43

Advancing a geospatial framework to the MapReduce model 45
Roberto Giachetta
1 Introduction 45
2 Related work 46
3 The AEGIS framework 47
4 Data management 48
5 Processing data 49
6 Conclusion 51
References 51

An out-of-core octree for massive point cloud processing 53
K. Wenzel, M. Rothermel, D. Fritsch, N. Haala
1 Introduction 53
1.1 Out-of-core octree 54
2 Pine Tree 55
2.1 Approach 55
2.2 Tasks 55
2.3 Memory management 56
2.4 Usage 56
3 Point cloud filtering 57
3.1 Algorithm 57
4 Results 57
5 Conclusions 57
References 59

A web-based distributed system to process large geometric models 61
Daniela Cabiddu and Marco Attene
1 Introduction 61
2 Related Work 62
2.1 3D Model Repositories 62
2.2 Geometric Experiment Replication 62

3 The Geometric Workflow System 62
4 Mesh Transfer Protocol 63
5 Results and Discussion 64
6 Conclusion 66
References 66

File-centric Organization of large LiDAR Point Clouds in a Big Data context . . . 69
Jan Boehm
1 Introduction 69
2 LiDAR Point Clouds as Big Data 70
3 Point Clouds and Databases 70
4 NoSQL Database for File-centric Storage 71
5 Implementation Details 72
6 Spatial Query 74
7 Conclusions 74
References 75

Recent Advances on LoD for Procedural Urban Models

Gonzalo Besuievsky and Gustavo Patow

Abstract Procedural Urban Models have been proven to be an efficient technique for generating detailed city models, and an increasing range of applications, such as urban planning and simulation, aside from visualization, are using these technique for model generation. However, the amount of geometry generated could be huge and properly controlling it is still a challenge.
This paper reviews our recent works for generating LoD in Procedural Urban Models, with the aim to discuss new trends that should be tackled to explore the use of this technique in urban applications.

1 Introduction

Urban models are complex systems that need to be managed at different correlated scales such as building-scale, district-scale or city-scale, depending on the application requirements. These large-data sets are usually built from different sources like cadastral data, digital images or CAD models. The treatment of the data concerning new techniques for generating different scale-model is a current research topic in domains like modeling simulation (Rob11; Bec12) and visualization (HBT12).

Concerning model generation, procedural modeling appears as an efficient solution to the labour-intensive modeling task of content creation. Although with this technique large models with details can be generated, in many situations ranging from a single-user walkthroughs up to urban simulations, there is no need of the full geometry. The generation of level of detail (LoD) is essential for any kind of city models. The concept of representing models at different LoDs is well-known from Computer Graphics and Visualization domains. However, for procedural city models, the classic solution of processing an already generated model by using reduction techniques to simplify the model from quality parameters (LWC02) is not the best way to proceed. The full geometry generation that could contain several millions of polygons must be avoid. Also, proposals should take into account not only the complexity in terms of number of polygons, but also their semantics or physical meaning, depending on the requirements. Thus, for procedural modeling, the simplification should be generated automatically within the procedural creation.

In this paper we report our recent effort for improving procedural techniques through the development of more flexible and automatic control tools for model generation.

Geometry and Graphics Group, Universitat de Girona, Spain
e-mail: gonzalo@imae.udg.edu

2 Level-of-Details Analysis

Specific works on level of detail for building models can be found in different contexts. For cartographic generalization, Anders (And05) proposed an approach for the aggregation of linearly arranged building groups. Other works focus on building simplification by collapsing faces from known constructive structures as walls and roofs. Chang et al. (CBZ08) presented a large scale simplification approach based on ”urban legibility” intended for better preserving understandability for complex urban spaces at all levels of simplification. Other LoD proposals like the CityGML schema (Kol09) differentiates between five consecutive LoD-levels, which become more detailed with increasing LoD regarding both geometry and thematic functionality differentiation.

Although the initiatives for LoD definitions, the main question for the conception and generation is given from the criteria that drives the geometry levels. This criteria is always attached to the particular requirements. For instance, standard LoD in Computer Graphics is driven by the complexity of the geometry, whereas for CityGML the semantic is also taken into account to define the levels.

3 LoD in Procedural Modeling

The seminal works by Wonka et al. (WWSR03) and Müller et al. (MWH06) introduced Grammar-based procedural modeling for buildings. The main concept of this technique is a shape grammar, which is based on a ruleset: starting from an initial axiom primitive (e.g. a building outline), rules are iteratively applied, replacing shapes with other shapes. A rule has a labeled shape on the left hand side, called predecessor, and one or multiple shapes (also called primitives) and commands on the right hand side, called successor: $predecessor \rightarrow CommandA, CommandB : labelB; labelB \rightarrow CommandC : labelC$. The resulting geometry is formed by shapes that can be optionally assigned new labels to be further processed. The main commands, the macros that create new shapes in the classic approach, are: *Subdivision*, that performs a subdivision of the current shape into multiple shapes; *Repeat*, that performs a repeated subdivision of one shape multiple times; *Component split*, that creates new components shapes (faces or edges) from initial volumes; and *Insert* command that replaces a pre-made asset on a current predecessor.

Traditionally, grammars create a hierarchy of shapes by processing each rule’s predecessor shape, replacing it by its products. This process is executed until only terminal shapes are left. The whole production process can be seen as a Directed Acyclic Graph (DAG), where each node represents an operation applied to its incoming geometry stream and the leaf nodes are the geometry assets (see Figure 1) (Pat12). One of the strategies followed for LoD generation in order to avoid simplification over the full model is to transform the graph according to some specific criteria. In this section we review our recent contributions to the subject, using the mentioned strategy.

3.1 LoD by Configurable Rules

One of the basic ways to achieve levels of detail for procedural modeling is by directly configuring rules. An initial proposal intended for city generation was presented by Parish and Müller (PM01) based on the L-system recursive nature. Automatic LoD-generation can be obtained by starting from the building envelope as axiom, and where the output of each iteration represents a refining step in the building generation. This kind of approach can be complemented by setting manually predefined LoD geometry for different levels. Industrial

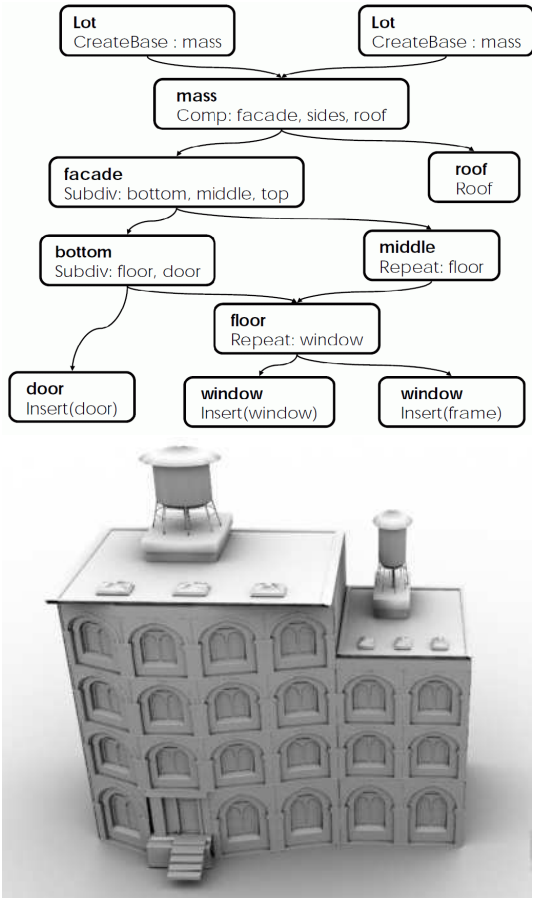


Fig. 1: A graph-based set of rules (top) is used to obtain a building model (bottom).

solutions, like CityEngine (Esr12), provide this kind of solution (see Fig. 2). Within this strategy, the final model is completed by manually introducing pre-made assets at different levels resolutions. The disadvantage here is that the mechanism requires manual writing rules for each case and also that the geometry is drastically reduced without a smooth transition between the different discrete levels.

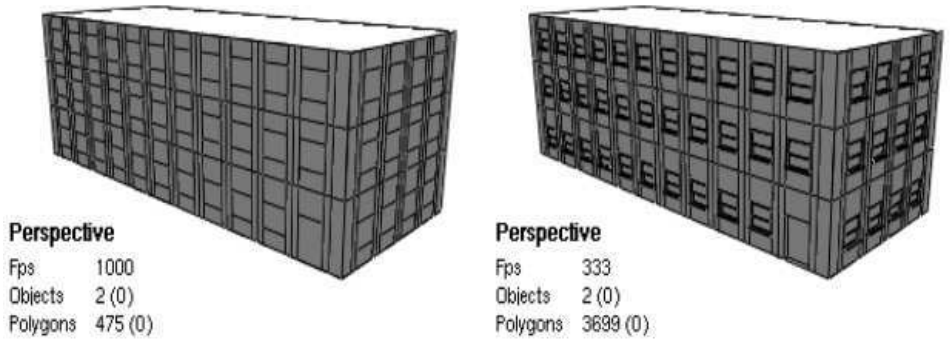


Fig. 2: LoD produced by manually configuring rules with CityEngine.

A more flexible configurable method, intended for specific simulation purposes, is presented in (BBBP14). In this case, the geometric resolution asset levels are also defined manually, but the replacements are done following a semantic criteria. First, the user decides the elements to be processed, such as windows, doors or balconies, and also provides the final assets details for all the desired level of detail (for example, windows at the LoD0, LoD1 and LoD2 levels). Then, the system searches and replaces automatically the affected elements (see Fig. 3). Although the geometry levels are still manually set, different criteria including distance to a Point of Interest, can be configured.

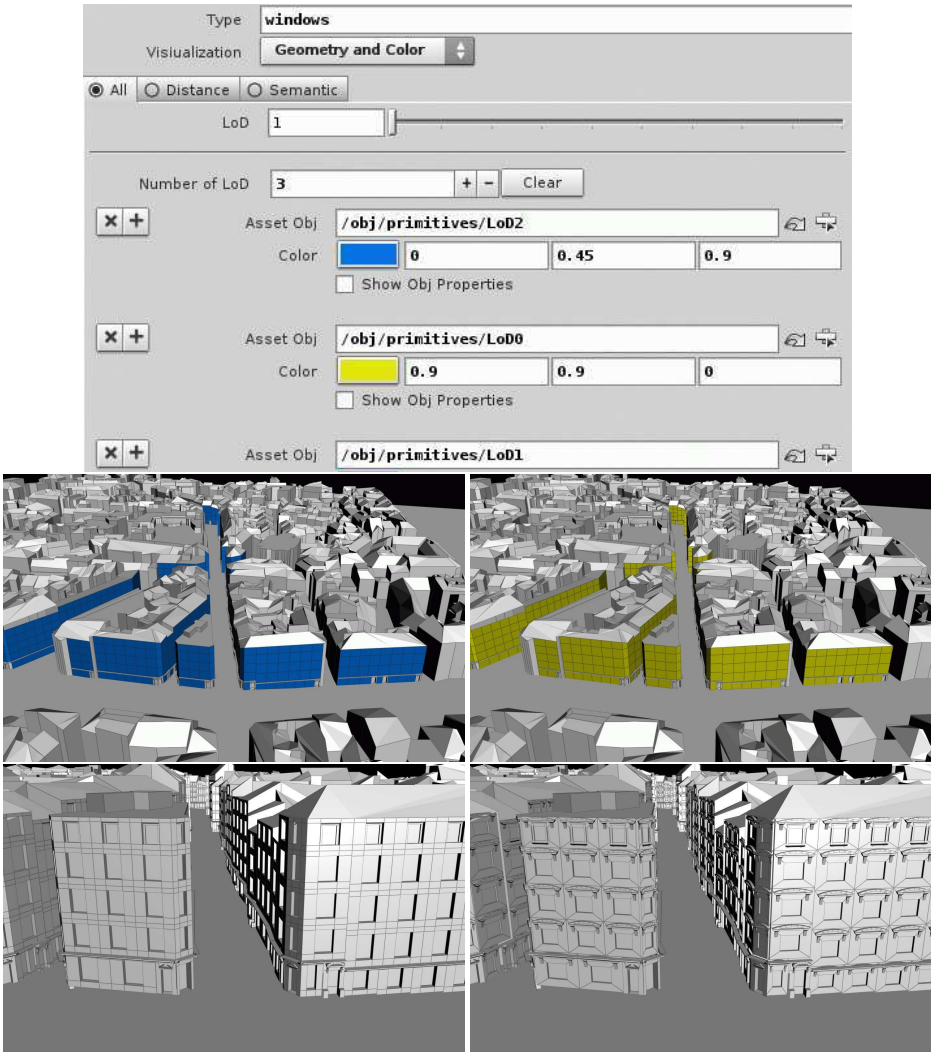


Fig. 3: Explicit LoD: interface configuration for the element window (top), geometry affected (middle) and final replacements (bottom).

3.2 Automatic LoD

In automatic LoD techniques the simplifications should be obtained from specific criteria by processing the rules without user intervention. One possible solution is to define new

commands that can be inserted in the graph with the purpose of evaluating the geometry for simplification. In (BP13b), this strategy is used in combination with programable criteria that can account not only for viewing parameters but also for semantic ones. The method processes the assets of the final graph using a system cache where a set of simplifications are stored for a continuous representation of the model. However, even with the use of an efficient cache system, this operation can turn prohibitive if evaluated for all the assets in all buildings of a large city.

Another possible strategy followed in order to process a city is to proceed hierarchically through the implicit urban levels, such as block (or district) level, building level, facade level and asset level (BP13a). The basic idea is to test from more general to more specific all structures (see Figure 4). Block-level LoD selects blocks for minimum LoD. The blocks that pass this test will be evaluated by the building-level LoD step, classifying buildings into those that fail the test and are set to a low LoD, and those that pass the test. Finally, the ones that passed are further processed for asset-level evaluation by the rewritten sets of rules.

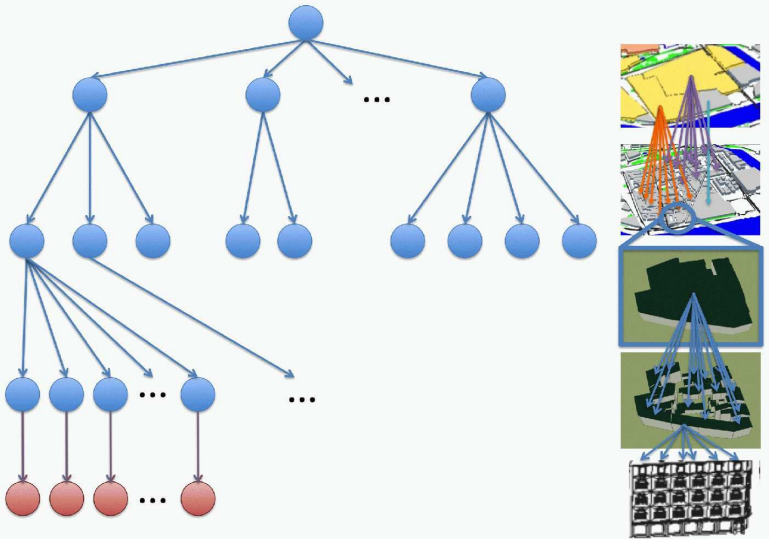


Fig. 4: The urban herarchy: districts, blocks, building, facades and final elements.

Block/Building level

The purpose of the block-oriented LoD verification is to quickly determine whether a block will be rendered at full LoD (i.e., 100%), at the lowest LoD possible (0%), or at any intermediate LoD level. A maximum range of affectation that defines a volume enclosed by a sphere is defined based on a distance criteria.

Once a block has been chosen for further processing, each of its buildings are evaluated in turn. The first test to perform is to evaluate the building convex hull against the volume range. If a building lies completely outside the volume, its LoD is set to 0% (see Fig. 5). Otherwise, the facades and the containing assets are tested.

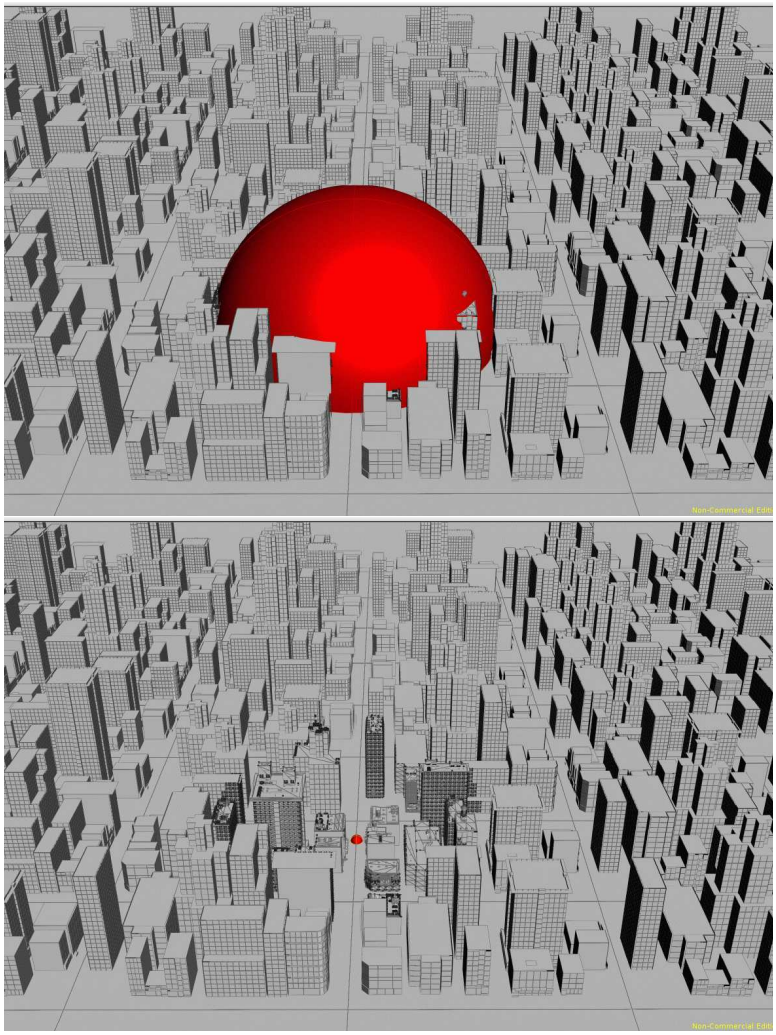


Fig. 5: The affectation region on city models. The big red sphere (left) encloses the affectation region, and it is centered at the viewpoint shown with the small sphere (right).

Asset level

As shown in Figure 1 the geometry details are inserted at the leaf of the DAG that represents the model (see Section 3). Thus, the main significant polygon reduction must be done at this level. In (BP13b), a system that automatically rewrite the graph based on specific criteria is presented. The resulted graph is further processed to obtain the final LoD model. For this purpose, new commands were introduced: *ConditionalLoD*, that evaluates the reduction that must be applied to the products from a given quality criteria, and the *InsertLoD* command that is responsible for the insertion of geometry assets at the corresponding level-of-detail.

The *ConditionalLoD* command introduces a flexible way to evaluate how much to simplify the shapes (i.e., kept at full resolution, reduced, or represented as a texture) depending on quality criteria. In the case of reduction, it also computes the percentage of polygons that must be kept. The shape to be replaced is evaluated, and then the *InsertLoD* command applies the reduction on the geometry of the corresponding asset. The *InsertLoD* command is responsible for the replacement of geometry assets at different reduction levels. It works like a switch-case scheme according to the three different kinds of replacement that can be

obtained. In the case of reduction, it simplifies the geometry according to the percentage of polygons to keep using an external polygon reduction engine that guarantees boundary preservation. Any polygon simplification algorithm that satisfies these properties can be used. Each time we require an asset reduction, we compute it and store the resulting geometry. We also provide a cache system that reuses already computed assets. Figure 6 shows several model resolution for an asset of a building that should be replaced according to the criteria used.

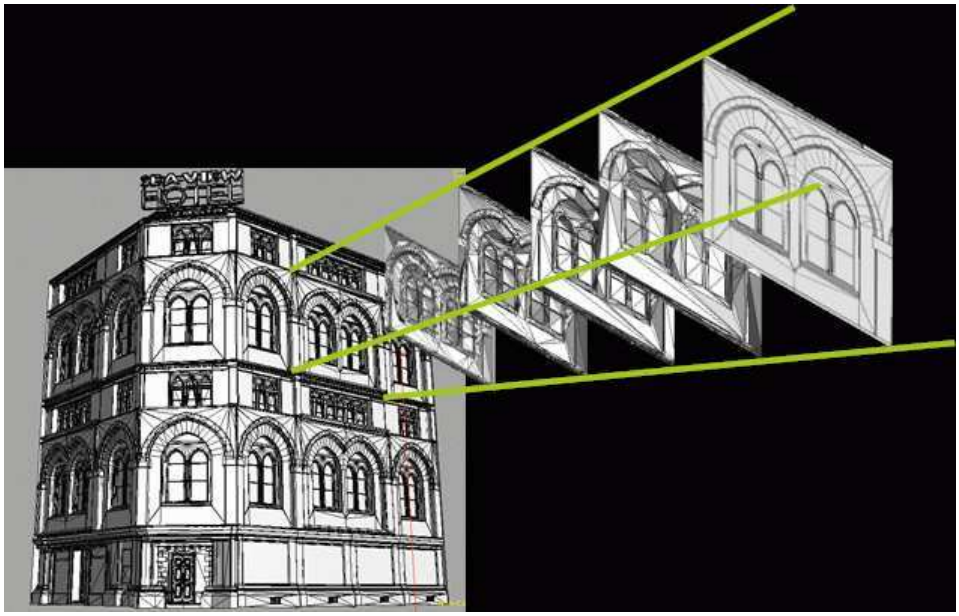


Fig. 6: Asset-level replacement. The system selects the correct asset resolution to use.

4 Challenges and Future Work

Although our promising results, completely automatic controlling the amount of geometry generated it is still a challenge for procedural modeling. Avoiding the *Geometric Explosion* would benefit this technique for extending their use for other applications requiring urban models.

Concerning next steps, we are currently working on a facade-level generalization technique. For example, a whole facade of a building could be replaced with a single texture polygon if all assets that compose it are being simplified completely to textures.

Our future work also include LoD criteria definition and uses. Among the more classical viewing criteria, LoD definitions can have specific purposes, like for example solar radiation simulation. In this case, the criteria for a higher level does not imply geometry complexity necessary but the real incidence the geometry has in the simulation.

Acknowledgements

This work was partially funded by the TIN2010-20590-C02-02 project from Ministerio de Ciencia e Innovación, Spain.

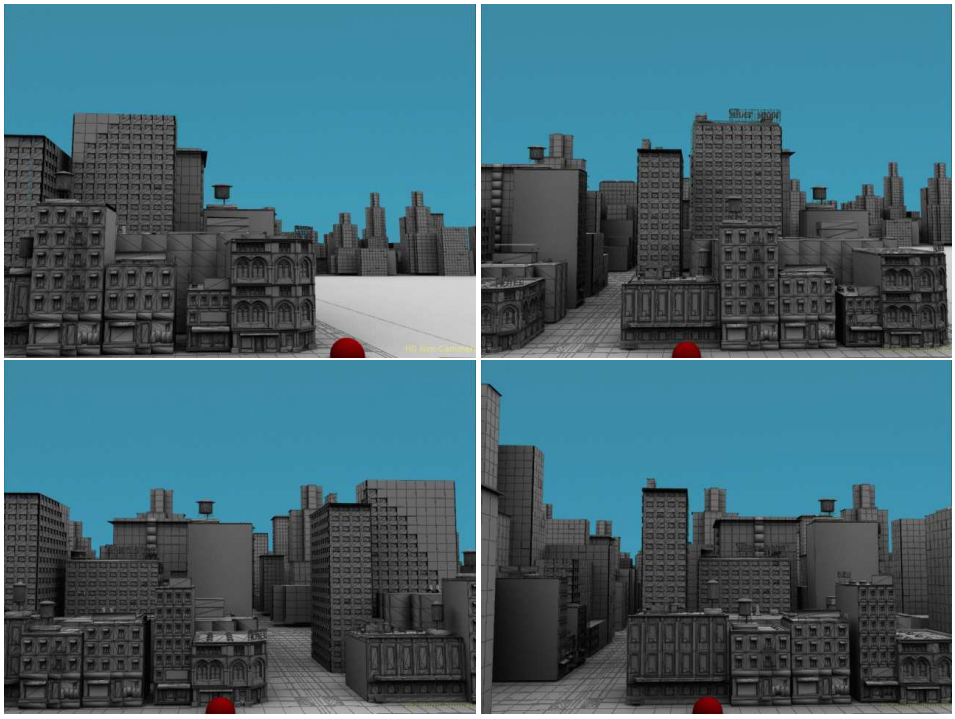


Fig. 7: Four fully rendered frames integrating the full LoD hierarchy in the complex city model.

References

[And05] ANDERS K.-H.: Level of detail generation of 3d building groups by aggregation and typification. In Proceedings of the XXII International Cartographic Conference (2005).

[BBBP14] BESUIEVSKY G., BARROSO S., BECKERS B., PATOW G.: A Configurable LoD for Procedural Urban Models intended for Daylight Simulation. Besuievsky G., Tourre V., (Eds.), Eurographics Association, pp. 19-24.

[Bec12] BECKERS B.: Solar Energy at Urban Scale. ISTE. Wiley, 2012.

[BP13a] BESUIEVSKY G., PATOW G.: City-level level-of-detail. In XXIII Spanish Computer Graphics Conference (2013), Eurographics Association, pp. 29-36.

[BP13b] BESUIEVSKY G., PATOW G.: Customizable lod for procedural architecture. Computer Graphics Forum 32, 8 (2013), 26-34.

[CBZ08] CHANG R., BUTKIEWICZ T., ZIEMKIEWICZ C., WARTELL Z., POLLARD N., RIBARSKY W.: Legible simplification of textured urban models. IEEE Comput. Graph. Appl. 28 (May 2008), 27-36.

[Esr12] ESRI: Cityengine, 2012. www.esril.com.

[HBT12] HE S., BESUIEVSKY G., TOURRE V., PATOW G., MOREAU G.: All range and heterogeneous multi-scale 3d city models. In Usage, usability, and utility of 3D city models. Edp sciences, 2012, p. 16.

[Kol09] KOLBE T. H.: Representing and exchanging 3d city models with citygml. In Lecture Notes in Geoinformation and Cartography (2009), Springer Verlag, p. 20.

[LWC02] LUEBKE D., WATSON B., COHEN J. D., REDDY M., VARSHNEY A.: Level of Detail for 3D Graphics. Elsevier Science Inc., New York, NY, USA, 2002.

[MWH06] MULLER P., WONKA P., HAEGLER S., ULMER A., VAN GOOL L.: Procedural modeling of buildings. ACM Trans. Graph. 25, 3 (2006), 614-623.

[Pat12] PATOW G.: User-friendly graph editing for procedural modeling of buildings. IEEE Computer Graphics and Applications 32 (2012), 66-75.

[PM01] PARISH Y. I. H., MULLER P.: Procedural modeling of cities. In Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (2001), Press, pp. 301-308.

[Rob11] ROBINSON D.: Computer Modelling for Sustainable Urban Design: Physical Principles, Methods and Applications. Earthscan Publications, Limited, 2011.

[WWSR03] WONKA P., WIMMER M., SILLION F., RIBARSKY W.: Instant architecture. ACM Transaction on Graphics 22, 3 (July 2003), 669-677. Proceedings ACM SIGGRAPH 2003.

Locally refined spline surfaces for shape representation, and feature detection in large GIS data sets

Oliver Barrowclough, Tor Dokken and Vibeke Skytt

Abstract GIS data sets are typically large and a major challenge is how to reduce Big Data to small data. In this paper we describe the use of a novel representation, namely, locally refined (LR) spline surfaces, in GIS. The representation offers a method to model various data sets that exhibit relatively smooth behavior in a compact way. We briefly describe the properties of the LR B-spline representation, and discuss the details of two approximation methods.

1 Introduction

Traditional tensor-product spline surfaces (piecewise polynomials) are a well established representation in fields such as CAD, where they are used primarily to model smooth shapes. Their use has also previously been explored in representing topographic data sets. For example, spline approximations of data from the Shuttle Radar Topography Mission (SRTM) have been used, particularly for the application of filling data voids (3). However, for data which exhibits local detail, tensor-product require global refinement, resulting in large data sets. This means that such representations are limited to globally smooth data sets. Locally refined spline surfaces have, in contrast to tensor produce spline surfaces, the ability to represent local variations in shape without globally increasing the data size of the surface. In this paper we describe the properties of LR B-spline and discuss two methods of approximation; namely, least squares approximation with smoothing and multilevel B-spline approximation.

2 LR B-spline surface representations

In the past decades, a number of approaches to local refinement of splines have been pursued, including hierarchical splines (4), T-splines (6) and LR B-splines (2). In this paper, we restrict our attention to LR B-splines. An LR B-spline surface is a parameterized surface and the parameter domain is rectangular in $\mathbb{R}^2$. The domain is composed of rectangular boxes, not a regular grid, as is the case for tensor product spline surfaces. Figure 3 shows the

Oliver Barrowclough, Tor Dokken and Vibeke Skytt
SINTEF ICT, Forskningsveien 1, PO Box 124 Blindern, 0314 Oslo, Norway,

e-mail: oliver.barrowclough@sintef.no ·
e-mail: tor.dokken@sintef.no ·
e-mail: vibeke.skytt@sintef.no

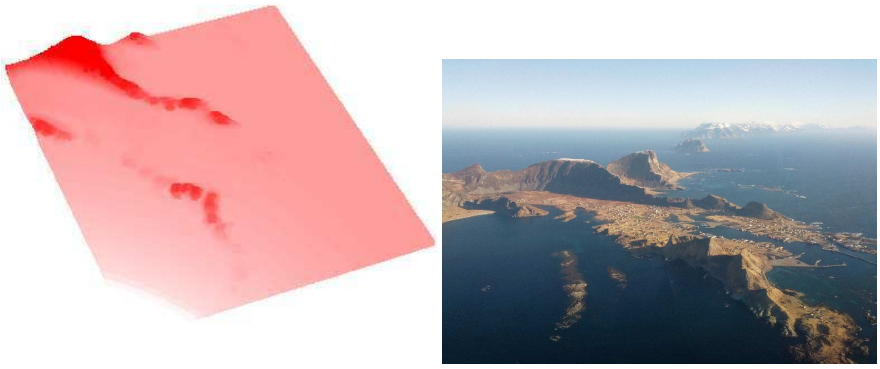


Fig. 1: A point cloud (left) covering a region of the island of Værøy, Northern Norway (right). Points are coloured according to the distance from sea level.

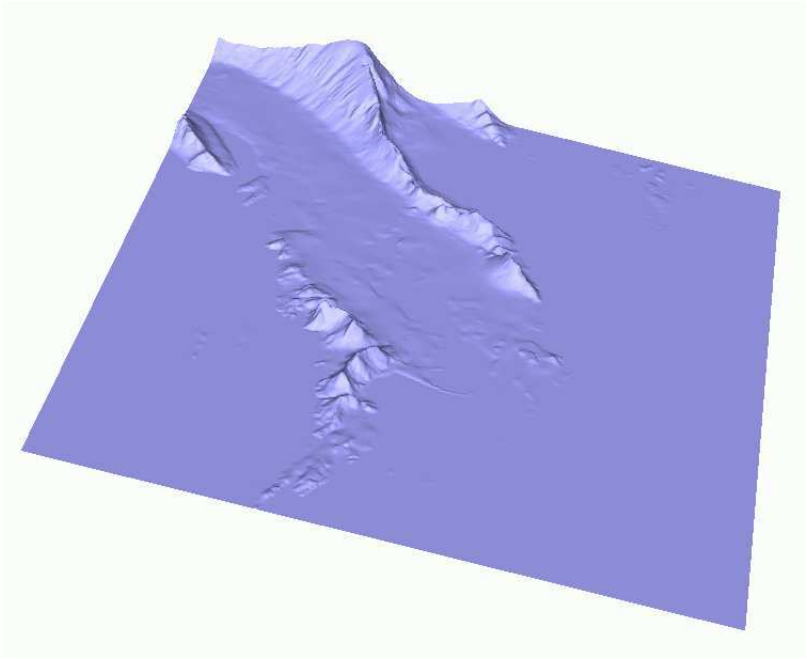


Fig. 2: An LR-B spline surface approximation of point cloud data from Værøy, Norway

parameter domain corresponding to a spline surface for an LR B-spline representation and for a tensor-product representation. The surface, pictured in Figure 1, can be represented by both spline formats.

An LR B-spline surface representation of geo-spatial data provides an approximation to the data points. The representation is well suited to compact modelling of mainly smooth point data sets, possibly with local details. This contrasts with triangulations, which normally interpolate the point data and are better suited to modelling non-smooth and discontinuous terrains, and features such as buildings or rock formations.

LR B-spline surfaces possess most of the properties of a tensor-product spline surface such as non-negative B-spline basis functions and partition of unity, which ensure numerical stability and modelling accuracy. However, some more effort is required to ensure linear

independence of the B-spline functions. We refer the reader to (2) for a detailed description of the LR-spline representation.

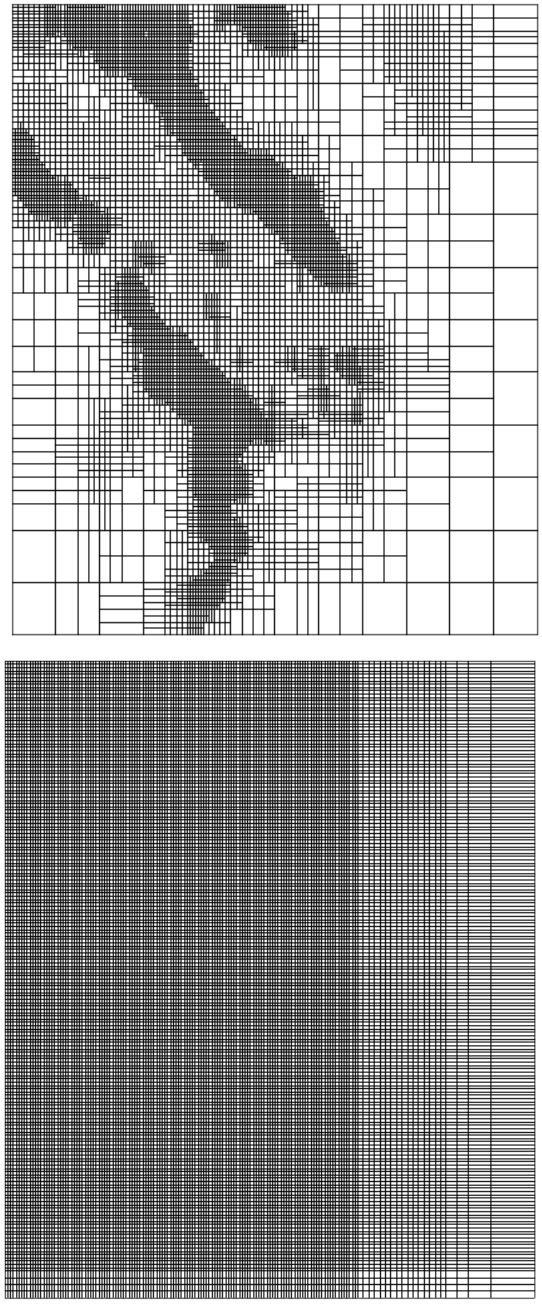


Fig. 3: Parameter domain meshes for an LR B-spline surface (left) and a tensor-product surface with the same level of detail. There are far fewer degrees of freedom in the LR case. The lines in the vertical and horizontal directions are known as knot lines, and they define the piecewise polynomial structure.

3 Approximation methods for LR B-spline surfaces

An LR-B spline surface F is expressed with respect to parameters u and v as

$$F(u,v) = \sum_{i=1}^{K} s_i P_i N_i^{d_1,d_2}(u,v),$$

where P_i are the surface coefficients, N_i are the associated B-splines and s_i are scaling factors that ensure partition of unity. The B-splines are defined on a set of knots in both parameter directions and have polynomial degree d_1 and d_2 in the first and second parameter direction respectively. The B-splines have limited support and the extent is given from the polynomial degrees. In the context of GIS, the surface is normally parameterized according to the x,y-coordinates of the point cloud giving a height function, also known as a 2.5D surface. However, in steep areas and areas with an overhang, a parameterized 3D surface may be beneficial.

Typically, an iterative approach is used to approximate the point cloud with an LR B-spline surface. The cloud is initially approximated by a very lean surface. The surface is then refined in regions where the distance between the points and the surface is not within an acceptable tolerance. The surface is refined by inserting new knot lines into the surface description, thereby splitting one or more B-splines. In contrast to the tensor-product case these knot lines do not need to cover the entire parameter domain of the surface. The local nature of these refinements means that the number of added degrees of freedom is kept to a minimum.

We have implemented two methods, either of which can be applied to compute the surface coefficients at each iteration step. The methods are described below.

3.1 Least squares approximation

Least squares approximation is a global method for surface approximation whereby the following expression is minimized with respect to the coefficients P_i, over the entire surface domain:

$$\alpha_1 J(\sigma) + \alpha_2 \sum_{k=1}^{K} (F(x_k,y_k) - z_k)^2.$$

Here, $\mathbf{x}_k = (x_k,y_k,z_k), k = 1,\ldots,K$ are the input data points. The approximation is weighted (by the scalars α_1 and α_2) in order to favour either the smoothing term or the least squares approximation respectively. The smoothing term is given by

$$J(F) = \iint_{\Omega} \int_0^{\pi} \sum_{i=0}^{3} w_i \left(\frac{\partial^i F(u_0 + r\cos\phi, v_0 + r\sin\phi)}{\partial r^i} \bigg|_{r=0} \right) \mathrm{d}\phi \mathrm{d}u_0 \mathrm{d}v_0.$$

The first, second and third partial derivatives of the surface in each point in the parameter domain, are used to define directional derivatives which are integrated around a circle. The result is integrated over the parameter domain. Experience shows that the approximation term must be prioritized in order to achieve a good approximation to the data points.

Elaborating the minimization functional gives a linear equation system in the surface coefficients.

3.2 Locally refined multilevel B-spline approximation (LR-MBA)

Multilevel B-spline approximation (MBA) is a local approximation method (5). This method is applied in the functional case. Each coefficient is updated with respect to the distance between the data points in its support and the function.

Let $\mathbf{x}_c = (x_c, y_c, z_c), c = 1, \ldots, C$ be the data points in the support of a given B-spline. The corresponding new coefficient P_i is determined by

$$P_i = \frac{\sum_c (s_i N_i(x_c, y_c))^2 \phi_c}{\sum_c (s_i N_i(x_c, y_c))^2},$$

where ϕ_c is computed for each data point as

$$\phi_c = \frac{s_i N_i(x_c, y_c) z_c}{\sum_l (s_l N_l(x_c, y_c))^2}.$$

The sum in the previous denominator is taken over all B-splines which contain (x_c, y_c) in their support.

In the multilevel tensor-product setting, the difference between the function and the data values is, at each step, used to compute a difference function approximating the error. The final surface is evaluated by computing the sum of the initial surface and all the difference functions. In the LR B-splines setting, the computed difference function is added to the initial surface at each step giving a unified expression without a global increase in data size.

3.3 Properties

The least squares approach is a global method with very good approximation properties. However, in particular if the data are non-smooth and a large weight on the approximation is chosen, it may oscillate in areas which are scarcely populated by data points to optimize the accuracy with respect to data points in nearby areas.

The MBA approach is local and does not see the characteristics of the point set outside the current B-spline. The method does not give the same accuracy as least squares for the same number of coefficients, but tends to produce well behaved surfaces in areas where the initial data set is unevenly distributed.

4 Examples for GIS data sets

Two LR B-spline surface approximations to this point set are pictured in Figure 4, using the methods outlined above. The first one is created as a least squares approximation of the points while the second is created using the LR-MBA algorithm. The second surface is fitted within a tolerance of 0.5m with an exception of 5 points and the average distance between the points and the surface is 0.02m. The least squares surface is less accurate, but also more lean. The point set is already thinned and quite small ($\sim$200 000 points) so the gain with respect to data size is small. For the MBA surface, the data size is about half that of the point set while for the least squares surface, the data size is halved again.

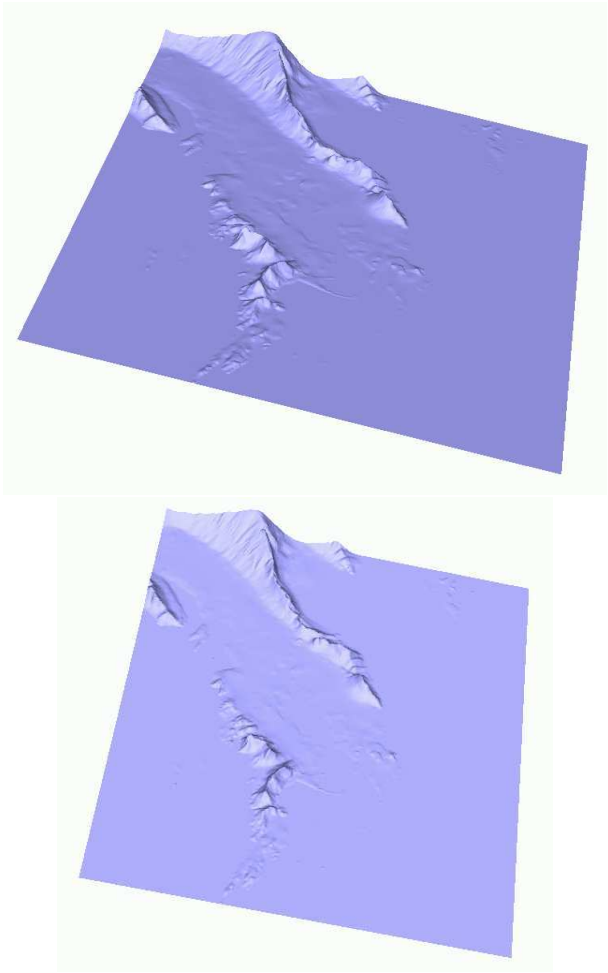


Fig. 4: The surfaces resulting from the approximation using the least squares method (left) and the LR-MBA method (right)

5 Detection of features in point sets

Smooth spline surfaces can also be used to capture the trend in a point cloud. By comparing the distance between the points and the surface, features can be extracted. Consider the less accurate version of the Værøy point cloud shown in Figure 1. Figure 5 shows this surface together with the points that are less well approximated. The green points lie below the surface and the red points are above. In the overview picture, we can find small rocks in the sea. Also, on closer inspection of the hill, we see that the ridge is clearly identified.

Acknowledgements The research leading to these results is partly funded by the European Communitys Seventh Framework Programme FP7/2007-2013 under grant agreement No. 318787 (IQmulus) and the SESAR Joint Undertaking project 12.04.09 (Single European Sky ATM Research).

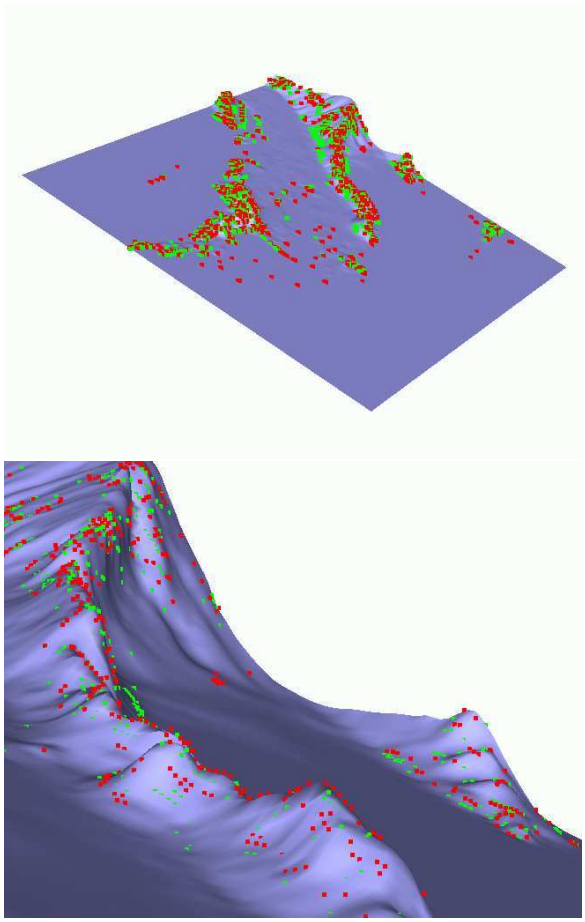


Fig. 5: The surface and points which are out-of-tolerance at the given level of approximation. Green points are below the surface and red points lie above the surface. On the figure to the right, the ridgeline is clearly identified.

References

[1] Cohen, E., Lyche, T. Riesenfeld, R.: Discrete B-splines and subdivision techniques in computer-aided geometric design and computer graphics. Computer Graphics and Image Processing 14 (1980), 87 – 111

[2] Dokken, T., Pettersen, K.F., Lyche, T.: Polynomial splines over locally refined box-partitions. Computer Aided Geometric Design (2013) doi:10.1016/j.cagd.2012.12.005

[3] Farr, T.G., et al.: The Shuttle Radar Topography Mission, Rev. Geophys., 45, RG2004, (2007), doi:10.1029/2005RG000183.

[4] Forsey, D.R., Bartels, R.H.: Hierarchical B-spline refinement. SIGGRAPH 88 Conference Proceedings, 4: 205-212, (1988)

[5] Lee, S., Wolberg, G., Shin, S.Y.: Scattered data interpolation with multilevel B-splines. IEEE Transaction on Visualization and Computer Graphics, 3(3): 229-244, (1997)

[6] Sederberg, T.W., Zheng, J., Bakenov, A., Nasri, A.: T-splines and T-NURCCs. ACM Trans. Graph., 22(3): 477-484, (2003) doi:10.1145/882262.882295

Automatic buildig reconstruction from digital surface models using 3D active shape models

Beril Sirmacek and Roderik Lindenbergh

Abstract Recent technological developments help us to acquire high quality 3D input data sampling man- made objects and vegetation. However, even the highest resolution 3D input data does not contain information about the locations and the 3D geometrical properties (i.e. boundaries, heights, shapes) of the objects. Therefore, robust and fully automatic detection and 3D reconstruction of the buildings is still an important research topic. Herein, we introduce an improved version of a previously proposed active shape fitting algorithm. The new active shape fitting algorithm uses height information instead of 2D binary images which was the considered input for the earlier version, that leads us to achieve higher accuracy in 3D modelling. For the first tests, we use airborne laser scanning (ALS) data of some buildings in Delft, The Netherlands. Our first results show us that the proposed algorithm can be used for fast 3D reconstruction of urban areas with very high accuracy even if the building geometries are very complex. The 3D reconstruction results can also be easily enhanced further, in order to represent rooftop, windows and other construction details, when it is necessary.

1 Introduction

Three-dimensional (3D) building models are useful especially for 3D change detection and quick map updating. Besides, the models can be used for generating 3D simulations of possible events (i.e. flood, earthquake, air pollution, etc.). Recent technological developments help us to acquire high quality 3D input data. However, even the highest resolution 3D data does not clearly indicate where the buildings are and what kind of 3D geometry they have. In order to bring a solution to this problem, we introduce a fully automatic method using digital elevation models (DSMs) as input. The quality and the ground sampling distance of the DSMs can be different. Besides, they might contain noise which makes the 3D representation cluttered. If we also consider the high geometrical variety of building structures, it becomes very obvious that robust and intelligent algorithms are needed. The earliest studies in this field generally depend on edge, line and polygon extraction from grayscale images or from DSMs [1, 2, 3, 4]. Sirmacek and Unsalan [5] developed a fast method to detect shapes of rectangular buildings which depends on growing a rectangular active shape in two dimensions. The active shape tries to achieve the best fitting position on a binary mask which contains the extracted building edges. Unfortunately, they could not detect complex building geometries, but only the rectangular ones. In a following study Sirmacek et al. [6] improved the algorithm in order to detect buildings with complex footprint shapes. This is

Delft University of Technology, Department of Geoscience and Remote Sensing, Stevinweg 1, 2628CN, Delft, The Netherlands, e-mail: B.Sirmacek@tudelft.nl · e-mail: R.C.Lindenbergh@tudelft.nl

done by fitting a chain of active shape models on the input data which is again a 2D binary mask containing building edges. Although they achieved good results to show building footprint shapes, building height values were not representative (only single height value is assigned to each building model). They have tested performance of this algorithm on six different DSM sources and compared the reconstruction accuracies quantitatively [7].

The proposed method basically uses a virtual model which is located on the automatically detected seed point locations. The virtual model grows in 3D space and also tries to fix its orientation in order to fit to the building information in the input DSM. Model based methods, as the name indicates, make use of a prior model of what is expected to be found in the image and typically attempt to find the best match of the model to the data in a new input image. Having the fitted models, one can apply further measurements, test and verification steps to see whether the object of interest is accurately presented or not. This is known as a top-down approach. In this study a 3D cube is the initial model which tries to fit on the building segments with elongated shape and uniform height. Using active shape models, we achieve to high performances to cope with the variability of the input data source and the geometry of the objects. Mathematical modules of this algorithm are similar to the previously proposed algorithm [7], however higher robustness and reconstruction accuracy is obtained by 3D active shape fitting instead of using 2D active shape models.

2 Method and initial results

The proposed method basically works by following the steps given below.

2.1 Extracting approximate building footprint segments

This step applies local thresholding to the input DSM in order to detect approximate location and footprint shapes of the buildings in the input DSM. (Figure 1 (a), (b), (d), (e)) Segments are labelled with connected component analysis and small size components are eliminated since they cannot represent buildings. However, rest of the segments do not indicate the building footprints straightforward. The noise in the input DSM might result having fluctuations at the building boundaries. Besides neighbouring trees might be detected as parts of the buildings (as the building segment in Figure 1 (d)).

2.2 Assigning seed points

For each segment, which is obtained by the previous step, a height histogram is generated (as the example illustrated in Fig. 2 (h)). Local maxima of the histogram are used to distinguish building parts with uniform height. If the detected parts together do not represent the whole building segment, then more seed points are located on the mass centers of the rest of the area. For each building part it is evaluated whether it is elongated or not. If it is not elongated, more seed points are located on the elongated pieces as it is introduced in the previous study [6]. For our showcase building, the automatically extracted initial seed points are shown in Fig. 2 (b) and (c).

2.3 Fitting 3D active shape models

3D active shape models are fit on the input DSM, by growing them starting from the seed point locations (Figure 2 (d-g)). Active shape model fitting is done by growing a cube shape starting from the detected seed points, till the 3D shape model finds the best orientation and size which gives the best energy value.

In the previously introduced work [5], Sirmacek and Unsalan proposed an automatic rectangular shape approximation approach (called box-fitting). First they used color-invariant features to extract possible building rooftop segments. Mass centers of the segments are assumed as seed-points (as approximate building centers). Seed-point locations are used to grow a virtual active rectangular shape based on an energy criteria. To fit box shapes to buildings, they start to grow the active rectangular shape on each (x_s, y_s) seed point location. When the active rectangular shape grows in θ direction and hits to the Canny edges, the growing process is stop and they calculate an energy value E_θ. This process is repeated for the same seed point for different θ directions and each time E_θ energy value is computed. This energy value is defined as the sum of minimum distance between virtual building edge pixels and the real building segment boundary pixels in perpendicular direction as given below;

$$E_\theta = \sum_{i=1}^{n} min(sqrt((x_v(i) - x_e(j))^2 - (y_v(i) - y_e(j))^2)) \quad (1)$$

Here, $(x_v(i), y_v(i))$ represent coordinates for ith pixel on the edges of the virtual rectangular shape. $(x_e(j), y_e(j))$ represents the jth pixel on the Canny edges of the building segment. For the same seed-point, they apply growing process for all $\theta \in [0, \pi/6, \pi/3, \pi/2, 2\pi/3, ..., 2\pi]$ angles with $\theta_{dif} = \pi/6$ radian turning steps. As they reduce θ_{dif} step sizes here, they can obtain more accurate approximations, however in this case they need more computation time. This issue is discussed in their article in detail. After calculating E_θ for all θ ($\theta \in [0, \pi/6, \pi/3, \pi/2, 2\pi/3, ..., 2\pi]$) angles, they pick the estimated box which exhibits the smallest E_θ energy as detected building shape. Since most buildings are in rectangular shapes or generally they appear like composition of rectangular building segments, it makes sense to extract rectangular shapes on buildings. The main advantage of using the box-fitting approach is that approximate building shapes still can be found even if the building edges are not well-determined, or even if there is not a closed edge. However, other region growing algorithms fail to extract an object shape in these cases, since the growing region can flow out easily when the parameters are not set precisely.

In another earlier work [6], Sirmacek et al. have improved the box-fitting approach to detect complex building shapes. In their approach, they start with deciding if the building segment is complex or not. If there are inner yards (holes) inside of the segments, they assume them as complex shape. They make this decision by computing Euler number on binary building segment. Euler number of a binary object is defined as the total number of objects (equal to 1 when only one building segment is taken) minus the total number of holes in those objects. If Euler number is calculated as equal or less than zero, the building is assumed as a complex shaped building. In this case, they use the following steps to estimate the shape. They divide the building segment into elongated pieces using its skeleton. To do so, they detect junctions and endpoints of the building segment skeleton. A junction is defined as a pixel which has more than two pixels indicating the building skeleton in the neighborhood. An endpoint is defined as a pixel which has one pixel in the neighborhood.They divide the skeleton into pieces by removing these junction pixels from the skeleton. For each obtained skeleton piece, they divide it again into l pixel length pieces if it is longer than l pixels. They assume center pixels of obtained skeleton pieces as their seed-point locations to run the box-fitting algorithm.

Herein, we apply the similar approach by using the similar approach by improving the Equation 1 in order to consider height information as well. To do so, at each growing it-

eration mean height value of the internal virtual box pixels and mean height value of the immediate outer edge neighbourhoods of the virtual box borders are compared to each other. If this difference is less than 2 meters than growing process is carried on as normal box-growing approach. Otherwise, the box growing is stop and the next orientation is considered for trying to fit the virtual box.

After detecting building groundoor shapes, it is possible to try to fit coarse rooftop models on the building models. To do so, it is possible to focus on classifying the building rooftops as flat or gable simply. Obtained information can be useful to insert 3D roof models. The classification can be done by checking the ridge-line information. The ridge-line's can be detected by using derivative filter. If there is not enough ridge-line information detected then the rooftop can be classified as 'flatroof' otherwise it can be classified as a 'gableroof'. The application will be discussed at the workshop on real showcases.

3 Conclusions and future work

In the workshop, we will show accuracy assessments on buildings with different geometrical complexities. Besides, we will test the algorithm on DSMs which are generated by different sensors and different techniques. We expect these quantitative tests will provide better understanding of qualities, capabilities and possibilities for applications. In addition to that, we would like to improve details of the 3D models by adding rooftop models as shown in the example study presented by Xiong et al. [8].

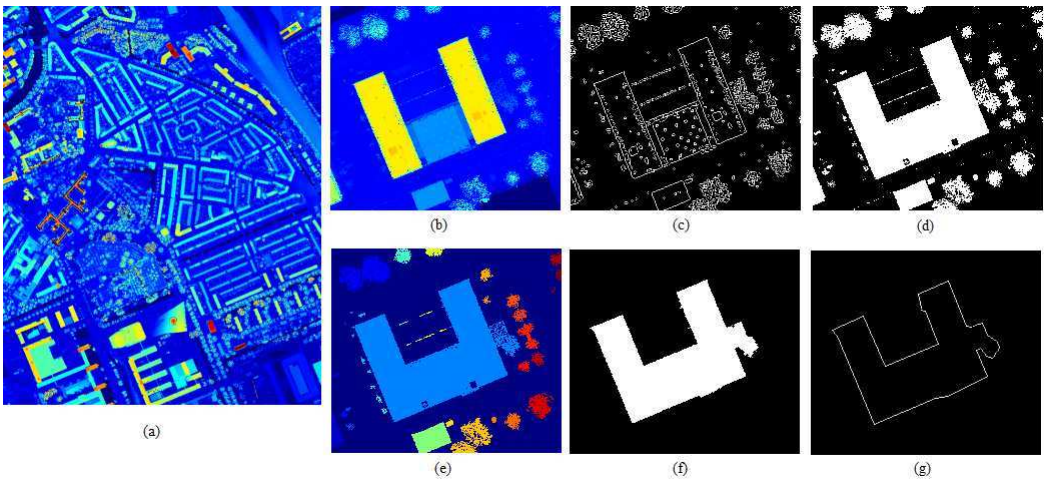


Fig. 1: (a) Airborne laser scanning DSM data taken over Delft city, (b) an example building to illustrate the steps of the proposed algorithm, (c) Canny edges of the DSM shown in (b), (d) local thresholding result for the DSM shown in (b), (e) connected components labelled with different colors for detected segments shown in (d), (f) object of interest is chosen by removing the connected components which are smaller than a certain size threshold, (g) result obtained by using 2D active shape fitting approach which is presented in the earlier work [8] (neighbour tree is also detected as a part of the building).

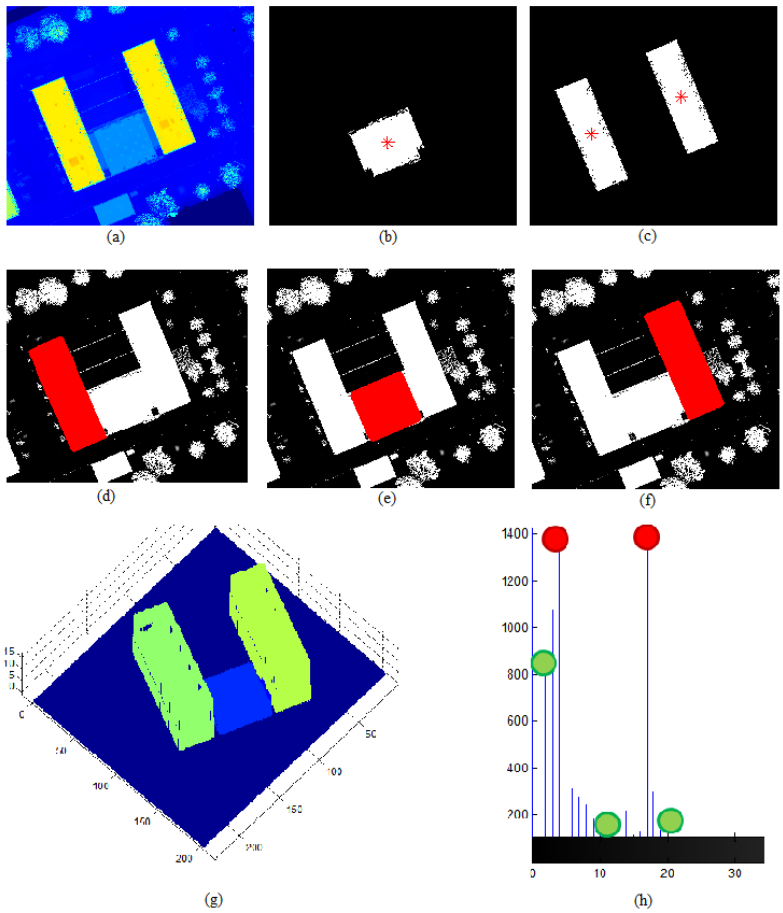


Fig. 2: (a) Gridded ALS data of an example TUDelft building, (b) and (c) show the automatically extracted seed points, (d), (e) and (f) show the 3D active shape fitting results using the seed points which are shown in (b) and (c), (g) all 3D active shape fitting results are shown together as a building model in 3D environment, (h) histogram of heights for the input ALS shown in (a) here green points indicate the detected local minimums and red points indicate the detected local maximums of the height histogram.

Acknowledgements This research is funded by the FP7 project IQmulus (FP7-ICT-2011-318787) a high volume fusion and analysis platform for geospatial point clouds, coverages and volumetric data set.

References

[8] S. Krishnamachari, R. Chellappa, ”Delineating buildings by grouping lines with MRFs”, IEEE Transactions on Image Processing, Vol. 5, No. 1, pp/ 164-168, 1996.

[2] P. Saeedi, H. Zwick, ”Automatic building detection in aerial and satellite images”, in International Conference on Control, Automation, Robotics and Visualization, Vol.1, pp. 623-629, ICARCV 2008, Hanoi, Vietnam, 2008.

[3] D. Canu, J. Gambotto, J. Sirat, ”Reconstruction of buildings from multiple high resolution images”, in proceedings of international conference on image processing, pp. 621-624, 1996.

[4] H. Arefi, J. Engels, M. Hahn, H. Mayer, ”Levels of detail in 3D building reconstruction from lidar data”, In proceedings of international archives photogrammetry, remote sensing, and spatial information sciences, vol. 37, pp. 485-490, 2008.

[5] B. Sirmacek, C. Unsalan, ”Building detection from aerial imagery using invariant color features and shadow information”, International Symposium on Computer and Information and Sciences ISCIS?2008, Istanbul, Turkey, October, 2008.

[6] B. Sirmacek, P. d’Angelo, P. Reinartz, ”Detecting complex building shapes in panchromatic satellite images for digital elevation model enhancement”, ISPRS Workshop on Modeling of Optical Airborne and Space-borne Sensors, Istanbul, Turkey, October 2010.

[7] B. Sirmacek, H. Taubenboeck, P. Reinartz, M. Ehlers, ”Evaluation of Automatically Generated 3D City Models Based on Six Different DSMs from Airborne and Space-borne Sensors”, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, Optical Multi-Angular Data Special Issue, January 2012.

[8] B. Xiong, S.J. Oude Elberink, G. Vosselman, ”A graph edit dictionary for correcting errors in roof topology graphs reconstructred from point clouds”, ISPRS Journal of Photogrammetry and Remote Sensing, March 2014.

Describing Paris: Automated 3D Scene Analysis via Distinctive Low-Level Geometric Features

Martin Weinmann, Boris Jutzi and Clément Mallet

1 Introduction

The automated analysis of 3D point clouds has become a topic of great importance in photogrammetry, remote sensing, computer vision and robotics. One avenue of research directly addresses the analysis of urban environments, where recent investigations focus on 3D reconstruction (18; 33; 13), consolidation of imperfect scan data (32; 5), object detection (24; 28; 4), extraction of roads and curbstones or road markings (2; 34; 9), urban accessibility analysis (25), recognition of power-line objects (12), extraction of building structures (27), vegetation mapping (31), large-scale city modeling (14), semantic perception for ground robotics (10) and semantization of complex 3D scenes (1). A common task for many of these different applications consists of point cloud classification (11; 19), where each 3D point is assigned a specific class label.

Addressing the issue of urban point cloud classification – where the spatial 3D data may be collected via airborne, terrestrial and/or mobile laser scanning – we face a variety of challenges arising from the complexity of respective 3D scenes caused by an irregular sampling and very different types of objects. Since the results of urban 3D scene analysis may vary from one dataset to another, publicly available standard datasets are desirable in order to compare the performance of different methodologies. Consequently, there has been a steadily increasing availability of 3D point cloud datasets in recent years (20). However, urban point clouds with respective point-wise manual annotations in terms of semantic class labels are still rarely available, although this represents a prerequisite for supervised point cloud classification. One of the most widely used datasets is the Oakland 3D Point Cloud Dataset (17) which contains approximately 1.6 million labeled 3D points. This dataset however is not tailored for designing large-scale processing pipelines.

Due to the recent technological advancements, it is meanwhile possible to collect multi-dimensional spatial data in a fast and efficient way via terrestrial and mobile laser scanning. In order to foster research in advanced 3D point cloud processing, two publicly available labeled point cloud datasets representing densely sampled urban environments have been presented (26; 22). These can be considered as a first step towards large geospatial datasets

Martin Weinmann
Institute of Photogrammetry and Remote Sensing, Karlsruhe Institute of Technology (KIT), Englerstrasse 7, 76131 Karlsruhe, Germany, e-mail: martin.weinmann@kit.edu

Boris Jutzi
Institute of Photogrammetry and Remote Sensing, Karlsruhe Institute of Technology (KIT), Englerstrasse 7, 76131 Karlsruhe, Germany, e-mail: boris.jutzi@kit.edu

Clément Mallet
IGN, MATIS lab., Université Paris Est, 73 avenue de Paris, 94160 Saint-Mandé, France, e-mail: clement.mallet@ign.fr

in terms of city-scale or even larger. The availability of such datasets is important for comparing large-scale processing workflows which is the core issue of a recent benchmark (22) and addressed in this paper.

In the following, we first present a methodology which is closely linked to recent investigations on 3D scene analysis involving optimal neighborhoods and different classifiers (30; 29). Based on these investigations with a very detailed evaluation, we can directly select the most appropriate solution with respect to urban 3D scene analysis. The considered criteria address feasibility in terms of simplicity and reproducibility of the involved components as well as performance in terms of accuracy and computational effort. Subsequently, in order to extend the applicability of the selected methodology, an adaptation for large-scale urban point cloud classification is presented in this paper which does not affect the quality of the results, but allows a successive processing of huge point clouds with billions of points in reasonable time.

2 Methodology

For 3D scene analysis in terms of uniquely assigning each 3D point a semantic label, we propose a fully automatic and non-parametric methodology which consists of three successive steps as shown in Fig. 1. In the first step, each individual 3D point is characterized with a local 3D neighborhood of optimal size (Section 2.1). This allows an extraction of highly distinctive features which is pursued in the second step, where various geometric 3D and 2D features are taken into consideration (Section 2.2). Finally, in the third step, the distinctive features and a small set of training examples are provided to a supervised classification scheme (Section 2.3).

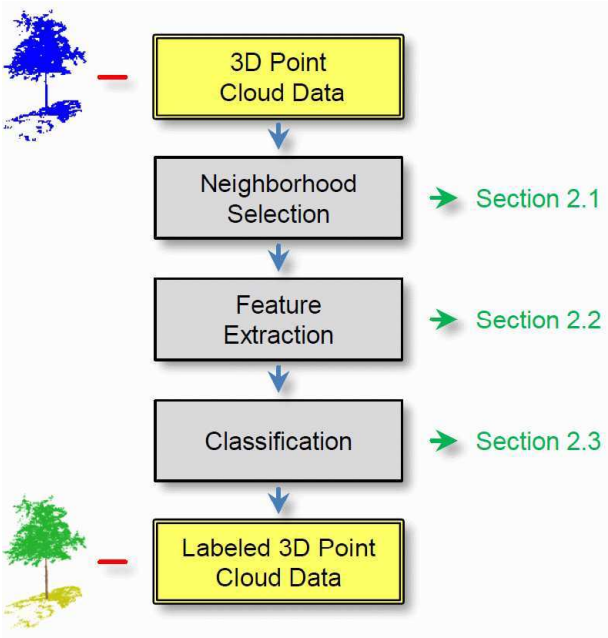


Fig. 1: The proposed methodology consisting of three components: (1) selection of optimal neighborhoods, (2) low-level geometric feature extraction and (3) individual point classification.

2.1 Selection of Optimal Neighborhoods

In general, there are very different opportunities to define the local 3D neighborhood of a given 3D point such as a spherical or a cylindrical neighborhood with fixed radius. A further definition accounting for more flexibility in case of varying point density is based on selecting the k closest neighbors of the respective 3D point. However, all these definitions rely on the specification of one free scale parameter. Consequently, it would be desirable to optimally select the respective parameter in an automatic manner. For this purpose, we may also take into account that the optimal choice certainly depends on the respective 3D structures and might thus strongly depend on the respective class label.

In order to adequately address these issues, we propose to derive the optimal neighborhood for each individual 3D point based on its closest neighbors. Respective state-of-the-art approaches which focus on the optimal choice for k are based on the local surface variation (23) or an iterative scheme involving curvature, point density and noise of normal vector estimation (16; 15). As an alternative, the use of dimensionality features has been proposed which is based on the idea that the optimal neighborhood size favors one dimensionality the most (7). Very recently, a definition based on minimizing the disorder of points within a 3D covariance ellipsoid has been proposed (30), which can be considered as more general since it avoids assumptions on the presence of specific geometric structures in the scene.

Exploiting the local disorder of 3D points as measure for deriving optimal neighborhoods, the resulting 3D neighborhood size depends on contextual information preserved in the spatial arrangement of neighboring 3D points and may even be different for each individual 3D point. Even though optimal neighborhood size selection causes a higher computational effort with respect to both processing time and memory consumption, it should be taken into consideration since the classification accuracy is significantly improved according to recent investigations (30).

2.2 Extraction of Distinctive Low-Level Geometric Features

Since many of the publicly available 3D point cloud datasets only contain information about the spatial 3D geometry, we focus on the use of geometric features. Once the optimal neighborhood size for a respective 3D point has been derived, we may assume that highly distinctive features can be derived at this scale. However, the selected scale typically corresponds to a relatively small absolute size, and the respective local 3D structure can therefore only be described with low-level features. In order to define adequate geometric features, we follow recent investigations involving a multitude of geometric 3D and 2D features (29; 30). For each 3D point, respective features are based on the eigenvalues of the 3D structure tensor, geometric properties of the considered 3D neighborhood or geometric properties based on a 2D projection onto a horizontally oriented plane.

In total, a set of 21 low-level geometric features is thus calculated for each individual 3D point, and their distinctiveness is increased by considering the optimal neighborhood size which has been derived in the previous step.

2.3 Random Forest based Classification

For classification, we exploit a standard supervised scheme based on a Random Forest classifier (3) which represents a modern discriminative method and provides efficiency in case of a large amount of input data. Given a small set of training examples, we first ensure

that the number of training examples per class is equally distributed since an unbalanced distribution may have a detrimental effect on the training process (6). This is done by randomly selecting the same number of 1,000 training examples for each class.

3 Adaptation for Large-Scale Urban Point Cloud Classification

Recent investigations clearly show that optimal neighborhoods have a significant, beneficial impact on the classification results (30). However, the additional calculations cause a drastic increase in computational effort, particularly for optimal neighborhood size selection and feature extraction. Consequently, the described methodology is suited to process point clouds containing up to a few millions of 3D points. When considering huge point clouds at city-scale with possibly billions of points – which is the aim of recent effort in order to obtain an adequate 3D model of a whole city like Paris – an adaptation has to be introduced.

The adaptation described in this paper does not affect the quality of 3D scene analysis, but only the scalability of the methodology in order to process larger datasets. Specifically, it introduces a successive processing based on a sliding window function which is shifted in discrete steps and involves a small buffer in order to avoid discontinuities at its borders.

4 Datasets

Since we focus on the issue of urban 3D point cloud classification and want to facilitate an objective comparison to other methodologies, we consider two publicly available and labeled datasets representing densely sampled urban environments. Both datasets have been acquired in the city of Paris, France, via mobile laser scanning (MLS) systems:

Paris-rue-Madame database (26): This dataset has been acquired with the mobile laser scanning system L3D2 (8) equipped with a Velodyne HDL32. It contains 20 million points corresponding to a digitized street section with a length of approximately 160 m. A respective annotation has been conducted in a manually assisted way and includes both point-wise labels (26 different classes) and segmented objects (642 objects in total).

Paris-rue-Cassette database (22): This dataset has been acquired with the mobile laser scanning system STEREOPOLIS II (21) in January 2013. It currently contains 12 million points corresponding to a digitized street section with a length of approximately 200 m, but it will be extended to a length of 10 km. A manually assisted annotation includes both point-wise labels and segmented objects.

5 Conclusion

We present a methodology for semantic 3D scene analysis and its adaptation for huge point clouds with billions of 3D points. The methodology requires a higher computational effort which, in turn, is justified as it significantly improves the classification results in comparison to state-of-the-art approaches (30). The introduced adaptation overcomes this limitation and also allows a large-scale urban 3D scene analysis. A visual impression of the classification results for the Paris-rue-Madame database is shown in Fig. 2.

Acknowledgements The project was partially supported by KIT-GRACE, the Graduate School for Climate and Environment at the Karlsruhe Institute of Technology (KIT). Furthermore, a funding for a stay

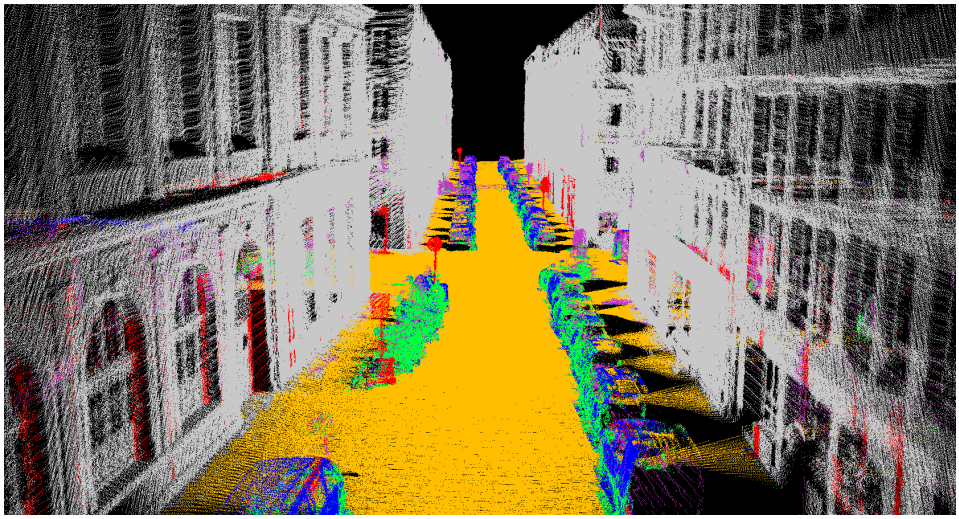


Fig. 2: Classified point cloud with assigned semantic labels (facade: gray, ground: brown, cars: blue, motorcycles: green, traffic signs: red, pedestrians: pink): The noisy appearance results from individual point classification.

abroad of the first author was provided by the Karlsruhe House of Young Scientists (KHYS) at KIT in order to support the collaboration.

References

[1] Boulch, A., Houllier, S., Marlet, R., Tournaire, O.: Semantizing complex 3D scenes using constrained attribute grammars. Computer Graphics Forum **32**(5), 33–42 (2013)

[2] Boyko, A., Funkhouser, T.: Extracting roads from dense point clouds in large scale urban environment. ISPRS Journal of Photogrammetry and Remote Sensing **66**(6), S2–S12 (2011)

[3] Breiman, L.: Random forests. Machine Learning **45**(1), 5–32 (2001)

[4] Bremer, M., Wichmann, V., Rutzinger, M.: Eigenvalue and graph-based object extraction from mobile laser scanning point clouds. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. II-5/W2, 55–60 (2013)

[5] Ceylan, D., Mitra, N.J., Zheng, Y., Pauly, M.: Coupled structure-from-motion and 3D symmetry detection for urban facades. ACM Transactions on Graphics **33**(1), 2:1–15 (2014)

[6] Criminisi, A., Shotton, J.: Decision forests for computer vision and medical image analysis. Advances in Computer Vision and Pattern Recognition, Springer, London, UK (2013)

[7] Demantké, J., Mallet, C., David, N., Vallet, B.: Dimensionality based scale selection in 3D lidar point clouds. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. XXXVIII-5/W12, 97–102 (2011)

[8] Goulette, F., Nashashibi, F., Abuhadrous, I., Ammoun, S., Laurgeau, C.: An integrated on-board laser range sensing system for on-the-way city and road modelling. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. XXXVI-1 (2006)

[9] Guan, H., Li, J., Yu, Y., Wang, C., Chapman, M., Yang, B.: Using mobile laser scanning data for automated extraction of road markings. ISPRS Journal of Photogrammetry and Remote Sensing **87**, 93–107 (2014)
[10] Hebert, M., Bagnell, J.A., Bajracharya, M., Daniilidis, K., Matthies, L.H., Mianzo, L., Navarro-Serment, L., Shi, J., Wellfare, M.: Semantic perception for ground robotics. In: Karlsen, R.E., Gage, D.W., Shoemaker, C.M., Gerhart, G.R. (eds.) Unmanned Systems Technology XIV, SPIE Proceedings **8387**, 83870Y:1–12 (2012)
[11] Hu, H., Munoz, D., Bagnell, J.A., Hebert, M.: Efficient 3-D scene analysis from streaming data. In: Proceedings of the IEEE International Conference on Robotics and Automation, 2297–2304 (2013)
[12] Kim, H.B., Sohn, G.: Random forests based multiple classifier system for powerline scene classification. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. XXXVIII-5/W12, 253–258 (2011)
[13] Lafarge, F., Alliez, P.: Surface reconstruction through point set structuring. Computer Graphics Forum **32**(2), 225–234 (2013)
[14] Lafarge, F., Mallet, C.: Creating large-scale city models from 3D-point clouds: a robust approach with hybrid representation. International Journal of Computer Vision **99**(1), 69–85 (2012)
[15] Lalonde, J.-F., Unnikrishnan, R., Vandapel, N., Hebert, M.: Scale selection for classification of point-sampled 3D surfaces. In: Proceedings of the International Conference on 3-D Digital Imaging and Modeling, 285–292 (2005)
[16] Mitra, N.J., Nguyen, A.: Estimating surface normals in noisy point cloud data. In: Proceedings of the Annual Symposium on Computational Geometry, 322–328 (2003)
[17] Munoz, D., Bagnell, J.A., Vandapel, N., Hebert, M.: Contextual classification with functional max-margin Markov networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 975–982 (2009)
[18] Musialski, P., Wonka, P., Aliaga, D.G., Wimmer, M., van Gool, L., Purgathofer, W.: A survey of urban reconstruction. Computer Graphics Forum **32**(6), 146–177 (2013)
[19] Niemeyer, J., Rottensteiner, F., Soergel, U.: Contextual classification of lidar data and building object detection in urban areas. ISPRS Journal of Photogrammetry and Remote Sensing **87**, 152–165 (2014)
[20] Nüchter, A., Lingemann, K.: Robotic 3D Scan Repository. Jacobs University Bremen gGmbH and University of Osnabrück (2011) Available at http://kos.informatik.uni-osnabrueck.de/3Dscans/ (last access: 15 May 2014)
[21] Paparoditis, N., Papelard, J.-P., Cannelle, B., Devaux, A., Soheilian, B., David, N., Houzay, E.: Stereopolis II: a multi-purpose and multi-sensor 3D mobile mapping system for street visualisation and 3D metrology. Revue Française de Photogrammétrie et de Télédétection **200**, 69–79 (2012)
[22] Paparoditis, N., Vallet, B., Marcotegui, B., Serna, A.: IQmulus & TerraMobilita Contest – Analysis of mobile laser scans (MLS) in dense urban environments. MATIS Laboratory, French National Mapping Agency (IGN) and Center for Mathematical Morphology (CMM), MINES ParisTech (2014) Available at http://data.ign.fr/benchmarks/UrbanAnalysis/ (last access: 15 May 2014)
[23] Pauly, M., Keiser, R., Gross, M.: Multi-scale feature extraction on point-sampled surfaces. Computer Graphics Forum **22**(3), 281–289 (2003)
[24] Pu, S., Rutzinger, M., Vosselman, G., Oude Elberink, S.: Recognizing basic structures from mobile laser scanning data for road inventory studies. ISPRS Journal of Photogrammetry and Remote Sensing **66**(6), S28–S39 (2011)
[25] Serna, A., Marcotegui, B.: Urban accessibility diagnosis from mobile laser scanning data. ISPRS Journal of Photogrammetry and Remote Sensing **84**, 23–32 (2013)
[26] Serna, A., Marcotegui, B., Goulette, F., Deschaud, J.-E.: Paris-rue-Madame database: a 3D mobile laser scanner dataset for benchmarking urban detection, segmentation and classification methods. In: Proceedings of the International Conference on Pattern Recognition Applications and Methods (2014)

[27] Vanegas, C.A., Aliaga, D.G., Benes, B.: Automatic extraction of manhattan-world building masses from 3D laser range scans. IEEE Transactions on Visualization and Computer Graphics **18**(10), 1627–1637 (2012)
[28] Velizhev, A., Shapovalov, R., Schindler, K.: Implicit shape models for object detection in 3D point clouds. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. I-3, 179–184 (2012)
[29] Weinmann, M., Jutzi, B., Mallet, C.: Feature relevance assessment for the semantic interpretation of 3D point cloud data. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. II-5/W2, 313–318 (2013)
[30] Weinmann, M., Jutzi, B., Mallet, C.: Semantic 3D scene interpretation: a framework combining optimal neighborhood size selection with relevant features. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences (2014)
[31] Wurm, K.M., Kretzschmar, H., Kümmerle, R., Stachniss, C., Burgard, W.: Identifying vegetation from laser data in structured outdoor environments. Robotics and Autonomous Systems **62**(5), 675–684 (2014)
[32] Zheng, Q., Sharf, A., Wan, G., Li, Y., Mitra, N.J., Cohen-Or, D., Chen, B.: Non-local scan consolidation for 3D urban scenes. ACM Transactions on Graphics **29**(4), 94:1–9 (2010)
[33] Zhou, Q.-Y., Neumann, U.: Complete residential urban area reconstruction from dense aerial LiDAR point clouds. Graphical Models **75**(3), 118–125 (2013)
[34] Zhou, L., Vosselman, G.: Mapping curbstones in airborne and mobile laser scanning data. International Journal of Applied Earth Observation and Geoinformation **18**(1), 293–304 (2012)

Towards Medial Axis-based point cloud simplification for LiDAR point clouds

Ravi Peters

Abstract Current methods from GIS practice to simplify large airborne LiDAR point clouds are unaware of significant surface characteristics and work under a 2.5D assumption of the sampled surface. I investigate an alternative simplification method based on the Medial Axis Transform that does not suffer from these shortcomings. An important precondition for this method to work successfully is the ability to construct an approximation of the Medial Axis Transform that is robust to the noise that is typically present in airborne LiDAR point clouds. Therefore I propose an adaptation of an existing algorithm to approximate the Medial Axis Transform. Preliminary results indicate that this significantly reduces effects of noise in the resulting approximation of the Medial Axis Transform.

1 Introduction

With recent advances in remote sensing technologies such as airborne LiDAR, photogrammetry and multi-beam echo-sounding we are able to acquire geometric data of the Earth's surface in unprecedented quantities and accuracy. The resulting elevation models (point clouds) have a broad range of applications such as flood modeling, dike monitoring, crisis management and city modeling (12). One of the first LiDAR-based point clouds with national coverage was the Dutch *Actueel Hoogtebestand Nederland* (AHN[1]). Its current iteration (AHN2) offers an average point density of 8 points per square meter, totaling up to hundreds of billions point measurements for the whole of the Netherlands. Furthermore, municipalities such as Rotterdam are maintaining their own LiDAR datasets with average point densities of over 30 points per square meter. Such datasets have big potential for the aforementioned applications.

However, it proves challenging to efficiently manage and process huge point clouds such as AHN2. Two not entirely unrelated issues can be identified here. First, because they require so much storage, they do not fit in a computer's internal memory (8). As a result, many of the conventional software tools are very inefficient with large datasets, and might even refuse to finish processing entirely. Second, in GIS, an elevation model is commonly treated as a 2.5D surface such as a raster or a TIN. While this alleviates memory requirements and simplifies computation, it comes at the price of a significant loss of information because 2.5D data structures are simply unable to represent the 3D information present in modern LiDAR-based point clouds. Consequently, the information

Ravi Peters
Delft University of Technology, Faculty of Architecture and The Built Environment, Jaffalaan 9, 2628 BX, Delft, The Netherlands e-mail: r.y.peters@tudelft.nl

[1] http://www.ahn.nl

that is lost during conversion from precise and fully 3D point clouds to (approximate) 2.5D surfaces, is no longer available for any subsequent processing.

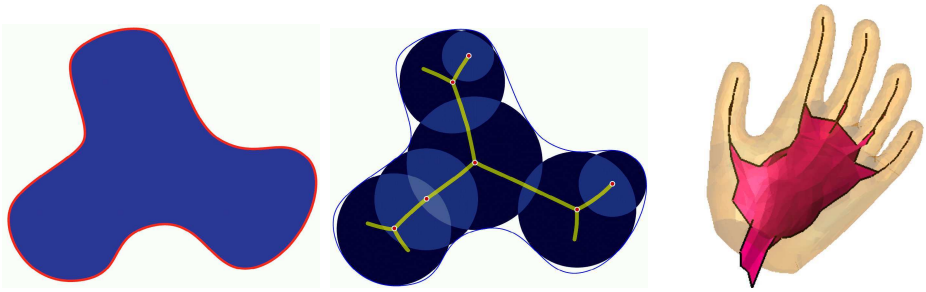
2 The simplification operator in practice

Simplification is the process of reducing the number of points in a point cloud, preferably while preserving the shape of the sampled surface as well as possible. This is often one of the first steps in a GIS processing pipeline, since it will also reduce the amount of CPU time required to execute the remaining part of the pipeline. Standard approaches implemented in GIS packages, such as ArcGIS, include random or n^{th} point selection, constraining the number of points contained by a moving rectangular window, and variants of TIN decimation (similar to Lee (9)). The first two are, because of their simplicity, quite fast to execute, but they treat all points equally, because the chance of removal is equal for every point. Yet, some points are more valuable than others, that is particularly true for points that describe significant surface features (such as building corners or the edges of a landform). The simplification operator should therefore attempt to preserve these points, and prioritize the removal of points that are relatively unimportant (for example in large planar regions). Simplification based on TIN decimation actually attempts this, albeit from a 2.5D perspective. In TIN decimation, points are assigned an importance value based on the vertical variation of their local neighbourhood and the least important points are the most likely to be removed. Although this method is much more expensive than the earlier two, it gives much better results, typically removing more points while staying closer to the original surface (3). Unfortunately, as noted by Crawford (3), this method should only be used for LiDAR point clouds with 1^{st}-return surfaces (thus ignoring any following reflections for the same laser pulse due to penetrable surfaces such as trees or glass). In other words: when the point cloud features actual 3D geometries, problems arise. This is due to the planar triangulation that takes place during TIN decimation, which assumes that the input point cloud only samples 2.5D shapes. This demonstrates the need for practical point cloud simplification methods that treat the input point clouds as truly 3D.

3 Medial Axis-based simplification of point clouds

My hypothesis is that the Medial Axis Transform (MAT) enables true 3D analysis and generalization of LiDAR point clouds in a practical manner. The MAT (2) is a shape descriptor that compactly represents both the geometry and the topology of shapes (see Figure 1). It is formally defined as the set of maximally inscribed (or *medial*) balls of a shape. The centers of these balls make up a skeleton structure. The MAT consists of both this skeleton and the medial balls (see Figures 1a and 1b). Succesfull Applications of the MAT include surface reconstruction (1; 5), shape simplification (13) and feature-aware mesh (7). Matuk (11) and Dakowicz and Gold (4) have shown that the 2D MAT can be succesfully applied for 2.5D terrain simplification.

Ma et al (10) introduced the shrinking ball algorithm to approximate a point approximation of the MAT from an oriented input point cloud. For each sample point the algorithm finds an empty maximal tangent ball is by iteratively reducing its radius using nearest neighbor queries (see Figure 4a). Compared to earlier (Voronoi-based) algorithms (e.g. Amenta et al (1) and Dey and Zhao (6)) it is simple, fast, robust in practice, and easy to parallelize. This makes it a good choice for approximating the MAT of large LiDAR point clouds. Ma et al (10) also demonstrated that their method can be employed for feature-aware simplification of carefully sampled input point clouds. This is based on the *local feature size* (LFS) measure, which is defined for every point in the input point cloud as the shortest distance



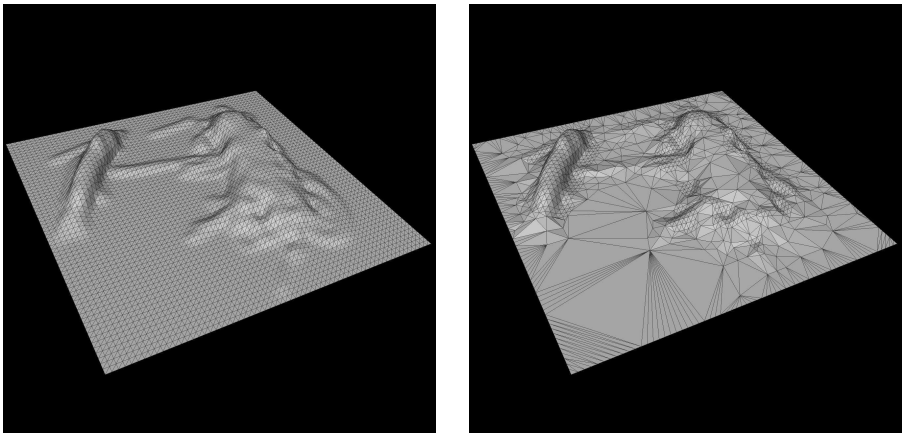
(a) A 2D shape

(b) Its (inner) medial axis and some medial balls.

(c) The 3D MAT of a surface model of a hand consists of faces, edges and vertices (1).

Fig. 1: The Medial Axis Transform

to the MAT (see Figure 3a). The main idea is to preserve points with a low LFS, as these are close to the MAT, which generally indicates high curvature. Figure 2 demonstrates this.



(a) Before simplification.

(b) After simplification.

Fig. 2: Example of LFS-based simplification on synthetic dataset. The point clouds are triangulated for illustrative purposes only.

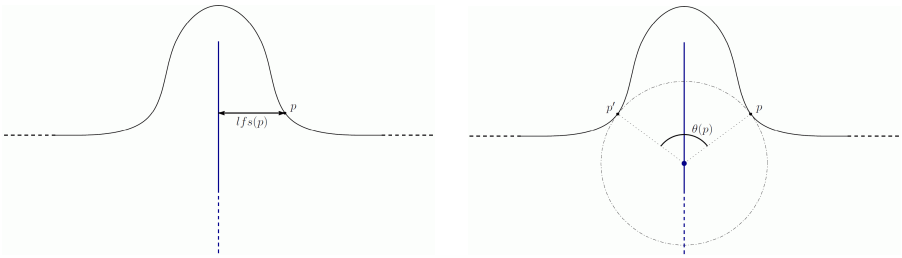
4 Handling noise

The MAT is notorious for its sensitivity to small perturbations in the object surface which greatly affects the effectiveness of application such as LFS-based simplification which requires a sufficiently dense representation of the MAT that is to some degree insensitive to noise. Airborne LiDAR point clouds on the other hand typically suffer from relatively low-density sampling and significant noise.

Existing methods to deal with noisy input points (e.g. the methods used in Amenta et al (1)) all *remove* noisy MAT points. The resulting MAT approximation can thus become very sparse for higher input noise levels. To deal with this problem I propose an adaptation of the original shrinking ball algorithm. It is based on analyzing the sequence of shrinking

balls that is generated for every input point (see Figure 4a). I make the key observation that in these sequences a noisy ball is often preceded by a ball that is approximately medial and unaffected by noise (compare balls 4 and 5 in Figure 4b). It thus comes down to picking the best candidate from the sequence of shrinking balls that is generated for every input point. In order to choose this *best* ball I use the *separation angle*, i.e. the angle that is formed by the vectors between a medial point and its two corresponding surface points, as illustrated in Figure 3b. Balls with a large separation are generally considered less likely to be influenced by noise (see e.g. (1)). Whenever the separation angle reaches below a threshold or when the difference in separation angle between two consecutive shrinking balls reaches above a second threshold, this ball is ignored and the ball-shrinking process is stopped. Figure 5 illustrates the effect of this adaptation of the shrinking ball algorithm. The resulting MAT approximation is generally much better delineated in the presence of noise.

Do note that these are preliminary results based on ongoing research.



(a) local feature size (b) separation angle

Fig. 3: Geometric measures on the MAT. MAT in blue, object surface in black.

Acknowledgements This research was financially supported by the Dutch Technology Foundation STW, which is part of the Netherlands Organisation for Scientific Research (NWO), and which is partly funded by the Ministry of Economic Affairs (project code: 12217);

References

[1] Amenta N, Choi S, Kolluri RK (2001) The power crust. In: Proceedings of the sixth ACM symposium on Solid modeling and applications, ACM, New York, NY, USA, SMA '01, pp 249–266

[2] Blum H (1967) A transformation for extracting new descriptors of shape. Models for the perception of speech and visual form 19(5):362–380

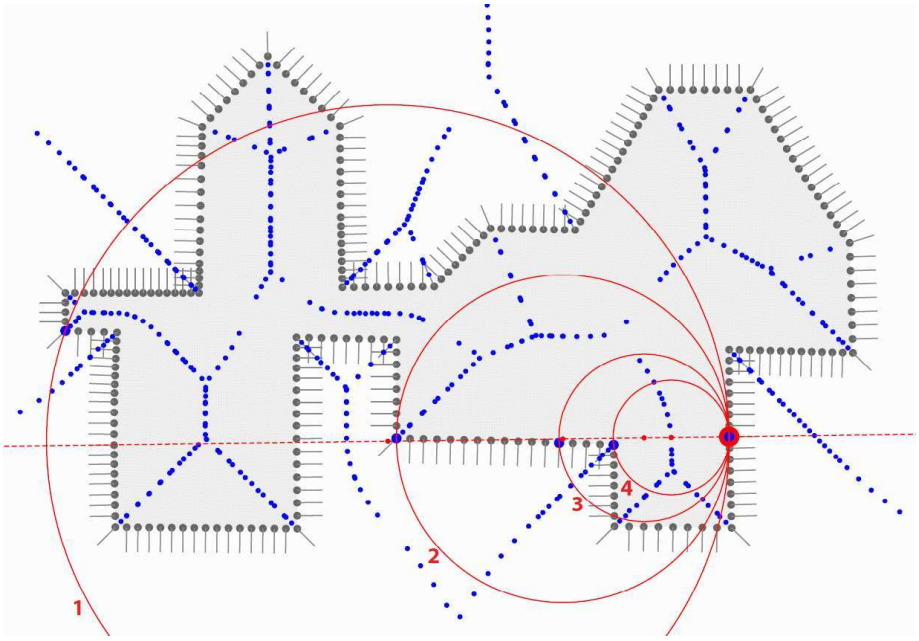
[3] Crawford C (2013) Assessing the use of point thinning on airborne lidar for dem production. Presentation slides, European Lidar Mapping Forum 2013, Amsterdam

[4] Dakowicz M, Gold C (2003) Extracting meaningful slopes from terrain contours. International Journal of Computational Geometry & Applications 13(04):339–357

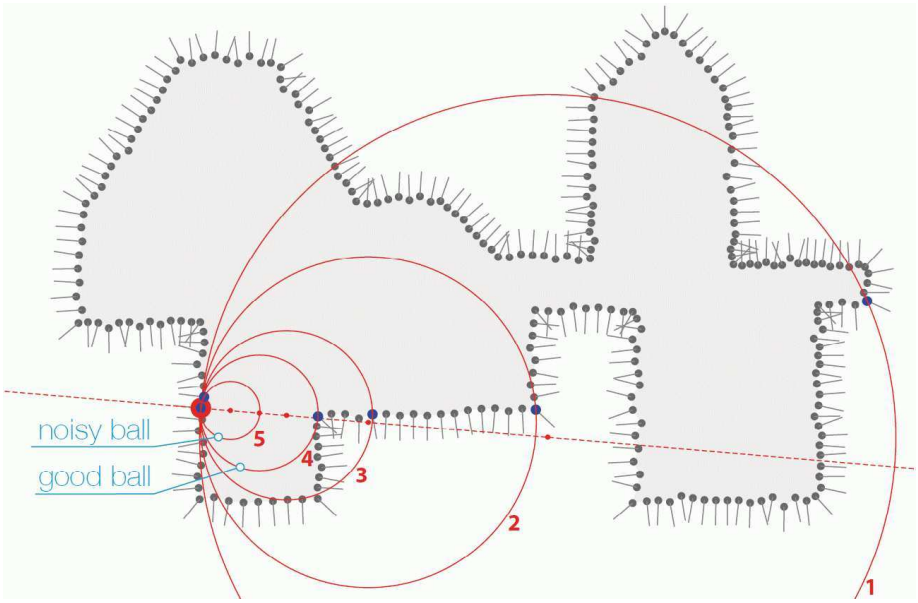
[5] Dey TK, Goswami S (2003) Tight cocone: a water-tight surface reconstructor. In: Proceedings of the eighth ACM symposium on Solid modeling and applications, ACM, New York, NY, USA, SM '03, pp 127–134

[6] Dey TK, Zhao W (2004) Approximate medial axis as a voronoi subcomplex. Computer-Aided Design 36(2):195–202

[7] Dey TK, Giesen J, Hudson J (2001) Decimating samples for mesh simplification. In: Proc. 13th Canadian Conf. Comput. Geom, pp 85–88



(a) Approximating the MAT with a noise-free input. Resulting approximation shown with blue points.



(b) Approximating the MAT with a noisy input.

Fig. 4: Sequence of shrinking balls in 2D for a single input point without noise (a) and with noise (b)
.

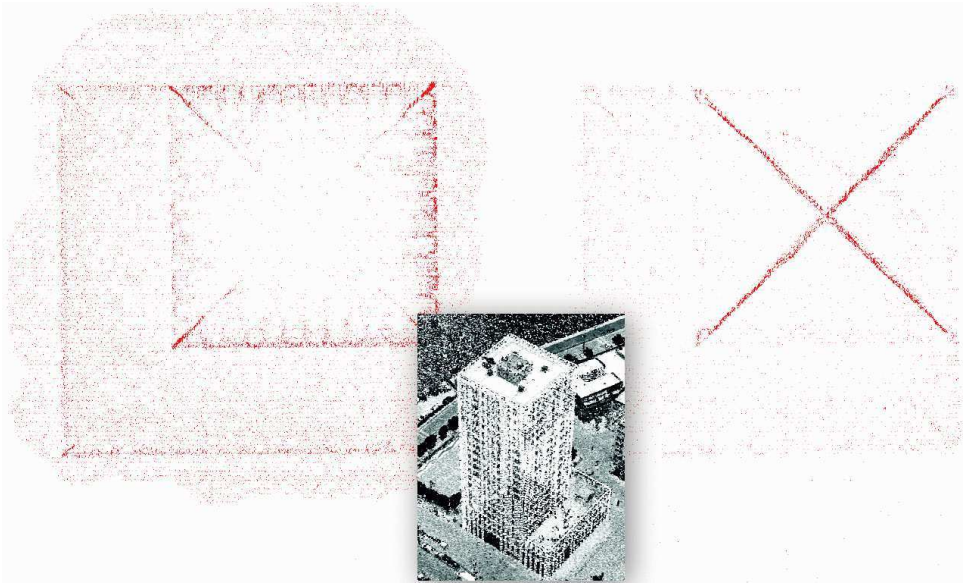


Fig. 5: Results of interior MAT approximation of a building from an airborne LiDAR dataset from the municipality of Rotterdam (middle) with the unmodified shrinking ball algorithm (left) and the modified shrinking ball algorithm (right). Shown are top views.

[8] Isenburg M, Liu Y, Shewchuk JR, Snoeyink J (2006) Streaming computation of Delaunay triangulations. ACM Transactions on Graphics 25(3):1049–1056
[9] Lee J (1989) A drop heuristic conversion method for extracting irregular network for digital elevation models. ASPRS/ACSM
[10] Ma J, Bae SW, Choi S (2012) 3d medial axis point approximation using nearest neighbors and the normal field. The Visual Computer 28(1):7–19
[11] Matuk K (2006) Feature-based terrain model simplification. PhD thesis, Hong Kong Polytechnic University
[12] Snyder GI (2013) The benefits of improved national elevation data. Photogrammetric Engineering and Remote Sensing 79(2)
[13] Tam R, Heidrich W (2003) Shape simplification based on the medial axis transform. In: Visualization, 2003. VIS 2003. IEEE, IEEE, pp 481–488

Voxel based scalable registration of laser scanned point cloud data by neighbourhood voting

Jinhu Wang, Roderik Lindenbergh and Massimo Menenti

Abstract We propose a novel technique for automatic scalable registration of 3D point cloud data acquired by laser scanners based on voxels. This algorithm is composed of two steps. Firstly, coarse registration. This step is achieved by first resampling the two point cloud data into 3D voxels of a uniform size. Then the second order moment features of the voxel are abstracted based on the points within the voxel, and afterwards neighborhood voting of the voxel is performed to strengthen the feature. Based on the correlation of the final voxel features of the two point clouds, correspondences between the two voxels are found. Consecutively the random sampling consensus (RANSAC) algorithm is employed to remove outlying correspondences until a coarse alignment can be made. Secondly, a fine registration is acquired in the final step by running the Iterative Closest Points (ICP) algorithm with just a few iterations.

Key words: Scalable registration; Neighborhood voting; Tensor; Point cloud

1 Introduction

Laser scanners provide an efficient solution to acquire high accuracy and dense point cloud data of objects. This enables for rapid modelling for 3D reconstruction, architecture, tunnels and civil engineering. However new laser scanners deliver up to 1,000,000 3D points per second, which makes the processing of the data time consuming, which many data is highly redundant. In (**?**), an octree was employed to organize point clouds to perform efficient processing, and also algorithms are developed based on such octree organization. However the features derived are point based, which is still computational expensive. Also the down sampling method used is brutal while some critical corners or edge points, which are very important features, may not be retained. While in (3; 1), point cloud is sampled as voxels and pole-like objects are detected. The advantages of these approaches are that the information and location of the key points are not disregarded. Still, the crucial issue about voxel based 3D point cloud processing is that for the point cloud data, different subdivision schemes will have different voxel structures and the possibility of strong voxel based feature abstraction from within single voxels is very limited. To improve the feature

Jinhu Wang
Delft University of Technology, Delft 2628CN, the Netherlands, e-mail: jinhu.wang@tudelft.nl

Roderik Lindelbergh
Delft University of Technology, Delft 2628CN, the Netherlands, e-mail: r.c.lindenbergh@tudelft.nl

Massimo Menenti
Delft University of Technology, Delft 2628CN, the Netherlands, e-mail: m.menenti@tudelft.nl

abstraction procedure, we propose that to take voxel neighbors into account. In (1; 4), the k-nearest neighbors (KNN) algorithms were used to obtain neighborhood points, consecutively principal component analysis (PCA) was used to computed the Eigen features from points that within the neighborhood. The relationship between the neighborhood points are then acquired by tensor voting. However, this method is computational expensive and is not appropriate to deployed for huge population of point clouds data as acquired by the new laser scanners. The method introduced in this paper is a voxel based approach and can cope with the large point cloud data sets which is demonstrated on a real registration scenario.

2 Methodology

The registration method introduced in this paper consists of two steps, i.e. coarse registration and fine registration. In the coarse alignment step, the two point cloud data were first sampled as voxels with a larger voxel size, then the Eigen feature of every voxels are computed based on the points in the voxel space. As shown in Figure 2, the three Eigen values are determined by decompose the points in the voxel with PCA, and the arrows denote the directions of the three Eigen vectors.

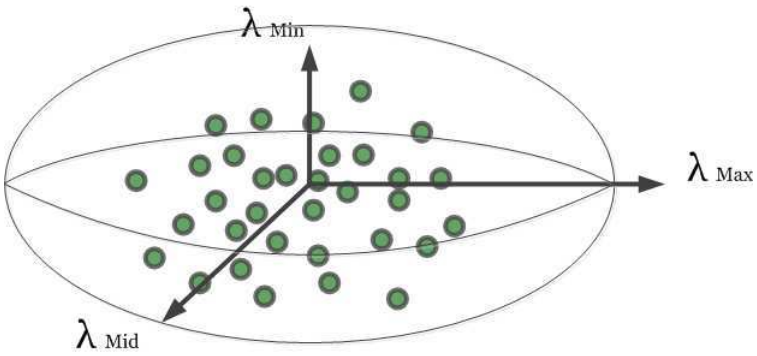


Fig. 1: Eigen features of the points within voxel cell

To make the voxel attributes distinguishable, a neighborhood voting scheme is performed to enhance the features. For a specific voxel cell, defined as a votee, its neighbor cells, which are categorized into three classes, i.e. face, edge and vertex neighbors are all defined as voters. As is shown in Figure 2, (1) and (2),(3),(4) and (5),(6) depict the face, edge and vertex neighboring cells respectively. The voxel cell in red is the defined votee cell, while all the rest cells are labelled by an index number are the voter cells.

Each of the voter cells Eigen features contributes to that of the votee. So the contribution of each voter to the votee is integrated with regard to a total 26 voter cells by neighborhood voting scheme as described in Figure 3. It depicts the voting process from the voter, which is a vertex neighbor cell with index 19, to the votee. The Eigen value of the voter in each direction gives a certain amount of token to the votee. After voting procedure, the votee obtained the neighborhood voted Eigen features, which then were imported to the correspondences searching procedure.

Correspondences are searched afterwards with regard to the abstracted Eigen features. Outliers are rejected with RANSAC, then the transformation is estimated. A smaller voxel size is used to perform the above processing again until the coarse registration is good enough to perform fine registration. In the fine registration, the well-known ICP algorithm is used to acquire the final transformation.

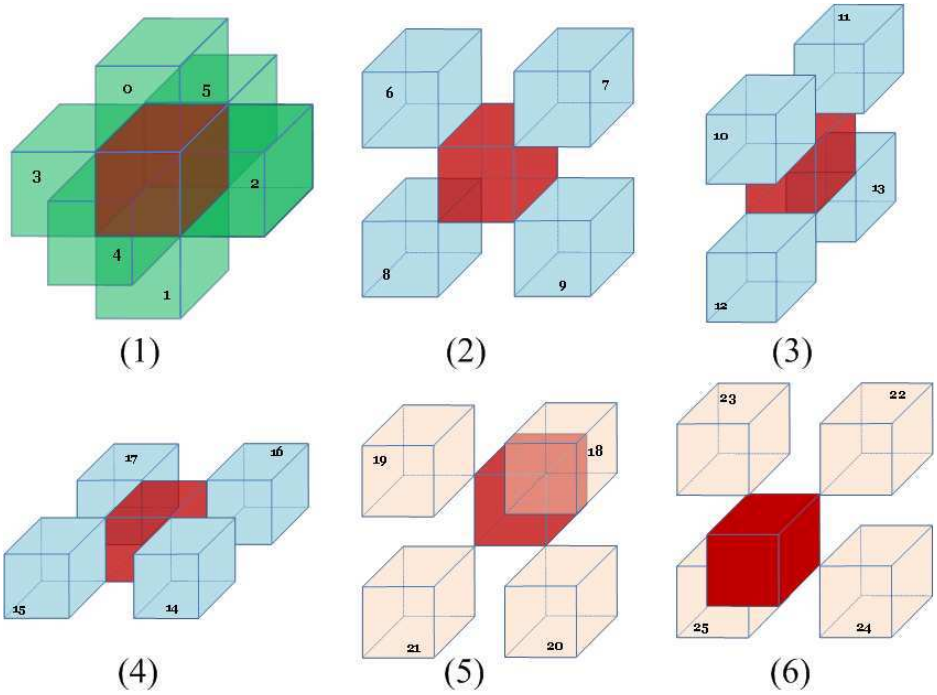


Fig. 2: Votee and its categorized neighbouring cells

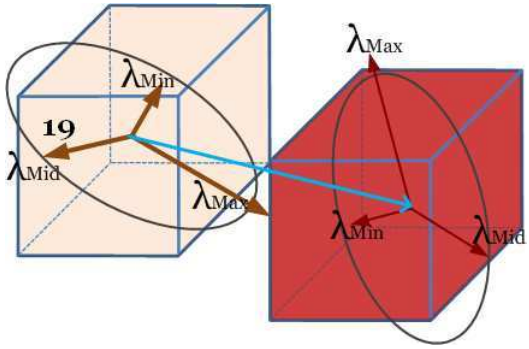


Fig. 3: Neighbourhood voting from a vertex neighbour to the votee

3 Initial Results

The voting algorithm is implemented using C++ and the visualization is in OpenGL. The test point cloud data sets are a laboratory scene scanned by Photon 20/120 and have a population 407,780 points. With a random transformation matrix multiplied to the original point cloud generated the second testing point cloud data. The data sets are shown in Figure 4.

Then voxels of the two data sets are then generated with an initial voxel cell size of 0.3 meter. As shown in Figure 5. Consecutively the correspondences are searched based on the similarity of the Eigen features abstracted previously. As shown in Figure 6. This test shows that more than 70 percent of the correspondences are truly matched, so the RANSAC algorithm could be employed to remove the outliers. Then the initial transform could be performed.

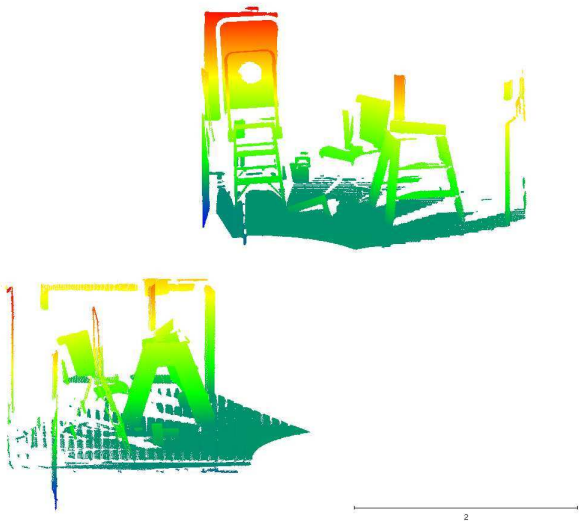


Fig. 4: Original point cloud data sets for registration

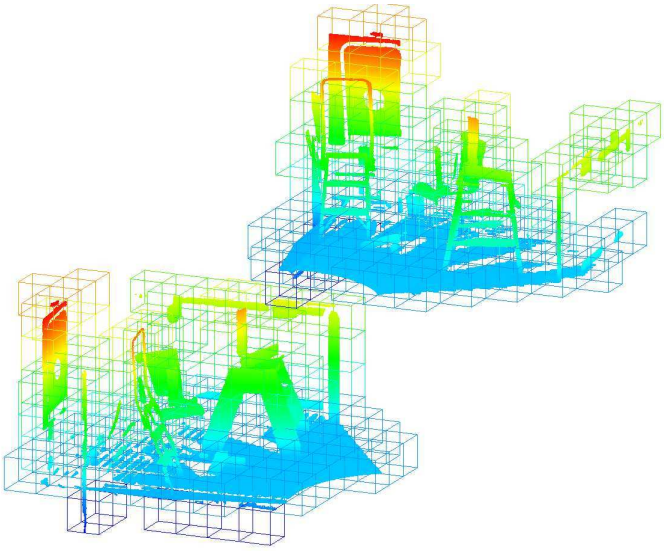


Fig. 5: Voxels generated from the two point cloud data sets

After this transformation, new voxels are generated with a smaller voxel cell size and obtain a finer alignment. A few iterations later the point clouds are almost fined aligned and then the ICP is employed to acquire the fine registration.

4 Conclusion

The method in this paper efficiently performs the coarse registration and act as an initiation of the ICP algorithm for a fine registration. And the method is validated with a sample point cloud data which verified its feasibility. Also since the method only taking the voxels into account, the algorithm is computational cheap and runs smoothly on the desktop PC.

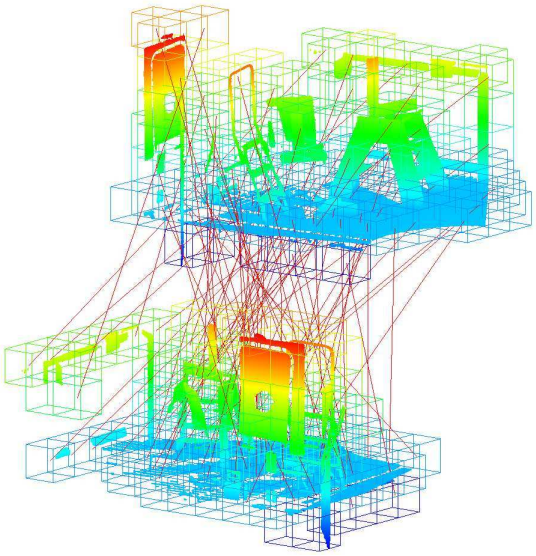


Fig. 6: Correspondences of the two voxels

References

[1] Cabo C, Ordoñez C, García-Cortés S, Martínez J. An algorithm for automatic detection of pole-like street furniture objects from Mobile Laser Scanner point clouds. ISPRS Hournal of Photogrammetry and Remote Sensing, 87:47-56

[2] Elseberg, Jan and Borrmann, Dorit and Nüchter, Andreas. One billion points in the cloud: an octree for efficient processing of 3D laser scans. ISPRS Journal of Photogrammetry and Remote Sensing, 10.004

[3] Payeur Pierre. A computational technique for free space localization in 3-D multiresolution probabilistic environment models. IEEE Transactions on Instrumentation and Measurement. 55:1734–1746

[4] Tang Chi Keung, Medioni Garard. Inference of integrated surface, curve, and junction descriptions from sparse 3D data. IEEE Transactions on Pattern Analysis and Machine Intelligence. 20:1206–1223

Advancing a geospatial framework to the MapReduce model

Roberto Giachetta

Abstract In recent years, cloud computing has reached many areas of computer science including geographic and remote sensing information systems. However distributed data processing solutions have primarily been focused on processing simple structured documents, rather than complex geospatial data. Hence, moving current algorithms and data management to cloud architecture may require a great deal of effort.
This article describes the evolution of the AEGIS spatio-temporal framework towards cloud based spatial data processing, based on the mainstream MapReduce paradigm by using the Apache Hadoop implementation. The architecture of the framework enhances both data storage capabilities and the processing model allowing the previously implemented algorithms to be easily adapted to distributed execution without the need of any transformation. Furthermore, extensions allow the processing of complex geospatial data including remotely sensed imagery in the Hadoop environment.

1 Introduction

In recent years cloud computing has become an active reseach field promoting a paradigm shift for organizations and businesses to use cloud services for their information infrastructure. Cloud computing presents virtually unlimited possibilities for data analysis, which is also required due to data sets becoming enormously large and complex (usually referred as *Big data*) (1).

Many concepts and systems have been developed for data processing in the cloud, most notably the *MapReduce* model (2), and its open-source implementation, the *Apache Hadoop* framework (3), which have become the industry standard. However, the original aim of the MapReduce paradigm was to process simple text documents, thus the implementation of complex algorithms and the management of heterogeneous data structures may become an overwhelming task. To counteract this, several extensions and toolkits have been introduced that operate over the Hadoop platform enabling a wide range of data management, mining and analysis possibilities (4; 5).

In terms of geographical information systems (GIS) and remotely sensed image analysis, the new paradigms have already been successfully applied in multiple cases, leading to frameworks specialized for spatial data storage and analysis. There have also been some experiments with remotely sensed image processing. However, all approaches require to

Roberto Giachetta[1,2]
[1]Dept. of Software Technology and Methodology, Eötvös Loránd University, Budapest, Hungary
[2]Institute of Geodesy, Cartography and Remote Sensing, Budapest, Hungary,
e-mail: groberto@inf.elte.hu

adapt the processing workflow to the specific environment, with the architecture and execution model in mind. In contrary, most spatial data analysis processes performed at organizations, such as the *Institute of Geodesy, Cartography and Remote Sensing (FÖMI)*, have their evolved workflows using multiple (proprietary or open-source) software and GIS expertise (6).

In a joint work between FÖMI and the *Eötvös Loránd University (ELTE), Faculty of Informatics*, a spatial and remote sensing data management and processing toolkit is developed under the name *AEGIS* (7), which is designed to be adaptable to multiple architectures. Certainly, advancing this framework to perform operations using MapReduce model requires several considerations, but the extension of the operation execution environment allows reuse of previously implemented operations without any modification. The implementation of this extension is currently in progress, but our early results are promising.

2 Related work

In the past decade, geographical information science has gained a significant role due to the spread of GPS localization, navigation systems and the publication of geographical data via Internet. As cloud computing became prominent, there have been several steps made to advance GIS to the cloud, which has led to *spatial cloud computing* (8). Several solutions have been researched, mostly focused on how the existing Hadoop framework can be improved. Cary et al. have shown that the Hadoop environment can be used for spatial data processing, even for building spatial indices using MapReduce processes (9).

Complete software packages have also been developed and published. For example, *SpatialHadoop* (10) extends functionality with efficient spatial data storage, spatial indexing, and spatial query support. By separating local from global indices, efficient query execution can be performed with MapReduce. *Hadoop-GIS* (11) offers complete spatial data-warehouse solution over *Hive* (12), and uses spatial partitioning to distribute data. Again spatial indices are built up (mainly using bulk R* trees), which are further used by spatial queries. Both solutions are primarily designed to enable efficient spatial queries of vector data.

In the context of raster data, image processing has also been successfully applied to the MapReduce model. Golpayegani and Halem showed that implementing image analysis operations in Hadoop can be performed in a straightforward way (13). By using a cell based allocation of images, the *Map* function is used both as a locator and initial processor, whilst the *Reduce* stage performes result summation. A different approach to distributed image processing is presented by Alonso-Calvo et al. (14). In this study, images are transformed to region-based graph representation allowing operations to work on the distributed regions. Although this method enables easy parallel execution of most image processing operations, the region based transformation may cause loss of information. Zhang et al. focus on the problem of managing images in the cloud file system, by extending the capabilities of the distribution system (15).

Some studies deal with the efficiency of data processing in the cloud. *Apache Spark* is a data processing engine built ont top of Hadoop (16). It supports cyclic data flow and in-memory computing, by relying on read-only memory caches, known as *resilient distributed datasets*. This approach is capable of significantly speeding up scientific data processing, as seen in case of the *Shark* data analytics engine (17). Unfortunately it does not exactly suit the requirements of image processing, but the basic idea can be incorporated.

3 The AEGIS framework

The AEGIS framework is a geospatial toolkit initially developed for education and research goals, and is currently used as a learning tool for computer science students at ELTE. It is based on well-known standards of the *Open Geospatial Consortium (OGC)*, and state of the art programming methodologies. The framework is implemented using *.NET/Mono Frameworks* to exploit the wide possibilities and the simple usage of this development platform.

The framework has a component based architecture in order to be as adaptable and customizable as possible. Components can be added and recognized by the system even during runtime. AEGIS supports both vector geometry and remotely sensed images based on the well known *Simple Feature Access* standard. A wide variety of source formats is supported for storage of spatial data. These sources are represented as data sets known as *entities* in the framework. Entities may refer to a single object, or may be collections consisting of multiple parts (child entities), allowing handling of partial or distributed datasets. An internal revision control system is also provided for maintaining information on alteration of data and reverting changes.

The processing module uses a meta descriptor system to support runtime management and extension. New operations may be added, or existing operations can be extended to support new kind of input data. Operations may also be provided in multiple versions, when adding new features or altering functionality. The execution of the operation is performed using an *operations engine*, which manages all operation metadata, and parallel execution. The operations engine can be implemented to any environment able to perform the execution of operations.

The schema of data and operation management can be seen in Figure 1.

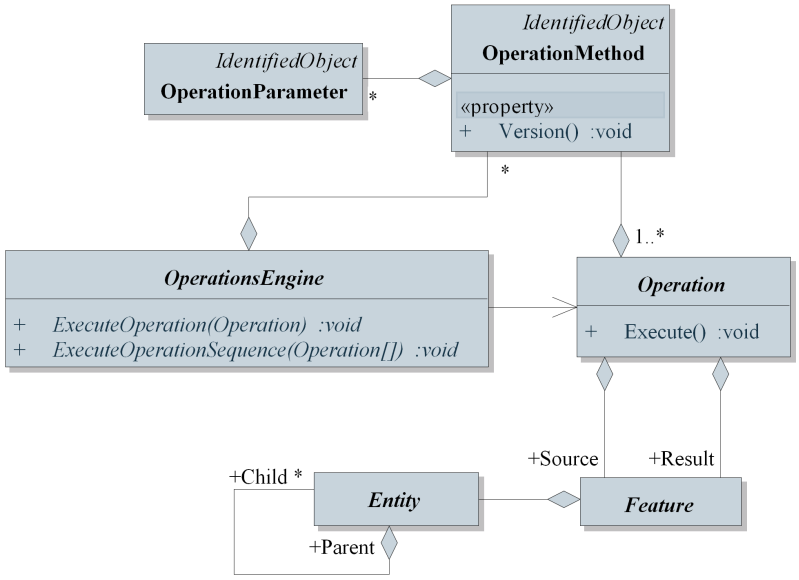


Fig. 1: The processing schema of the AEGIS framework

As data management, entities and operations engines are all interchangeable, the advancement of the framework to use Hadoop as storage and execution platform, can be performed by only modifying these components. Adaptation is facilitated by the streaming capabilities of Hadoop, and the easy file system access using REST. However to benefit

from the strengthen parts of the distributed environment, many extensions should be added, mainly to data management.

4 Data management

To enable spatial data processing in the cloud, well designed data management is required, as spatial features may come either in vector or raster format, and may contain descriptive information (metadata).

As seen in Section 2, to enhance the performance of spatial queries, the data can be distributed in a variety of ways. For most spatial operations, distribution based on spatial extent can drastically enhance performance of spatial operations. For efficient access, indices can be constructed at node level and system level (18). Parts of the data can be stored as files (preferably in a simple format). From the system's point of view, these parts can still be considered as a single feature, as they can be modeled as child entities of a single (parent) entity. The parent entity can also be considered as a spatial index structure mapping the partitions to their original position within the data.

However, different operations may prefer a different approach to distribution. For example, in the case of remotely sensed imagery, a clustering operation usually works on the multispectral space of the image instead the original domain. Thus, the data should be partitioned based on the position of spectral values instead of their spatial position in the image. Another concern is that queries may also be performed based on metadata, which is not in accordance with the spatial or spectral data based distribution. To overcome this, feature metadata should be separately stored and indexed from the geometry, isolating nodes that store vector and raster content, and nodes that store metadata.

One can think of several other methods to distribute the original data based on different usage criteria. This results in the *distribution strategy* becoming an argument in the process, which can be defined based on the knowledge of possible future operations. Even multiple strategies can be applied to the same data, resulting in redundant storage, but more efficient execution of operations. Figure 2 displays the usage of three different partition strategies performed on the image. In the distributed file system the image parts are stored based both on spatial content and spectral space content, whilst the metadata is stored separately.

In case the data is exported from the file system, an *aggregation strategy* may be performed to combine the parts to form a single feature.

As properties of the dataset may change over time (due to input from new datasets or removal of stored data), a maintenance service is also required that can perform redistribution of data and index updating. The redistribution may also apply new distribution strategies to the data, when new requirements are made by operations. As maintenance requires the moving of data from one node to another, the service should monitor system activities and perform tasks in the background, when the specified nodes are not in use.

When working with Hadoop, one must also consider the limitations of the *Hadoop Distributed File System* (HDFS) (19). A complex image processing operation, such as object-based thematic classification, requires multiple transformations performed on the image creating intermediate results (20). In contrary, HDFS is designed for writing once, reading many times, thus performing such operations on disk may result in performance bottleneck. To overcome this, memory based caching is required, which must be shared by the processes performing steps of the operation.

The final extension comes from user aspect. As most organizations have their spatial datasets already stored in a specific place and format (e.g. relational database), transferring all data to the cloud may not be a preferred, or may not be available (for example, the cloud is public, and the data is copyrighted). Although this may cause severe performance reduction, the possibility must be present to access data from its original source, outside

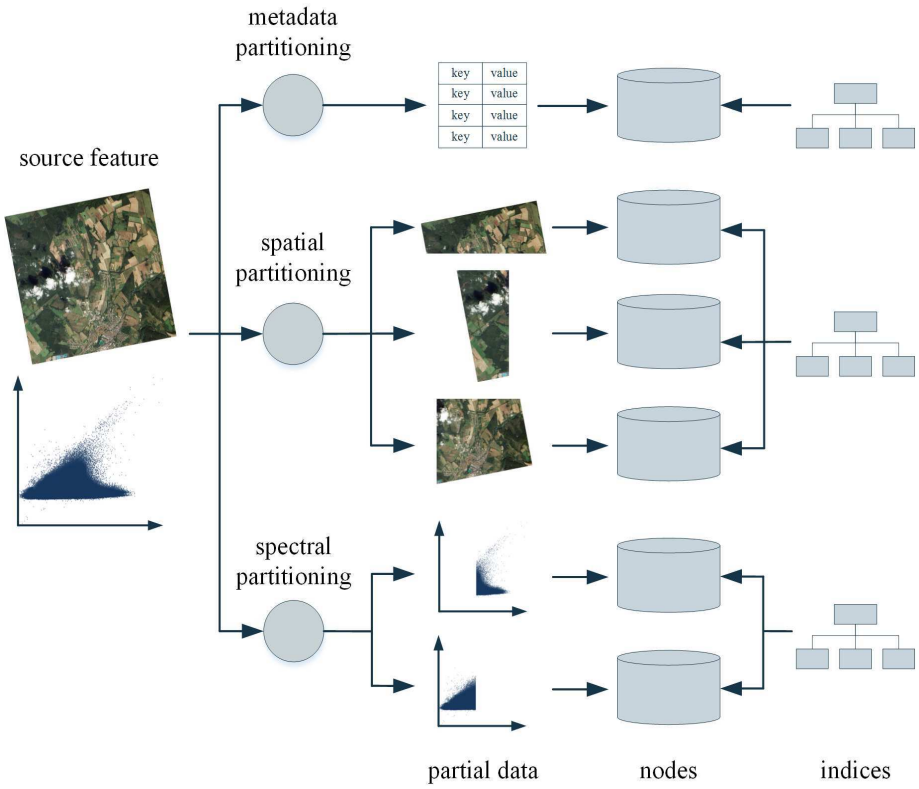


Fig. 2: The distributed storage model performing partitioning using three strategies

the cloud. Under some circumstances (e.g. network bandwidth, replication of the database server), accessing external data may become as fast as accessing data located on HDFS.

In conclusion, the data management component of the framework is extended with three services.

- The *data import/export service*, performing distribution and aggregation of data using one or more strategies, building indices, and allowing access to both data located on HDFS, and data located on external sources. Each dataset becomes an entity, and each distributed part becomes a child entity to be accessed by AEGIS operations.
- The *data maintenance service* performing redistribution of data and updating of indices in the background. The maintenance should have no impact on the properties of the entities.
- The *cache manager* operating the local in-memory cache at each node. These caches realize entities during wokflow execution.

5 Processing data

Based on the data management system described in Section 4, adapting the execution model to MapReduce paradigm is a straightforward process.

The enhancement to data management enable the distributed processing of data, as operations on a single feature can be performed by processing the partial features, which have been created using a specific distribution strategy. Before execution, the best fitting distribution of the data can be chosen based on the properties of the operation, for example, spatial

partitions in case of image segmentation, spectral partitions in case of image clustering. The parts are processed in parallel, with the resulting data written to the distributed file system. As operations may produce spatial or spectral data in the same, or greater magnitude as the source data, moving results from one node to another is not always an option. Thus, these parts are not aggregated together physically, but can still be considered as parts of a single result. For this purpose, the initial structure of the parent entity (e.g. the spatial index) is copied to form a parent entity of the resulting dataset. An example for waterlogging detection can be seen in Figure 3.

These operations are performed as MapReduce processes by an operations engine working in the Hadoop environment. The *Map* function executes the main operation on the partial data on a single node. In case of most operations, the results are independent of another, thus no further action is required. However, in some cases the results of multiple parts need to be merged together. For example, image segments computed on individual parts may be combined. This combination may be performed using an aggregation strategy, executed in the *Reduce* function.

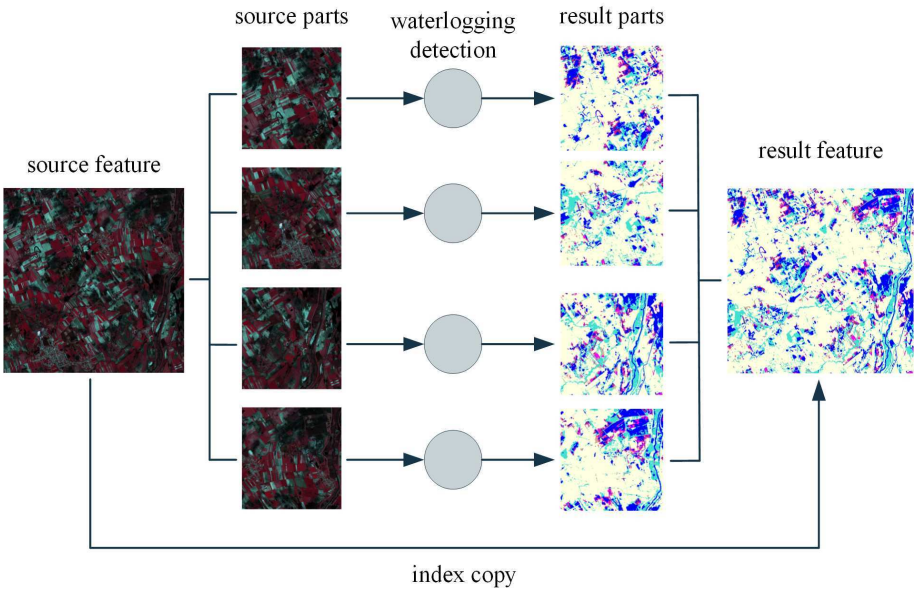


Fig. 3: Waterlogging detection performed in distributed environment

The engine can also handle the execution of workflows built up from multiple operations that are performed sequentially. For example, thematic classification can be performed by first segmenting the image, then clustering the segments, and finally classifying the clusters. If no merging of intermediate results is required, each operation can be performed in the same Map stage. In this case each intermediate result can be stored in the local cache to enhance performance.

To summarize, operations are performed using the MapReduce model in the following manner:

- In case no merging of results is required, the operation, or multiple operations can be performed using a single Map function, and the Reduce functionality is omitted.
- In case merging of results is required, the operation is performed using the Map function, and an aggregation is performed in the Reduce function. When multiple operations in a workflow require merging of intermediate results, multiple MapReduce processes are used for performing processing.

The execution process is managed by the operations engine and the Hadoop environment. All operations are automatically loaded on the dedicated node (containing the partial data), and executed in MapReduce form. The executed operations are not specialized for Hadoop in any way, as the environment handles all specialization responsibilities.

6 Conclusion

The paradigm shift to cloud computing can be a tough challenge, and usually requires a great deal of effort, because data management has to be reconsidered, algorithms have to be redesigned. However, when working with a flexible framework, the modification may be completely performed on system side, without the need of reimplementing algorithms. The porting of the AEGIS framework to Hadoop requires some effort, but does not require complete overhaul. Much thought has been given on achieving the most performance in the given circumstances.

After the implementation is finished, evaluation will follow by examining the system from the performance aspect. Based on the exact results, the system will be further tuned to achieve the most valuable speedup.

Acknowledgements This research is founded by the FP7 project IQmulus (FP7-ICT-2011-318787) a high volume fusion and analysis platform for geospatial point clouds, coverages and volumetric data set.

References

[1] Agrawal D., Das S., El Abbadi A.: Big Data and Cloud Computing: Current State and Future Opportunities. In *Proceedings of the 14th International Conference on Extending Database Technology (EDBT/ICDT '11)*, 530–533 (2011).

[2] Dean J., Ghemawat S.: MapReduce: Simplified Data Processing on Large Clusters. *Commun. ACM*, 51 (1), 107–113 (2008).

[3] Bhandarkar M.: MapReduce programming with Apache Hadoop. In *Parallel Distributed Processing (IPDPS), 2010 IEEE International Symposium on*, 1–1 (2010).

[4] Che D., Safran M., Peng Z.: From Big Data to Big Data Mining: Challenges, Issues, and Opportunities. In *Database Systems for Advanced Applications*, vol. 7827 of *Lecture Notes in Computer Science*, 1–15 (2013).

[5] Franklin M.: The Berkeley Data Analytics Stack: Present and future. In *Big Data, 2013 IEEE International Conference on*, 2–3 (2013).

[6] Csornai G., Mikus G., Nádor G., Hubik I., László I., Suba Z.: The first seven years of the remote sensing based Ragweed Monitoring and Control System. In *EARSeL eProceesings*, 110–118 (2011).

[7] Giachetta R.: AEGIS - A state-of-the-art spatio-temporal framework for education and research. *OSGeo Journal*, 13, 68–77 (2014).

[8] Yang C., Goodchild M., Huang Q., Nebert D., Raskin R., Xu Y., Bambacus M., Fay D.: Spatial cloud computing: how can the geospatial sciences use and help shape cloud computing? *International Journal of Digital Earth*, 4 (4), 305–329 (2011).

[9] Cary A., Sun Z., Hristidis V., Rishe N.: Experiences on Processing Spatial Data with MapReduce. *Scientific and Statistical Database Management*, vol. 5566 of *Lecture Notes in Computer Science*, 302–319 (2009).

[10] Eldawy A., Mokbel M. F.: A Demonstration of SpatialHadoop: An Efficient Mapreduce Framework for Spatial Data. *Proc. VLDB Endow.*, 6 (12), 1230–1233 (2013).

[11] Aji A., Wang F., Vo H., Lee R., Liu Q., Zhang X., Saltz J.: Hadoop GIS: A High Performance Spatial Data Warehousing System over Mapreduce. *Proc. VLDB Endow.*, 6 (11), 1009–1020 (2013).

[12] Thusoo A., Sarma J. S., Jain N., Shao Z., Chakka P., Anthony S., Liu H., Wyckoff P., Murthy R.: Hive: A Warehousing Solution over a Map-reduce Framework.*Proc. VLDB Endow.*, 2 (2), 1626–1629 (2009).

[13] Golpayegani N., Halem M.: Cloud Computing for Satellite Data Processing on High End Compute Clusters. In *Cloud Computing, 2009. CLOUD '09. IEEE International Conference on*, 88–92 (2009).

[14] Alonso-Calvo R., Crespo J., Garc'ia-Remesal M., Anguita A., Maojo V.: On distributing load in cloud computing: A real application for very-large image datasets. *Procedia Computer Science*, 1 (1), 2669 – 2677, (2010).

[15] Zhang C., Sterck H., Aboulnaga A., Djambazian H., Sladek R.: Case Study of Scientific Data Processing on a Cloud Using Hadoop. In *High Performance Computing Systems and Applications*, 400–415 (2010).

[16] Zaharia M., Chowdhury M., Franklin M. J., Shenker S., Stoica I.: Spark: cluster computing with working sets. In *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, 10–16 (2010).

[17] Engle C., Lupher A., Xin R., Zaharia M., Franklin M. J., Shenker S., Stoica I.: Shark: Fast Data Analysis Using Coarse-grained Distributed Memory. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data (SIGMOD '12)*, 689–692 (2012).

[18] Liao H., Han J., Fang J.: Multi-dimensional Index on Hadoop Distributed File System. In *Networking, Architecture and Storage, 2013 IEEE Eighth International Conference on*, 240–249 (2010).

[19] Shvachko K., Kuang H., Radia S., Chansler R.: The Hadoop Distributed File System. In *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, 1–10 (2010).

[20] Dezső B., Fekete I., Gera D., Giachetta R., László I.: Object-based image analysis in remote sensing applications using various segmentation techniques. *Ann. Univ. Sci. Budap. Rolando Eötvös, Sect. Comput.*, 37 (1), 103–120 (2012).

An out-of-core octree for massive point cloud processing

K. Wenzel, M. Rothermel, D. Fritsch, N. Haala

Abstract Image based surface acquisition using dense image matching methods enables the retrieval of 3D information for each pixel. While the sensor resolutions increase, also the desire of acquiring and handling datasets with more images increases. Consequently, massive point clouds and other surface representations are produced, which need to be accessed and processed efficiently. For this purpose, we present a framework called Pine Tree, which is based on an out-of-core octree. It is currently focused on point clouds, but can be extended to other data types such as meshes or volumetric representations. It enables spatially indexed data storage and quick data queries. Moreover, only parts of the data are loaded from the hard disk to the memory, in order to be able to process big data on common hardware. Within this paper, we present the Pine Tree framework as well as an example filtering operation, which uses the redundancy of overlapping point clouds in order to perform outlier rejection and data reduction while preserving accuracy.

1 Introduction

Fig. 1: Point cloud derived from airborne imagery

Institute for Photogrammetry, University of Stuttgart, Geschwister-Scholl-Str. 24D, 70174 Stuttgart, Germany e-mail: [konrad.wenzel, mathias.rothermel, dieter.fritsch, norbert.haala]@ifp.uni-stuttgart.de

Point cloud datasets in common projects can contain billions of points nowadays in particular for dense image matching applications, where on the one hand the amount of images and on the other hand the resolution for each image increases. Quick data access is essential for general bounding box retrieval, but also for processing tasks like filtering, surface reconstruction or segmentation.

For this purpose, we would like to implement a flexible data structure focussed on non-uniformly distributed point clouds supporting unlimited custom data fields. In order to adapt automatically to data of varying density, a maximum depth shall be avoided. Furthermore, frequent update and removal of data should be supported. Consequently, the following requirements can be defined:

Big data. The framework should be able to handle massive point clouds or other big spatial data.

Inhomogeneous data. The distribution, density and precision of the acquired data are varying.

Data queries. Efficient searching on the data is essential for filtering, visualization or queries on bounding volumes.

Data updating. Data adding and removal are essential for manipulation processes such as filtering.

Precision preservation In order to maintain data quality, simplification, resampling or lossy compression are avoided.

1.1 Out-of-core octree

In order to meet the specified requirements concerning data querying, we are using an *octree* (9) as access structure. Octrees are based on a *space-driven* partitioning approach which can have the disadvantage of imbalance if the data is not well distributed. In such cases, data-driven approaches like *KD-Trees* as proposed by (1) would require less memory for the tree structure and enable faster data access.

However, in our case we want to support quick data update, which is better supported by the regular octree, since the tree partitioning is not relying on the current data, and thus, doesnt require an update of the structure when adding or removing data. Furthermore, the octree is well suited for *out-of-core* implementations, since the partitioning at each level is identical and thus, does not require reading additional information.

Out-of-core refers to algorithms, which store the data not only in the main memory, but also stream from an external memory source. This is particularly useful when the data to be accessed can be significantly larger than the available main memory.

Out-of-core tree structures are widely used in research often for visualization, such as (13), (4) or (8). Beside this application, also processing on the data is performed for example Poisson surface reconstruction like in (2) or mesh simplification (3).

In general, only few open implementations are available. The *Point Cloud Library* (PCL) at pointclouds.org offers an out-of-core module for point clouds. However, at the current version 1.7, it is rather focused on visualization and uniformly distributed point clouds only, since the tree is created completely for an a priori defined depth.

The *out-of-core* data structure introduced by (5) and published in the 3D Toolkit, overcomes this limitation and works with dynamically splitting the tree until a minimum number of points are exceeded or predefined maximum depth is reached. Thus, it adapts much better to non-uniformly distributed data, since no subdivision is performed at unoccupied parts of the dataset.

2 Pine Tree

2.1 Approach

The *Pine Tree* framework provides methods for data updating and querying, while taking decisions for data storage and retrieval automatically. This enables the development of processing algorithms for large data with low efforts. In order to meet the remaining requirements discussed in the specification of section 1, several features have been implemented:

Dynamic depth. Instead of limiting the tree structure to a pre-defined depth, the depth is dynamically adapted locally. By defining thresholds for the maximum data storage, the depth can be controlled dynamically to the needs of the particular tasks of updating and querying. Thus, the requirement of local adaption to non-uniformly distributed datasets is met.

Dynamic loading and writing. Since an octree is used, each node is split into eight subsequent nodes. These nodes can be stored in a folder structure on the hard disk, where each folder has eight subfolders. At the final node (leaf node), the data can be stored as a file.

Dynamic memory management. In order to perform efficient processing on the tree, as much data should be in the memory (in-core) as possible, while not exceeding the limits of the main memory. For this purpose, a maximum count of nodes in-core can be defined. The framework detects, writes and de-allocates unused parts of the tree based on a usage history.

2.2 Tasks

Finding nodes In order to find the node corresponding to an X, Y, Z position, the tree can be traversed from top to down. At each node, the containing sub-node is found. By repeating this process until the node in the final level is reached, the destination leaf node can be determined. In order to improve speed for the frequent operation of node finding, we do avoid a check on the *bounding box* for all eight sub-nodes, but only compare to each coordinate of the node center once.

Data adding In order to add data to the tree, the destination node is found for each point. Subsequently, the data is added to the node data vector. In order to avoid time consuming data allocation, data copying and data de-allocation as happening for typical push back approaches, we determine the destination nodes firstly and then add the data to the nodes as blocks.

Node splitting and merging While most in-core processes require a small count of data items per node for efficient operations, the *out-of-core* part requires large data blocks. Therefore, node splitting and merging is frequently required. During splitting, the node data is distributed to its eight sub-nodes and during merging, the data of the sub-nodes is added to their parent node.

Data reading and writing In order to write a node to disk, the subsequent tree structure is created as folder structure on the hard drive, where each folder indicates the sub-node number (e.g Tree/0/3/4/2 for a branch node at level 4). Nodes are only written to the hard disk, if they contain data. By merging nodes, the resulting larger files can be accessed more efficiently.

2.3 Memory management

The memory management in the *Node Manager* is based on a Node history. This history is represented by a cycle stack vector of pointers to Nodes with a constant predefined size n. By storing the index i of the last accessed element, the elements can be written in a cycling manner with overwriting always the oldest element.

Node tracking. In order to keep track of the nodes in memory, the history is updated each time a new node is loaded to the memory. This occurs in particular during node splitting and node reading. Instead of keeping track of all leaf nodes, we only track their directly parental branch nodes to reduce the processing overhead.

Node de-allocation. As soon as a new node is registered, we write it to the history vector at the position i, which represents the oldest node. The previous node at this position is written to disk and de-allocated.

Node sharing. The key challenge is the avoidance of de-allocation, if a node is still in use. Beside the fact that multiple processes can add the same node to the history, there might also be a dependency from another process to one of the child nodes. De-allocation would then lead to highly frequent reading and writing operations, which would slow down the process.

The solution to the problem of de-allocating shared nodes, is to keep track whether each branch is been in use by other branches. Instead of explicitly validating this for each element of the history at each time of node registration, we encode this access implicitly in our tree structure. Each time a node is registered, we store the access number defined by the history in the index i in the node. De-allocation and writing for a node is only performed, if the access number is equal to the current access index i. This way, we can avoid the de-allocation of nodes, which are still required by other nodes in memory.

In order to minimize the overhead of the node tracking, we only register a node pointer if it is not the equal to the previously registered node and if it is not equal to the previously registered parent node. This way we can avoid overhead during frequent occurring subsequent splitting steps. Thus, few registrations are required during tree unfolding e.g. during data adding, in particular if large blocks of data are processed at once or dynamic splitting during processing for faster queries.

Folding and unfolding. A key bottleneck of an out-of-core tree is the access to the hard disk. While for the processing in-core many nodes with small data portions are beneficial, the writing of many small files onto the hard disk requires a lot of time. This is due to the general access latency for each file, but also the overhead of updating the file system tree structure. For this purpose, we merge nodes until they contain a *minimum data count* before writing and split them after reading.

2.4 Usage

The Pine Tree is implemented in a C++ environment. Currently, the tree can contain point clouds with various fields and can easily be extended to other data types like meshes or volumetric representations. Operations on the tree can be implemented using only basic functions like *getData* or *getSubnodes*. The whole memory management is performed automatically in background.

3 Point cloud filtering

Within the following section, we would like to present a simple filter exploiting and reducing the redundancy occurring for overlapping point clouds. Within the local neighbourhood only the densest point cloud is preserved, while points from clouds with locally less density get rejected.

This is particularly useful for applications, where point clouds were retrieved from multiple stations as occurring for point clouds from dense image matching and laser scanning. The overlaying clouds can on the one hand be used to validate each other in order to reject outliers by enforcing a minimum redundancy. On the other hand, only the locally densest cloud is preserved, which rejects noisier data from stations far away.

3.1 Algorithm

The core concept of the algorithm is to split the tree, until each node contains from each point cloud one point at maximum. Thus, no node can have two points from the same point cloud.

By preserving the information which cloud was the densest locally during the recursive splitting, we can then preserve only the point from this cloud while rejecting all points from clouds with locally less resolution. Consequently, only the cloud with the highest density remains.

The splitting of the tree is performed by splitting the leaf to become a branch node, recursively calling the filtering function again on the sub-nodes and merging the resulting nodes. Consequently, the tree is unfolded locally for the filtering and subsequently folded again by merging the resulting data.

Additionally to the local density filtering, we can constrain a certain redundancy to perform point consistency validation. This is particularly useful when outliers remain in the point cloud for example if only stereo image pairs were used for the point retrieval.

The redundancy constraint can be applied on the final leaf nodes after the recursive splitting process described above. Since each leaf node can have only one point cloud source, we can reject the leaf as soon as the point count is less than the minimum defined fold (min fold). Results are shown in the appendix.

4 Results

In order to evaluate the performance of the tree as well as the filter, several data sets have been processed with the software *SURE* (11). The resulting highly overlapping point clouds for each image have been added to the tree without previous sorting. Subsequently, the data was added to the PineTree and the filtering approach described in section 3 was applied. The point density is adapting to the local maximum resolution available, while rejecting redundancy and outliers.

5 Conclusions

Within this paper, we presented an out-of-core octree structure for processing massive point clouds. By indexing data spatially bounding box or nearest neighbor queries can be performed efficiently for tasks like filtering or the retrieval of regions of interest. The out-of-

Fig. 2: Troll. 9 images with 4 MP each. Fold: 2. Pointcount: 15.2 Mio. before filtering, 2.3 Mio. after filtering.

Fig. 3: Perth - airborne image dataset. Left: result overview, right: detail. Pointcount: 394 Mio. points before filtering, 73 Mio. points after filtering with fold 2. Imagery kindly provided by Aerodata International Surveys.

Fig. 4: Munich airborne image dataset. Result for filtering with fold 2. Pointcount: 2.8 Bio. points before filtering, 437 Mio. points after filtering with fold 2, 218 Mio. points after filtering with fold 3.

core memory management only loads desired parts of the dataset from the hard disk and thus enables big data handling on common hardware.

Within an example application, overlapping clouds from multiple sources are filtered by preserving only the locally densest point cloud. Additionally, redundancy constraints can be used to validate data and to reject outliers.

Within future work, the tree structure can be extended to support other data types such as meshes or scalar fields to achieve a volumetric data representation. Furthermore, the tree can be applied for visualization purposes or tasks like simplification, segmentation and modeling.

Acknowledgements We would like to acknowledge the work of Minwei Tang, who evaluated existing out-of-core algorithms in his study thesis Out-of-core algorithms for large point clouds in photogrammetric applications.

References

[1] Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. Communications of the ACM, 18(9), 509-517.

[2] Bolitho, M., Kazhdan, M., Burns, R. and Hoppe, H. (2007, July). Multilevel streaming for out-of-core surface reconstruction. In Symposium on geometry processing (pp. 69-78).

[3] Cignoni, P., Montani, C., Rocchini, C. and Scopigno, R. (2003). External memory management and simplification of huge meshes. Visualization and Computer Graphics, IEEE Transactions on, 9(4), 525-537.

[4] Correa, W. T., Klosowski, J. T. and Silva, C. T. (2002). iWalk: Interactive out-of-core rendering of large models. Technical Report TR-653-02, Princeton University.

[5] Elseberg, J., Borrmann, D. and Nuchter, A. (2011). Efficient processing of large 3d point clouds. In Information, Communication and Automation Technologies (ICAT), 2011 XXIII International Symposium on (pp. 1-7). IEEE.

[6] Finkel, R. A. and Bentley, J. L. (1974). Quad trees a data structure for retrieval on composite keys. Acta informatica, 4(1), 1-9.

[7] Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching (Vol. 14, No. 2, pp. 47-57). ACM.

[8] Lindstrom, P. (2003, April). Out-of-core construction and visualization of multiresolution surfaces. In Proceedings of the 2003 symposium on Interactive 3D graphics (pp. 93-102). ACM.

[9] Meagher, D. J. (1980). Octree encoding: A new technique for the representation, manipulation and display of arbitrary 3-d objects by computer. Electrical and Systems Engineering Department Rensseiaer Polytechnic Institute Image Processing Laboratory.

[10] Moussa, W., Wenzel, K., Rothermel., M., Abdel-Wahab, M. and Fritsch, D. (2013). Complementing TLS Point Clouds by Dense Image Matching. Int. Journal of Heritage in the Digital Era, Vol. 2, No. 3, pp. 453-470.

[11] Rothermel, M., Wenzel, K., Fritsch, D., Haala, N. (2012). SURE: Photogrammetric Surface Reconstruction from Imagery. Proceedings LC3D Workshop, Berlin, December 2012. http://www.ifp.uni-stuttgart.de/publications/softwa re/sure/index.en.html

[12] Tang, M. (2014). Out-of-core algorithms for large point clouds in photogrammetric applications. Study thesis, Institute for Photogrammetry, University of Stuttgart, 2014.

[13] Ueng, S. K., Sikorski, C. and Ma, K. L. (1997). Out-of-core streamline visualization on large unstructured meshes. Visualization and Computer Graphics, IEEE Transactions on, 3(4), 370-380.

[14] Wenzel, K., Rothermel, M., Haala, N. and Fritsch, D. (2013). SURE The ifp Software for Dense Image Matching. Photogrammetric Week '13, Ed. D. Fritsch, Wichmann, Berlin/Offenbach, pp. 59-70.

A web-based distributed system to process large geometric models

Daniela Cabiddu and Marco Attene

1 Introduction

Running experiments is a fundamental activity in geometry processing research. A typical experiment in this area consists in considering an input 3D object (eg. a polygonal mesh), performing a sequence of operations on it, and analyzing the results. Sometimes a fixed sequence of operations is used to process a variety of data sets, whereas some other times the operation list is slightly changed while keeping the input constant.

Modern collaborative environments need to enable the sharing of experiments that must be available and reusable by other researchers. It must be possible to rerun state-of-the-art algorithms or to compare them with other work. Since researchers often prefer not to share their source code or binary, state-of-the-art algorithms are reimplemented from paper descriptions by who wants to reuse them but, due to their complexity, this easily becomes a costly and error prone operation.

Online repositories are the most common solution used to share algorithm results. The existing repositories explicitly store the uploaded models; since each of them can easily be made of millions of triangles, significant storage resources are required.

Our system provides an effective solution. Based on the assumption that "algorithm results" are, in most cases, modifications of the original, it is possible to produce these models on demand instead of storing them explicitly. A standard Web browser is sufficient to remotely run complex geometric algorithms by distributing the various sequential steps on different servers that expose algorithms in form of Web services. Researchers can simply expose their new algorithms as web services so that other researchers can exploit them for extension or comparison purposes, without the need to distribute source code or executable files. Our main scientific contribution is an efficient transfer protocol that allows to take advantage of the system for processing large models and avoid the possible bottleneck due to the huge amount of data that may be required to be sent through the net.

The advantage of a repository endowed with our system are diverse. First, such a repository requires substantially less resources to share algorithm results. Secondly, researchers can stack geometric algorithms to construct workflows and execute them, with no need to locally install any software or library. Also, researchers in other fields can exploit geometric algorithms without the need to be skilled programmers.

Daniela Cabiddu
CNR IMATI, Via De Marini 6, Genova, Italy, e-mail: daniela.cabiddu@ge.imati.cnr.it

Marco Attene
CNR IMATI, Via De Marini 6, Genova, Italy, e-mail: marco.attene@ge.imati.cnr.it

2 Related Work

2.1 3D Model Repositories

The Stanford 3D Scanning Repository (1) is one of the earliest widely-used online collections of 3D models, and several authors used its meshes to demonstrate the benefits of their algorithms and the improvements achieved over the state of the art. Other collections exist that deal with synthetic CAD models (2) or that focus on specific algorithms, such as the Princeton Shape Benchmark (8).

The more recent Digital Shape Workbench (DSW) (3) tries to encapsulate most of the functionalities provided by previous repositories. It provides a data repository and a knowledge management system able to perform data browsing and discovery. The data repository is aimed to collect and share a large number of 3D shapes and state-of-the-art tools that can be used to process digital models.

Redundancy is one of their main problems, due to the fact that outputs of similar processes, which are often very similar to each other, are individually uploaded with no attention to avoid possible duplicated models and data.

2.2 Geometric Experiment Replication

In computer graphics and geometry processing, polygon meshes are the dominant representations for 3D objects, and diverse mesh processing software tools exist. Among them, MeshLab (5) and OpenFlipper (6) allow to interactively edit a mesh, save the sequential list of executed operations and locally re-execute the workflow from their user interfaces. Workflows can be shared and rerun on different machines where the stand-alone applications need to be installed.

To get rid of any specific software, hardware, and operating system, Campen and colleagues published an online service called WebBSP (4) which is able to remotely run a few specific geometric operations. The system is accessible from a standard web browser and the user is required to upload an input mesh; then, a single geometric algorithm must be selected from a set of available operations. The algorithm is actually run on the server and a link to download its output is sent to the user by email. The available operations are not customizable by users, only one of them can be run at each call, and the service is accessible only from the WebBSP graphical interface.

Geometric Web services were previously considered by Pitikakis (7) with the objective of defining semantic requirements to guarantee their interoperability. Though in Pitikakis's work Web services are stacked into hardcoded sequences, users are not allowed to dynamically construct workflows, and geometric issues such as the evaluation of mesh qualities (necessary to support conditional tasks and loops) and the transmission of large models are not dealt with.

3 The Geometric Workflow System

Our system allows the user to build workflows to process and analyse 3D triangular meshes and efficiently share the experiments through the Internet.

The framework architecture (see Figure 1) is organized in three layers: (1) a graphical user interface that allows building new workflows from scratch, uploading existing pipelines and invoking available ones, (2) the workflow engine responsible of runtime execution, (3) the web services that wrap geometry processing tools.

The workflow engine is the core of the system and orchestrates the invocation of the various web services involved. From the user interface it receives the specification of a geometry processing workflow and possibly the address of an input mesh to be downloaded from the Internet. When all the data is available, the workflow engine sequentially invokes the various algorithms and returns the URL of the eventual result to the user interface. The engine is also able to manage the execution of conditional tasks and loops, and delegates the evaluation of the condition itself to specific web services.

Web services, instead, can be considered as black boxes able to run a simple operation on a 3D triangular mesh using possible input parameters, store the output on the server where it is located and make the output available by returning its address. Note that a single server (i.e. a provider) can expose a plurality of web services implementing a variety of algorithms.

4 Mesh Transfer Protocol

The transfer of large meshes from a server to another according to the aforementioned protocol constitutes a bottleneck in the workflow execution. Mesh compression techniques can be used to reduce the input size, but they do not solve the intrinsic problem. In order to improve the transfer speed and thus efficiently support the processing of large meshes, we designed a mesh transfer protocol inspired on the prediction/correction metaphor used in data compression.

We have observed that there are numerous mesh processing algorithms that simply transform an input mesh into an output by computing and applying local or global modifications. Furthermore, in many cases modifications can be only local (e.g. sharp feature restoration), may involve the geometry only while keeping the connectivity unaltered (e.g. most mesh deformation algorithms), or may modify both geometry and connectivity while minimally changing the overall shape (e.g. remeshing). In all these cases it is possible to predict the result by assuming that it will be identical to the input, and it is reasonable to expect that the corrections to be transmitted can be more compactly encoded than the explicit result of the process.

The protocol works as follows (an example of execution of a simple workflow composed by three tasks is shown in Figure 2). Through the user interface, the user selects/sends a workflow and possibly the URL of an input mesh to the workflow engine. The engine analyses the workflow, locates the most appropriate servers hosting the involved web services, and sends in parallel to each of them the address of the input mesh. Each server is triggered to download the input model and save it locally. At the first step of the experiment, the workflow engine triggers the suitable web service that runs the algorithm, produces the result, and locally stores the output mesh and the correction file (both compressed). Their URLs are returned to the workflow engine that forwards them to all the subsequent servers

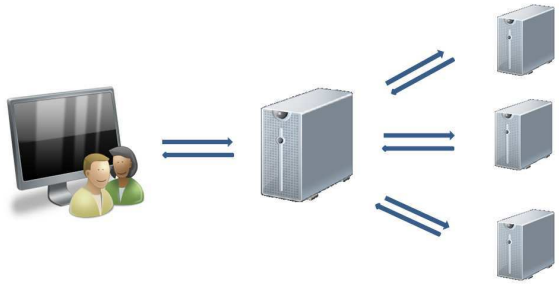


Fig. 1 The three-layered system architecture.

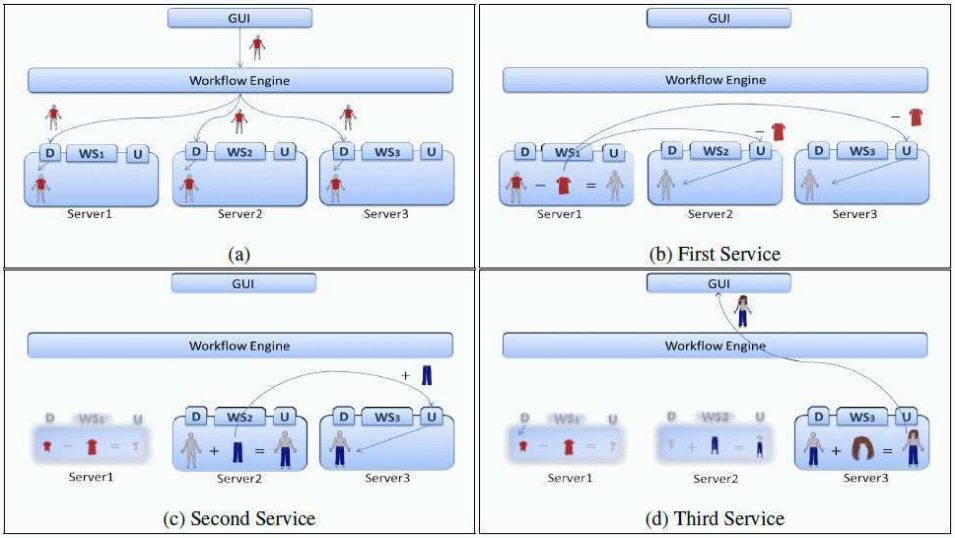


Fig. 2: Mesh Transfer Protocol Example. Three servers are involved into the workflow execution. Each of them exposes a web service to support a geometry processing algorithm and two modules able to download (D) meshes and update (U) the previously downloaded mesh by applying the corrections. (2a) The engine shares in parallel the address of the input mesh with all the involved servers that proceed with the download. (2b) The first service runs the task, produces the corrections and returns the corresponding address to the engine that shares it in parallel to the following involved servers. Both download the file and correct the prediction. (2c) The second service is invoked, runs the task and makes the correction available, so that the third involved server can download it and update its local copy of the mesh. (2d) The engine triggers the third service that runs the algorithm and makes available the modified output mesh so that it can be directly downloaded by the user.

involved in the workflow. Each server downloads the correction and applies it to the mesh it already has in memory in order to update the local copy of the model. Then, the workflow engine triggers the next service for which an up-to-date copy of the mesh is readily available on its local server. At the end of the workflow execution, the engine receives the address of the output produced by the last invoked web service and returns it to the user interface, so that the user can perform the download.

In this scenario, the entire input mesh is broadcasted only once at the beginning of the process, whereas the final result is transmitted only once at the end. Inbetween, only the corrections are broadcasted to the subsequent servers. Thus, when the corrections are actually smaller than the partial results, this procedure produces significant benefits; otherwise, the subsequent web services can directly download the output instead of the corrections and no degradation is introduced.

5 Results and Discussion

For the sake of experimentation, the proposed Workflow Management System has been deployed on a standard server running Windows 7, whereas web services implementing atomic tasks and check mesh qualities have been deployed on different machines to constitute a distributed environment. However, since all the servers involved in our tests were in the same lab with a gigabit network connection, we simulated a long-distance network by artificially limiting the transfer bandwidth to 5 Mbps.

Table 1: Dataset extracted from the Digital Shape Workbench.

Mesh	 Isidore	 Nicolo	 Neptune	 Ramesses	 Raptor	 Dancers
Vertices	1.071.671	945.924	1.321.838	775.712	1.000.080	703.207
Triangles	2.128.494	1.886.968	2.643.684	1.537.462	2.000.000	1.399.805
Components	161	103	1	308	51	1
Boundaries	404	157	0	824	0	105

Table 2: Output sizes (in KB). For each mesh and for each task, the first line shows the size of the compressed output mesh, while the second line reports the size of the compressed correction. Acronyms indicate Removal of Smallest Components (RSC), Laplacian Smoothing (LS), Hole Filling (HF), and Removal of Degenerate Triangles (RDT).

Mesh	**RSC**	**LS**	**HF**	**RDT**
Isidore	20.573 11	23.333 9.433	23.717 154	25.497 2
Nicolo	19.498 3	21.447 9.296	20.601 48	20.171 2
Neptune	39.881 1	40.131 15.237	39.891 1	39.937 1
Ramesses	17.484 3	19.544 8.754	19.934 149	19.802 3
Raptor	14.465 688	15.621 10.195	15.552 1	15.441 1
Dancers	16.457 1	18.037 7.220	18.325 80	18.116 1

Then, to test such a system we defined multiple processing workflows involving the available web services. The dataset has been constructed by selecting some of the most complex meshes currently stored within the Digital Shape Workbench (see Table 1).

The same workflow was run on all the other meshes in our dataset to better evaluate the performance gain achievable thanks to our concurrent mesh transfer protocol. Table 2 reports the size of the output mesh and the size of the correction file after each operation (both after compression) whereas Table 3 shows the total time spent by the workflow along with a more detailed timing for each single phase.

As expected, the corrections related to tasks that locally modify the model are significantly smaller than the whole output mesh by several orders of magnitude; corrections regarding more “global” tasks are also smaller than the output mesh, although in this latter case the correction file is just two/three times smaller than the whole output. Nevertheless, these results confirm that the proposed concurrent mesh transfer protocol provides significant benefits when the single steps produce mainly little or local mesh modifications.

Clearly, the additional instructions introduced in the geometry processing algorithms to stream out the corrections should be considered for a fair comparison, but we have verified that such an overhead is negligible with respect to the overall processing time of each algorithm, and therefore has not been reported in Table 3.

To summarize, our tests show that the concurrent mesh transfer protocol considerably reduces the amount of data transferred among the servers, and thus the total elaboration time.

Table 3: Elaboration times (in seconds). Acronyms indicate Input Broadcasting (IB), Removal of Smallest Components (RSC), Laplacian Smoothing (LS), Hole Filling (HF), and Removal of Degenerate Triangles (RDT). Cells labelled by $\mathbf{T}_i$ indicate the time needed to transfer the correction file. Cells labelled by $\mathbf{U}_i$ indicate the time needed to update the mesh by applying the correction.

Mesh	**IB**	**RSC**	$\mathbf{T}_1$	$\mathbf{U}_1$	**LS**	$\mathbf{T}_2$	$\mathbf{U}_2$	**HF**	$\mathbf{T}_3$	$\mathbf{U}_3$	**RDT**	**Total**	**Benefits**
Isidore	33,0	7,7	0,0	5,8	12,4	15,1	7,1	8,4	0,2	6,0	13,8	109,5	67%
	33,0	7,7	32,9		12,4	37,3		8,4	37,9		13,8	183,4	
Nicolo	31,2	6,5	0,0	4,8	10,5	14,9	6,1	7,5	0,1	4,9	11,5	98,0	69%
	31,2	6,5	31,2		10,5	34,3		7,5	33,0		11,5	165,7	
Neptune	63,8	13,0	0,0	0,0	18,6	24,4	11,0	12,6	0,0	0,0	14,4	157,8	99%
	63,8	13,0	63,8		18,6	64,2		12,6	63,8		14,4	314,2	
Ramesses	28,0	6,7	0,0	4,3	9,6	14,0	5,4	7,0	0,2	4,5	10,3	90,0	70%
	28,0	6,7	28,0		9,6	31,3		7,0	31,9		10,3	152,8	
Raptor	26,7	7,7	1,1	5,6	9,6	16,3	5,8	6,0	0,0	0,0	9,3	88,1	50%
	26,7	7,7	23,1		9,6	25,0		6,0	24,9		9,3	132,3	
Dancers	26,3	4,9	0,0	0,0	7,3	11,6	4,3	5,2	0,1	3,6	7,0	70,3	92%
	26,3	4,9	26,3		7,3	28,9		5,2	29,3		7,0	135,2	

6 Conclusion

We proposed a workflow-based framework to support collaborative research in geometry processing. It allows scientists to remotely run geometric algorithms provided by other researchers as Web services and to combine them in order to create workflows to be executed on any appropriate input mesh.

By distributing the workload, our system can count on considerable computational resources and can be easily extended if necessary, while the potential mesh delivery bottleneck has been resolved by our concurrent data transfer protocol.

Future works are intended to improve the system performance by introducing the possibility to include parallel algorithms in workflow executions. Such algorithms would enable out-of-memory processing of big data and would reduce the overall execution time.

Acknowledgements This work has been partly supported by the PO CRO Fondo Sociale Europeo Regione Liguria 2007-2013 Asse IV “Capitale Umano” Ob. Specifico I/6 through the project “Tecniche di visualizzazione avanzata di immagini e dati 3D in ambito biomedicale”, and by the European Commission under grant agreement 262044 “VISIONAIR”. The authors are grateful to all the colleagues at IMATI and Softeco Sismat Srl for the helpful discussions.

References

[1] The Stanford 3D Scanning Repository. http://graphics.stanford.edu/data/3Dscanrep. 1996
[2] 3D CAD Browser. http://www.3dcadbrowser.com. 2001
[3] DSW V5.0 - Visualization Virtual Services. http://visionair.ge.imati.cnr.it. 2012
[4] Campen, M.: WebBSP 0.3 beta. http://www.graphics.rwth-aachen.de/webbsp. 2010
[5] Cignoni, P., Corsini, M., Ranzuglia, G.: MeshLab: an Open-Source 3D Mesh Processing System. ERCIM News, **73**, 45–46, April 2008
[6] Möbius, J., Kobbelt, L.: OpenFlipper: An Open Source Geometry Processing and Rendering Framework. In: Curves and Surfaces - Lecture Notes in Computer Science, **6920**, 488–500, Springer Berlin / Heidelberg, 2012.
[7] Pitikakis, M.: A Semantic Based Approach For Knowledge Management, Discovery and Service Composition Applied To 3D Scientif Objects. PhD thesis, University

of Thessaly, School of Engineering, Department of Computer and Communication Engineering. 2010

[8] Shilane, P., Min, P., Kazhdan, M., Funkhouser, T.: The Princeton Shape Benchmark. In: SMI'04: Proceedings of the Shape Modeling International 2004 (SMI'04), 167–178, 2004

File-centric Organization of large LiDAR Point Clouds in a Big Data context

Jan Boehm

1 Introduction

Current LiDAR point cloud processing workflows often involve classic desktop software packages or command line interface executables. Many of these programs read one or multiple files, perform some degree of processing and write one or multiple files. Examples of free or open source software collections for LiDAR processing are LASTools (9) and some tools from GDAL (and the future PDAL). Files have proven to be a very reliable and consistent form to store and exchange LiDAR data. In particular the ASPRS LAS format, (10) has evolved into an industry standard which is supported by every relevant tool. For more complicated geometric sensor configurations the ASTM E57 (7) seems to follow the same path. For both formats open source, royalty free libraries are available for reading and writing. There are now also emerging file standards for full-waveform data. File formats sometimes also incorporate compression, which very efficiently reduces overall data size. Examples for this are LASZip (8) and the newly launched ESRI Optimized LAS (5). Table 1 shows the compact representation of a single point in the LAS file format (Point Type 0). Millions of these records are stored in a single file. It is possible to represent the coordinates with a 4 byte integer, because the header of the file stores an offset and a scale factor, which are unique for the whole file. In combination this allows a LAS file to represent global coordinates in a projected system.

Item	**Format**	**Size**	**Required**
X	long	4 bytes	*
Y	long	4 bytes	*
Z	long	4 bytes	*
Intensity	unsigned short	2 bytes	
Return Number	3 bits (bits 0, 1, 2)	3 bits	*
Number of Returns	3 bits (bits 3, 4, 5)	3 bits	*
Scan Direction Flag	1 bit (bit 6)	1 bit	*
Edge of Flight Line	1 bit (bit 7)	1 bit	*
Classification	unsigned char	1 byte	*
Scan Angle Rank	char	1 byte	*
User Data	unsigned char	1 byte	
Point Source ID	unsigned short	2 bytes	*
		20 bytes	

Table 1: Compact representation of a single point in the LAS file format.

University College London

This established tool chain and exchange mechanism constitutes a significant investment both from the data vendors and from a data consumer side. Particularly where file formats are made open they provide long-term security of investment and provide maximum interoperability. It could therefore be highly attractive to secure this investment and continue to make best use of it. However, it is obvious that the file-centric organization of data is problematic for very large collections of LiDAR data as it lacks scalability.

2 LiDAR Point Clouds as Big Data

Developments in LiDAR technology over the past decades have made LiDAR to become a mature and widely accepted source of geospatial information. This in turn has led to an enormous growth in data volume. For airborne LiDAR a typical product today which can be bought form a data vendor is a 25 points per square meter point cloud stored in a LAS file. This clearly exceeds by an order of magnitude a 1 meter DEM raster file, which was a GIS standard product not so long ago. Not only did the point count per square meter increase but the extent in the collection of data has significantly increased as well. As an example we can use the official Dutch height network (abbreviated AHN), which has recently been made publicly available (1). The Netherlands extend across approximately 40,000 square kilometres (including water surfaces). At 25 points per square meter or 25,000,000 points per square kilometre a full coverage could theoretically result in 2510640103=1012 or one trillion points. The current AHN2 covers less ground and provides an estimated 400 billion points (13). It delivers more than 1300 tiles of about one square kilometre each of filtered terrain-only points at an average density of 10 points per square meter. The same volume is available for the residual filtered out points (e.g. vegetation and buildings). Together this is over a terabyte of data in more than 2500 files.

Figure 1 shows a single tile of that dataset. It is one of the smallest tiles of the dataset. The single tile contains 56,603,846 points and extends from 141408.67 to 145000.00 in Easting and 600000.00 to 601964.94 in Northing. In compressed LAZ format it uses 82 MB disk space and uncompressed it uses 1.05 GB. In terrestrial mobile scanning acquisition rates have now surpassed airborne acquisition rates and therefore data volumes can become even larger. Organization such as public transport authorities are scanning their tunnel systems in regular intervals for inspection and monitoring purposes. Repetitive acquisitions at centimetre and even millimetre spacing result in large collections which accumulate over the years. In order to detect changes over several epochs data from previous acquisitions needs to be available just as well as the most recent acquisition. The examples described here clearly result in tough requirements on data storage, redundancy, scalability and availability. Just as clearly traditional file-centric organization of data faces some challenges to meet these requirements. However databases have dealt with these requirements successfully for years.

3 Point Clouds and Databases

The simplest approach to store LiDAR point clouds in a relational database, would be to store every point in a row of a three column table where the columns represent X, Y and Z. Further columns could represent additional attributes (see Table 1). As (12) has mentioned classic relational databases are not cable to store hundreds of billions of rows for performance reasons. However, this would be necessary as follows from the examples above. Classic databases can maybe store millions of rows. There have been nevertheless efforts to approach this. The solution is typically to collect a larger set of points and store them as a single object in a row. The two major examples for this are Oracle Spatial and

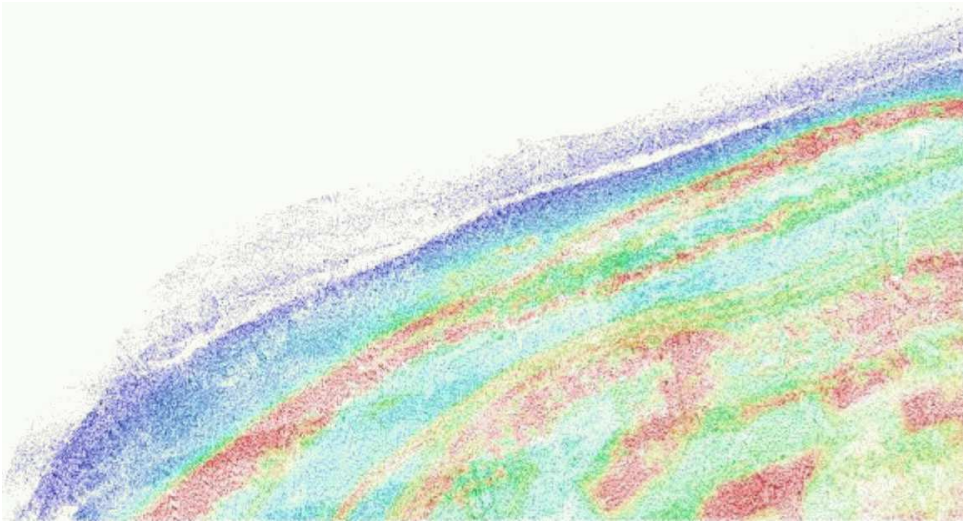


Fig. 1: Visualization of a single point cloud tile stored in a LAS file. The colours indicate height.

PostGIS. PostGIS refers to this concept as point cloud patches (PcPatch). The obvious disadvantage is that to access the actual geometry, i.e. the individual points you need to unpack these patches (PC_Explode) and casted to classic GIS points, which is an additional operation. For PostGIS the recommendation is to use patches with a maximum of 600 points, i.e. rather small patches. Googles Bigtable (3) finally promised to break the storage boundaries of traditional databases. According to the authors Bigtable was designed as a distributed database system to hold petabytes of data across thousands of commodity servers. The number of rows in a database is virtually unlimited. A Bigtable inspired open source distributed database system HBase was used as a storage backend for Megatree (11). Megatree is an octree like spatial data structure to hold billions of points. It is now maintained by hiDOF (6). NoSQL databases depart from the idea of storing data in tables. A significant portion of NoSQL databases (mongodb, couchbase, clusterpoint) are instead document oriented. If one was to draw a comparison to relational databases documents were the equivalent to rows in a table. A collection of documents then makes up the table. The decisive difference is that the documents in a collection need not follow the same schema. They can contain different attributes while the database is still able to query across all documents in a collection. These NoSQL databases are highly scalable and are one of the most significant tools for Big Data problems. At the time of this writing we are not aware of a system that uses NoSQL to store large LiDAR point clouds. We sketch a possible solution that follows a file-centric approach in the following section.

4 NoSQL Database for File-centric Storage

The central idea for a file-centric storage of LiDAR point clouds is the observation that large collections of LiDAR data are typically delivered as large collections of files, rather than single files of terabyte size. This split of the dataset, commonly referred to as tiling, was usually done to accommodate a specific processing pipeline. It makes therefore sense to preserve this split.

A document oriented NoSQL database can easily emulate this data partitioning, by representing each tile (file) in a separate document. The document stores the metadata of the tile. Different file formats could be accommodated by different attributes in the document,

as NoSQL does not enforce a strict schemata. The actual files cannot efficiently be stored inside a document as they are too large. A different mechanism is needed. We choose to use MongoDB a highly scalable document oriented NoSQL database. MongoDB offers GridFS which emulates a distributed file system. This brings the possibility to store large LiDAR files over several servers and thus ensures scalability. GridFS is a database convention to enable file storage. A file is split up into smaller chunks which are stored in separate documents linked via a common id. An index keeps track of the chunks and stores the associated file attributes. The idea to store large geospatial collections in a distributed file system is not dissimilar to Spatial Hadoop which uses HDFS for this purpose (4). Figure 2 gives an overview of the proposed architecture of the database. Figure 3 details the attributes that are stored in a document. Note that this is not meant to be a fixed schema, it is rather a minimal set of information which can be easily extended.

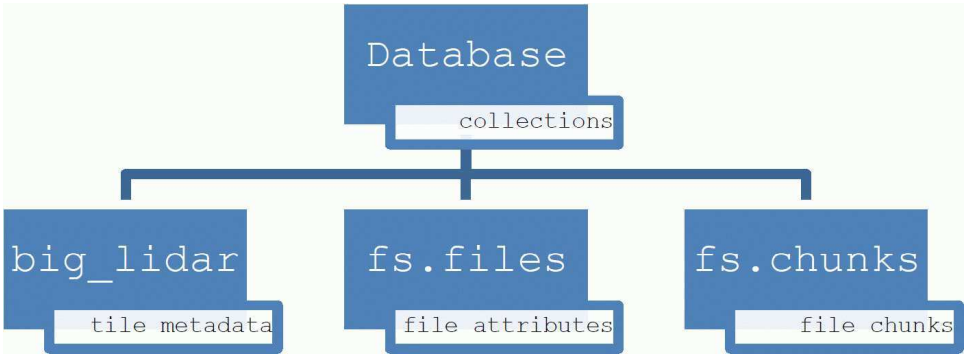


Fig. 2: Overview of the three collections that make up the database for big LiDAR data

5 Implementation Details

We start from the data provided by the LAS files. The information in their headers provides the crucial metadata for later queries. We use liblas (2) and its python bindings to parse the files. While we show all code excerpts in python for brevity there is nothing language specific in the data organization. The following python code excerpt shows an example of metadata that can be extracted. In this example we use the files signature, the version of LAS, the project ID and the date to describe the files content. We also extract the minimum and maximum of coordinate values to later construct the bounding box

```
# open LAS/LAZ file
file_name = 'g01cz1.laz'
f = file.File(file_name, mode='r')
header = f.header
# read meta data from header
s = header.file_signature
v = header.version
i = header.project_id
d = header.date
min = header.min
max = header.max
```

Many of the NoSQL databases target web and mobile development. Hence their geospatial indices are often restricted to GPS coordinates, which are most commonly represented

in WGS84. LiDAR data on the other hand is usually locally projected. Therefore any coordinates extracted from a LiDAR file need to me transformed to the correct coordinate system supported by the database. This is a very common operation in GIS. We use the PROJ library for this purpose. Again we provide some sample code which shows the transformation from the original coordinate systems (Amersfoort / RD New to WGS84 in this case). As you can see we only transform the bounding box of the data. The actual LiDAR data remains untouched.

```
p1 = Proj('+proj=sterea
+lat_0=52.15616055555555
+lon_0=5.38763888888889
+y_0=463000 +ellps=Bessel
+units=m +no_defs')
p2 = Proj('+proj=longlat +ellps=WGS84
+datum=WGS84 +no_defs')
min = transform(p1, p2, min[0], min[1])
max = transform(p1, p2, max[0], max[1])
loc = {"type": "Polygon",
"coordinates" :
[[[min[0], min[1]],
  [max[0], min[1]],
  [max[0], max[1]],
  [min[0], max[1]],
  [min[0], min[1]]]
]
}
```

For all of the above the actual data content of the LiDAR file never gets read. This is important to avoid unnecessary overhead. The file gets stored in full and unaltered into the database. As mentioned above MongoDB provides a distributed file system for this called GridFS. We show in code below how the compressed LAS file gets stored into the database. We store a pointer to the file (a file ID) to connect it to the metadata in the next step. We have now all the information in place to generate a document which combines the meta data of the LiDAR file, the geometric key and a pointer to the actual data content in the database (see Figure 3). This NoSQL document represents one tile of the collection of LiDAR data. The document is represented as a BSON object. This representation is very similar to the well-known JSON representation, but optimized for storage. The actual creation of the document and its storage are very simple. The next code sample shows all that is required.

```
# open mongodb big_lidar database
client = MongoClient()
db = client.test_database
big_lidar = db.big_lidar
# add file to GridFS
file = open(file_name, 'rb')
gfs = gridfs.GridFS(db)
gf = gfs.put(file, filename = file_name)
file.close()
# add one tile to big_lidar database
tile = {"type": s, "version": v, "id": i,
"date": d, "loc": loc,
"filename": file_name,
"gridfs_id": gf}
big_lidar.insert(tile)
```

6 Spatial Query

MongoDB like any NoSQL database allows for queries on the attributes of the document. When an index is created on a certain attribute queries are accelerated. As a specialty MongoDB allows spatial indices and spatial queries. Hence we can perform spatial queries on the bounding boxes of the LiDAR tiles. We show a very simple example of an interactive query on the Python console. The query uses a polygon to describe the search area. The database returns the tile metadata including the GridFS index. Using the GridFS index the actual point cloud data can be read from the database. It is simply the LiDAR point cloud file that was previously put into the database. It is important to note that no conversion or alteration of the files was done.

```
>>> tiles=list(big_lidar.find({ "loc" :
  { "$geoWithin" : { "$geometry" :
    {"type": "Polygon", "coordinates" :
      [[[5.1, 53.3], [5.5, 53.3],
        [5.5, 53.5], [5.1, 53.5],
        [5.1, 53.3] ]]}}}
    }))
>>> len(tiles)
4
>>> tiles[0]['filename']
u'g01cz1.laz'
>>> gf_id = tiles[0]['gridfs_id']
>>> gf = gfs.get(gf_id)
>>> las_file = open('export_' + file_name,
'wb')
>>> las_file.write(gf.read())
>>> las_file.close()
```

In Figure 4 we give the visualization of the four tiles recovered by the spatial query. Since the bounding boxes are stored as BSON objects, it is straight forward to export them as GeoJSON files. In the example we attach the filename as attributes and plot the bounding boxes over a base map using the filenames as labels. The leftmost tile corresponds to the point cloud visualized in Figure 1.

7 Conclusions

We have presented a file-centric storage and retrieval system for large collections of LiDAR point cloud tiles based on scalable NoSQL technology. The system supports spatial queries on the tile geometry. Inserting and retrieving files on a locally installed database is comparable to native file access speed. At the moment we are working on finding the best parameters for chunk size of the files and number of nodes for the servers to optimize performance and memory use for the terabyte sized AHN2 dataset. Building the system on MongoDB, a proven NoSQL database, brings in a range of advantageous features such as

- Scalability
- Replication
- High Availability
- Auto-Sharding

MongoDB supports Map-Reduce internally for database queries. However it is also known to work with external Map-Reduce frameworks such as Hadoop. A special adapter to access

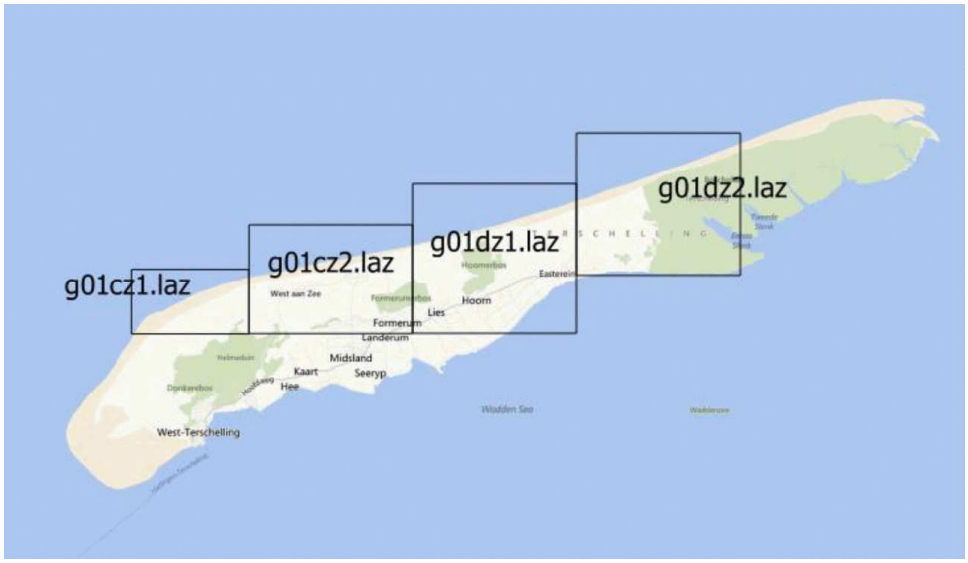


Fig. 3: Example GeoJSON output visualized using QGIS. The MongoDB spatial query delivered four tiles on the coast of the Netherlands. The GeoJSON contains the bounding boxes and the file names are used as labels. The actual point cloud is written in a separate LAS file.

MongoDB from Hadoop is provided. This offers very interesting future opportunities to combine Map-Reduce based processing with NoSQL spatial queries.

Acknowledgements We would like to acknowledge that this work is in part supported by an Amazon AWS in Education Research Grant award. Also part of this work is supported by EU grant FP7-ICT-2011-318787 (IQmulus).

References

[1] AHN website, (WWW Document), 2014. URL http://www.ahn.nl/ (accessed 5.19.14).
[2] Butler, H., Loskot, M., Vachon, P., Vales, M., Warmerdam, F., 2010. libLAS: ASPRS LAS LiDAR Data Toolkit [WWW Document]. IbLAS - LAS 101112 ASPRS LiDAR Data Transl. Toolset. URL http://www.liblas.org/
[3] Chang, F., Dean, J., Ghemawat, S., Hsieh, W.C., Wallach, D.A., Burrows, M., Chandra, T., Fikes, A., Gruber, R.E., 2008. Bigtable: A distributed storage system for structured data. ACM Trans. Comput. Syst. TOCS 26, 4.
[4] Eldawy, A., Mokbel, M.F., 2013. A demonstration of SpatialHadoop: an efficient mapreduce framework for spatial data. Proc. VLDB Endow. 6, 12301233
[5] Esri Introduces Optimized LAS (WWW Document), 2014. URL http://blogs.esri.com/esri/arcgis/2014/01/10/esri-introduces-optimized-las/ (accessed 5.19.14).
[6] hiDOF (WWW Document), 2014. URL http://hidof.com/
[7] Huber, D., 2011. The ASTM E57 file format for 3D imaging data exchange, in: IS&T/SPIE Electronic Imaging. International Society
[8] Isenburg, M., 2013. LASzip: lossless compression of LiDAR data. Photogramm. Eng. Remote Sens. 79, 209217.
[9] Isenburg, M., Schewchuck, J., 2007. LAStools: converting, viewing, and compressing LIDAR data in LAS format. Available: http://www.Cs.Unc.Edu/Isenburglastools.

[10] LAS Specification Version 1.3, 2009.
[11] Megatree - ROS Wiki (WWW Document), 2008. URL http://wiki.ros.org/megatree
[12] Ramsey, P., 2013. LIDAR In PostgreSQL With Pointcloud
[13] Swart, L.T., 2010. How the Up-to-date Height Model of the Netherlands (AHN) became a massive point data cloud. NCG KNAW 17.