MULTIVARIATE SURFACE GENERATION–VERSION 3

Deliverable D4.4.3

Circulation:	PU - Public
Lead partner:	SINTEF
Contributing partners:	CNR-IMATI, IGN, UCL
Authors:	Vibeke Skytt (SINTEF), Michela Mortara (CNR-IMATI), Laurent Caraffa (IGN), Murat Yirci (IGN), Christian Alis (UCL)
Quality Controller:	Roderik Lindenbergh (TU Delft)
Version:	1.0
Date:	28.04.2016

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	29.04.2016	100%	Final version

[1] Integers correspond to submitted versions

1 EXECUTIVE SUMMARY

This deliverable describes the third version of the IQmulus *Multivariate Surface Generation Toolbox* and is deliverable D4.4.3 in the IQmulus project. In the final period of Task 4.4 13 new services have been developed. The services vary in complexity from simple services organizing metadata for IQmulus work flows to innovative and complex functionality. Several of the new services are new versions of already existing services described in D4.4.1 and D4.4.2, which are prepared for execution on tiled data sets. The services are described with respect to functionality and algorithm, quality, scalability and degree of human intervention. Two services have been subject to major updates and are also described in detail.

A major motivation for the work in this project period has been to complete the IQmulus work flows defined in D1.2.3 and D1.2.4 with relevant Task 4.4 functionality. In addition, strategies for handling "big data" have been in focus.

The toolbox now contains 29 services including services related to point cloud approximation with LR B-spline surfaces, triangulations, watertight surface reconstruction, rainfall approximation, organization of metadata for work flows and utility functionality.

2 TABLE OF CONTENTS

3 TABLE OF FIGURES

4 INTRODUCTION

In this introduction, we give a short overview of the work performed within Task 4.4 that lead to this final toolbox report.

4.1 MOTIVATION OF SERVICE DEVELOPMENT

This is the last deliverable for Task 4.4. The motivation for choosing to develop a new service has changed slightly in the three project periods covered by the task. In the first period, the main emphasize was to create a broad foundation for the continued work. The service selection was mainly based on the collected user stories. However, there was also an intention to open potential paths for development and take benefit of configurations where added functionality could be obtained with relatively small effort by altering some aspects of a service to create a new one. Variations of the same theme can be seen in the services 9 and 55 which approximates point clouds and rasters with LR B-spline surfaces, and in 49 and 51 where the same type of input is converted to triangulated surfaces. One example of a potential path that has not been developed further is the construction of fully 3D LR B-spline surfaces. Services developed to support this approach are 56 and 84, and 3D spline approximation is also supported by service 9. The approach is interesting when dealing with non-projectable or hardly projectable point clouds, but the development was stopped after the first period due to other priorities.

In the second project period, the initial showcases had been revised and a number of detailed work flows had been defined (D1.2.3). In addition, the constantly existing focus on large data sets had been further emphasized. Service development became more focused on creating complete service chains supporting the work flows and to prepare new and existing services for large data sizes.

The development direction from the second period has been continued and further sharpened in the last project period, which is presented in this report. In the two first periods, substantial research has been performed in T4.4. This applies to LR B-spline approximation (Service 9 and related services), Service 48 and watertight surface reconstruction (Service 95). With a couple of exceptions, the focus in the last period has been to complement already started effort and to create the final toolbox. The exception is a service computing point cloud normal (Service 63) and deconfliction (Service 123) which is essential in work flow MS1. The deconfliction approach depends heavily on previous work on approximation of point clouds with LR B-spline surfaces, handling of large data sets by tiling and stitching and tailored visualization to be able to control the quality of deconfliction decisions. Thus, this work could not be started until the ground was sufficiently prepared. Most of the new services implemented in this period can be related to the deconfliction work flow (MS1).

Finally, the aspect of exploitation has been more prominent in directing the work towards the end of the project.

4.2 NEWLY RELEASED SERVICES

The services that have been released in the last period are with some exceptions parallel versions to previously released services. The new versions are designed to operate on tiled data in a parallel environment. It is a matter of definition whether these services are categorized as new or updated, but as the non-tiled versions of these services are kept as separate services, the tiled version will be classified as new. In addition, some services have been defined to complete

work flows. A new utility service computing normal from point clouds is also included. An overview of the new services is given in Table 1. Details on the new services can be found in Section 9.

TABLE 1: NEWLY RELEASED SERVICES

Id	Service Name	Lead	Showcase	Comment
48	Confidential – see Appendix B	CNR IMATI		
63	Extracting surface normals from point clouds	UCL		General purpose utility service
107	Confidential – see Appendix B	CNRIMATI		
117	LR B-spline field to triangulation	SINTEF	LS2	To complete work flow
122	Point cloud tile to LR B-spline surface	SINTEF	MS1/MS2	Parallel version of service 9
123	Deconfliction	SINTEF	MS1	Essential in work flow
124	Update spline surface with point tile	SINTEF	MS1	Parallel version of service 57
125	Trim LR B-spline surface for one tile	SINTEF	MS1/MS2	Parallel version of service 90
126	Distance field surface versus point cloud tile	SINTEF	MS1/MS2	Parallel version of service 88
128	Define regular tiling	SINTEF	MS1/MS2	Preparation for work flow
129	Perform regular tiling	SINTEF	MS1/MS2	Preparation for work flow
136	Metadata for data surveys	SINTEF	MS1/MS2	Preparation for work flow

4.3 UPDATED SERVICES

Most services have, in the last period, been updated to satisfy requirements from scalability testing or bug fixing. The interfaces of some services have been modified to obtain better integration in the associated work flow. This is considered as normal maintenance and will not be specified in this report. The services particularly mentioned in Table 2 and in Section 8 have been subject to major algorithmic updates.

TABLE 2: UPDATED SERVICES

Id	Service Name	Lead	Showcase	Comment
95	Watertight surface reconstruction	IGN	US	
110	3D Delaunay triangulation	IGN	US	

4.4 SERVICE CHARACTERISTICS & RELEASE STATUS

In deliverable D4.1.3, an updated procedure for service development and release was documented. In this procedure, the target platform (Ubuntu 14.04) for the executable was specified as well as the service repository and a standard for testing, including a so-called Jenkins test. In Table 3 an overview the release statuses of the T4.4 services is presented. The same list of services can be found in Section 4.5 with additional information.

Several services are prepared to be executed at different nodes working on tiled data or various time steps, but the services themselves run on a single node in a single or multi threaded fashion. The actual parallelization is performed by the IQmulus infrastructure. A number of services occur in pairs, one single node service for relatively small data sets and one service adapted for tiled data sets. Examples of such service pairs are (9, 122) and (90, 125), see also the information in Table 1. Note that some services have passed the Jenkins testing, but are still not viewed as released. In these cases, there is ongoing work to adapt the services to be run in a work flow using the infrastructure. This process may require an update of the service. The service will be released once this adaption is completed.

TABLE 3: SERVICE CHARACTERISTICS

Id	Language	Ubuntu 14.04	Execution	Metadata available	Artifactory uploaded	Jenkins test run	Released
9	C++	Yes	Multi threaded, single node	Yes	Yes	Yes	Yes
40	C++, Fortran		Multi threaded, parallelize on time step	Yes	Yes	Yes	Yes
48	Confidential – see Appendix B						
49	C++	Yes	Single node	Yes	Yes	Yes	Yes
51	C++	Yes	Single node	Yes	Yes	No	No
55	C++	Yes	Single node	Yes	Yes	Yes	Yes
56	C++	Yes	Single node	Yes	Yes	Yes	Yes
57	C++	Yes	Single node	Yes	Yes	Yes	Yes
58	C++	Yes	Possible to parallelize on time step	Yes	Yes	Yes	Yes

63	Python	Yes	Fully parallel	Yes	Yes	Ongoing*	No
66	C++	Yes	Parallelize on tiles, multi threaded	Yes	Yes	Ongoing	No
67	Matlab / Octave	Yes	Possible to parallelize on time step	Yes	Yes	Yes	Yes
84	C++	Yes	Single node	Yes	Yes	Yes	Yes
85	C++	Yes	Single node	Yes	Yes	Yes	No
86	C++	Yes	Multi threaded, single node	Yes	Yes	Yes	Yes
88	C++	Yes	Multi threaded, single node	Yes	Yes	Yes	Yes
90	C++	Yes	Single node	Yes	Yes	Yes	Yes
91	C++	Yes	Single node	Yes	Yes	Yes	Yes
95	C++	Yes		Yes	Yes	Ongoing	No
107	Confidential – see Appendix B	IMATI					
110	C++	Yes		Yes	Yes	Ongoing	No
122	C++	Yes	Parallelize on tile, multi threaded	Yes	Yes	Yes	No
123	C++	Yes	Parallelize on tile	Yes	Yes	Yes	No
124	C++	Yes	Parallelize on tile, multi threaded	Yes	Yes	Ongoing	No
125	C++	Yes	Parallelize on tile	Yes	Yes	Yes	No
126	C++	Yes	Parallelize on tile, multi threaded	Yes	Yes	Yes	No
128	C++	Yes	Single node	Yes	Yes	Yes	No

129	C++	Yes	Parallelize on data survey	Yes	Yes	Yes	No
136	C++	Yes	Single node	Yes	Yes	Yes	No

* Service 63 is waiting for necessary infrastructure adaptions for Spark services.

4.5 SUMMARY OF THE FINAL TOOLBOX

This section contains a complete overview of all services being developed in Task 4.4. The detailed description of the services can be found in the reports referenced to in the table below.

TABLE 4: FINAL TOOLBOX

Id	Service Name	Lead	Showcase	Released
9	Spline surface from point cloud	SINTEF	MS4	D4.41/D4.4.2
40	Rainfall using kriging	CNR IMATI	LS2	D4.4.1/D4.4.2
48	Confidential- see Appendix B	CNR IMATI		
49	Constrained triangulation	CNR IMATI	LS1	D4.4.1
51	Triangulation from raster	CNR IMATI		D4.4.1
55	Spline surface from raster	SINTEF		D4.4.1
56	Parameterize triangulated points	SINTEF		D4.4.1
57	Update spline surface with point cloud	SINTEF		D4.4.1
58	Rainfall using splines	SINTEF	LS2	D4.4.2
63	Extracting surface normals from point clouds	UCL		D4.4.3
66	Point cloud dimensionality	IGN	US2	D4.4.1
67	Rainfall using radial basis functions	CNR IMATI	LS2	D4.4.1
84	Parameterization using spline surface projection	SINTEF		D4.4.1
85	Stitching of LR B-spline surfaces	SINTEF	MS1/MS2	D4.4.2
86	Classify points from surface distance	SINTEF		D4.4.2
88	Distance field with respect to surface	SINTEF	MS4	D4.4.2
90	Trim spline surface	SINTEF	MS4	D4.4.2
91	Spline surface to raster	SINTEF	MS4	D4.4.2
95	Watertight surface reconstruction	IGN	US	D4.4.2/D4.4.3

107	Confidential – see Appendix B	CNR IMATI		
110	3D Delaunay triangulation	IGN	US	D4.4.2/D4.4.3
122	Point cloud tile to LR B-spline surface	SINTEF	MS1/MS2	D4.4.3
123	Deconfliction	SINTEF	MS1	D4.4.3
124	Update spline surface with point tile	SINTEF	MS1	D4.4.3
125	Trim LR B-spline surface for one tile	SINTEF	MS1/MS2	D4.4.3
126	Distance field surface – point cloud tile	SINTEF	MS1/MS2	D4.4.3
128	Define regular tiling	SINTEF	MS1/MS2	D4.4.3
129	Perform regular tiling	SINTEF	MS1/MS2	D4.4.3
136	Metadata for data surveys	SINTEF	MS1/MS2	D4.4.3

4.6 WORK FLOWS

The IQmulus services are atomic functions that are designed to be chained up in work flows, and in the Land, Marine and Urban Scenarios, a number of work flows are pre described (D1.2.3 and D1.2.4). To be able to execute a work flow, information has to be passed from service to service. Moreover, a need for services with the sole purpose of preparing for other services arises. Examples of such services are 128 and 129 described in Section 9, which also describes the other services mentioned in this overview. They construct a tiling tailored for approximation of point clouds by LR B-splines.

Transferring information from one service to another in the IQmulus infrastructure is performed using files. Metadata are often described in the .json file format. Consider the deconfliction work flow (MS1) in the Marine Scenario. The core of this work flow is the deconfliction service (123) and surface generation (Services 122 and 124), but before any essential service can be applied, the information flow through the service chain must be established (Service 136). A potentially large data size is handled by tiling and stitching, thus the initial data sets must be tiled (Services 128 and 129) and the surfaces constructed for each tile must be stitched (Service 85) to produce a seamless result. The surfaces are by nature defined on a rectangular domain, thus the must be restricted to the domain defined by the input data sets (Service 125). To perform quality control on the deconfliction and surface generation result, Service 126 is applied to compute one or more distance fields that can be inspected in IQmulusViz. The result of the work flow is a regular set of LR B-spline surfaces and associated distance fields, but to adapt to a usage outside of IQmulus conversion functionality to a DEM is being prepared.

The metadata information flow is initiated in Service 136, maintained, and extended throughout the work flow. The service chain takes a number of overlapping data surveys as input and the number of points in the survey as well as the bounding box of each survey, yields the initial metadata information. This is later extended with information on the defined tile structure, the connection between data surveys, tiles and surfaces, and the connection to associated distance fields. In addition, statistics on approximation accuracy are stored in the metadata files.

The explanation above is related to the Marine Scenario, but also the Land and the Urban work flows use meta data to handle data collections.

5 LIBRARIES

5.1 APACHE SPARK

Apache Spark (M. Zaharia et al., 2010) is an open source cloud computing engine for large-scale data processing. It builds on top of the Hadoop Distributed File System (HDFS) and can integrate with Hadoop's cluster manager YARN. See D4.3.2 for more information on Apache Spark.

6 DATA SETS

6.1 BATHYMETRY DATA SURVEYS SOUTH OF BRITAIN

A total of 317 survey blocks off the coast of Britain originating from different type of acquisitions are used as test cases for the services belonging to the Marine Scenario work flows MS1 and MS2. The data sets vary spatially and with respect to acquisition time and each data set typically overlaps with many other data sets. When several data surveys are processed in combination this gives a very non-uniform point cloud, see Figure 1. Such point cloud collections can add up to large data sets. In the test examples shown in this report, the size of the collections range from 0.1 GB to 6.7 GB. The data surveys were originally represented in the WGS84 coordinate system, but have been transformed to UTM. One data survey from this collection has been presented in D4.4.2.

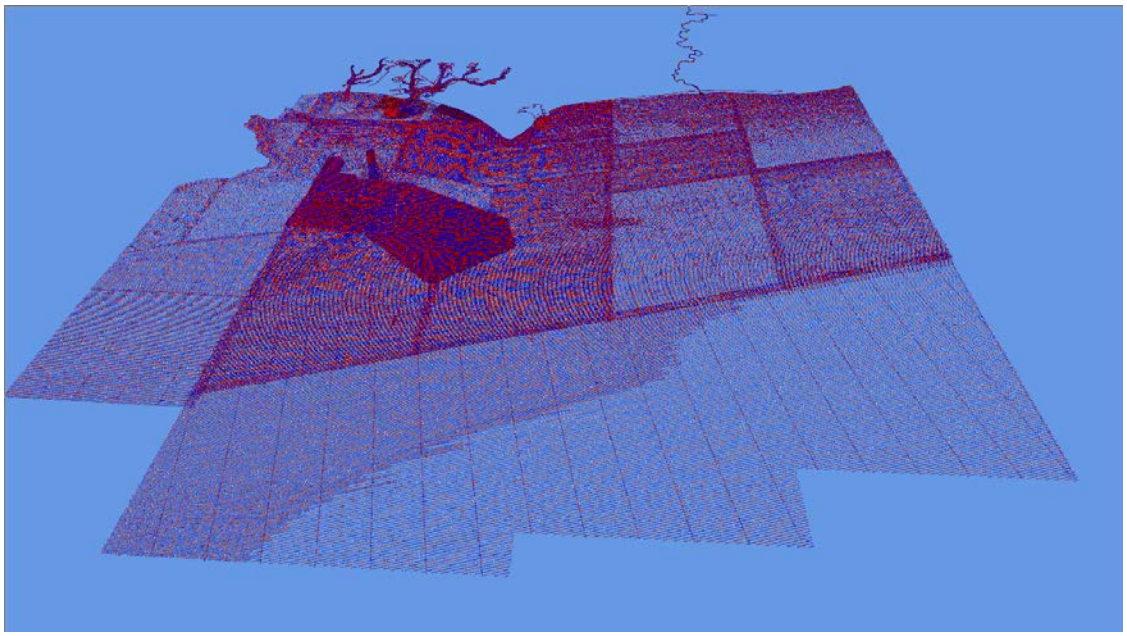


FIGURE 1: Data collection of size 0.57 GB consisting of 60 data sets. The density of the point cloud shows high variation and a typical pattern of scan lines also implies that the distance betwen two individual points can vary greatly.

6.2 DATA SURVEYS FOR DECONFLICTION

A collection of data surveys is equipped with priority scores to serve as a test case for the deconfliction work flow (MS1). The data sets are represented in WGS84 and given as csv-files with the following entries for each point: block identity, x, y, z, score. The data sets represent sea bottom bathymetry and have a similar configuration to the data collection presented in Section 6.1. The scores are numbers between 0 and 1 and are used to prioritize the data surveys in areas with conflicting data. Figure 2 shows one example of a test data set. The 15 survey blocks are coloured according to their priority score, green represents a low score while blue is high. A clear blue colour indicates a higher score than a more dimmed colour. The score difference is

often small resulting in very similar colours. The points have been scaled in x and y to facilitate the visualization.

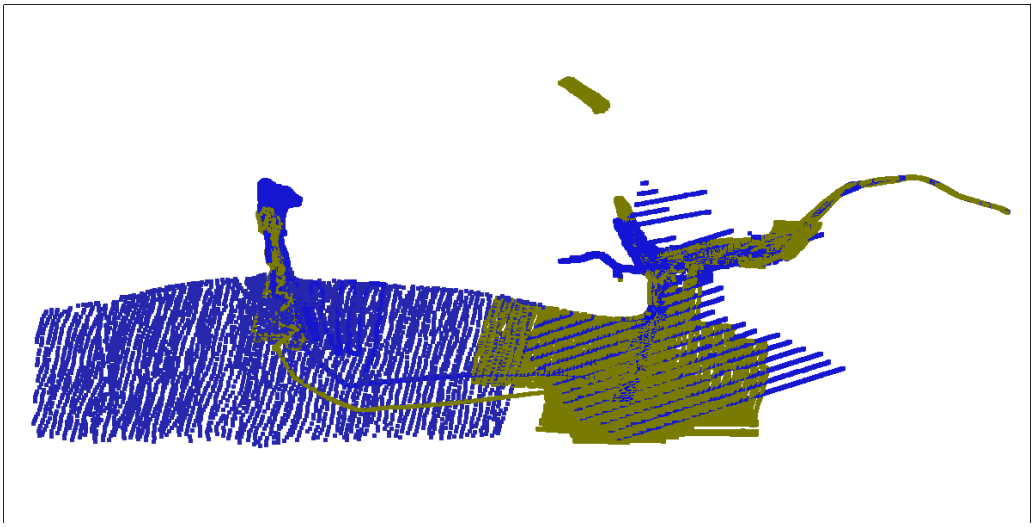


FIGURE 2: A set of 15 data surveys. In some areas, there are many overlapping surveys, other areas are covered with only one data set

6.3 BENCHMARK DATA FOR WATERTIGHT SURFACE GENERATION

The benchmark is introduced here: http://www.cs.utah.edu/~bergerm/recon_bench/. It allows evaluation of the surface according to both error metrics and topological aspects. The benchmark is dedicated to surface reconstruction from point clouds with normals. It consists of five data-sets generated from an implicit function with a synthetic scanner. For each dataset, 48 acquisitions are generated with different variations of the scanner's parameters like noise or different cameras positions, which lead to occlusions.

6.4 HEMISPHERICAL AND TRIANGULAR WAVE SYNTHETIC DATA

Synthetic data for benchmarking Service 63 is found under UCL/hemispherical and UCL/triangular on the Fraunhofer cloud. These data are stored as LAS files and range from 1000 (19.7 KB) to 1B points (18.6 GB). The hemispherical point cloud (Figure 3, left) is as a smooth point cloud while the triangular wave point cloud (Figure 3, right) is designed as a point cloud containing sharp, non-differentiable edges. These two data sets will be used to evaluate the performance of the normal extraction service, Service 63.

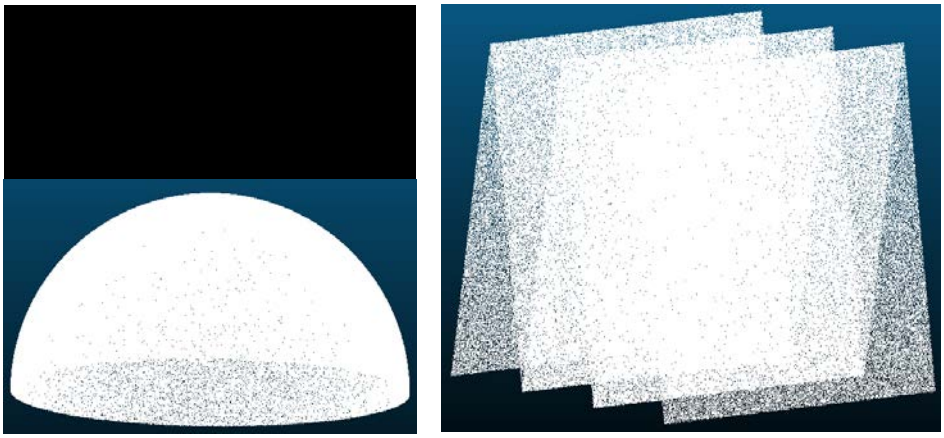


FIGURE 3: Rendering of hemispherical (left) and triangular wave (right) point clouds consisting of 1,000,000 points.

7 BIG DATA STRATEGY

TABLE 5: VARIETIES OF "BIG DATA"

Indicator	Comment	Relevant for
Volume	Modern data acquisition technology creates high volumes of data.	Services that face initial point clouds
Variation	Topographic and bathymetric data gained by different equipment and with different quality properties give heterogeneous point clouds. Coverage data related to physical or environmental phenomena	Services faced with point clouds from different sources. Services related to rainfall
Velocity	The data set changes in time due to changes in the source for the data acquisition	Rapid for rain falls, more slowly for moving sand dunes. Obstacles in the scene in the urban scenario.
Analytics	Represent the initial data in a way suitable for further processing	For instance deconfliction
Data reduction/ organization	Make acquired data more manageable by applying a different and more lean storage format or reorganize data for tailored access	Spline approximation, 48, watertight surface reconstruction

As indicated by Table 5, the concept "big data" can be interpreted in a number of ways. In D1.2.3, the IQmulus work flows where discussed with respect to most of these indicators. This section is mainly concerned with high data volumes, but it should be noted that several services are dedicated to represent the information carried by the large point clouds in a way that is more suitable for further processing. One example is approximation by LR B-splines, which normally leads to a considerable data reduction. Another example is service 48 discussed in Appendix B. Heterogeneous point clouds are typical for data sets sampling the sea bottom at different times as visualized in Figure 1 and Figure 2.

In the remainder of this section, we will discuss how large data volumes are addressed in the context of surface generation.

7.1 MULTI THREADING

A number of services are multi threaded, at least for critical operations. This reduces the execution times on nodes with several processors, but can increase the memory usage. Thus, use

of the multi threaded version is advisable only for moderately sized data sets. Use of multi threading can be combined with multi node execution.

7.2 REGULAR TILING IN THE CONTEXT OF LR B-SPLINES

The scalability of approximating point clouds with LR B-spline surfaces mainly depends on three factors that are partly interlinked:

- The number of points in the given data set. The behaviour of the approximation algorithm is roughly linear with respect to the number of points.
- The size of the produced LR B-spline surface. Since the structure of the polynomial patches is not tensor product like, the theory behind the LR B-spline format is outlined in [4], an LR B-spline surface has a more complex data structure than a tensor product spline surface. For small and moderately sized surfaces, the execution time for traversal of the data structure is neglectable. Such traversals are performed for instance when the surface is refined in the adaptive approximation algorithm applied in Service 9, 57, 122 and 124, see D4.4.1 and Section 9.5. When the surface grows large, the majority of the time spent in an operation is related to data structure traversals.
- The number of iterations in the adaptive approximation algorithm. Larger domains and more points implies that there is a possibility for more local variations in the shape of the terrain corresponding to the point cloud. This again imply that there is a need for more iterations to accurately approximate the point cloud and for each iteration the surface grows larger and more time is spent in traversing the data structure. The data size of a surface can as a maximum be multiplied with four for each iteration, but due to adaptivity a much smaller increase is expected and the surface growth will decrease during the execution.

Large data sets needs to be split to keep the data size under control for approximations and other operations, but even more to limit the size of the corresponding LR B-spline surface. The data set is split into regular tiles for two reasons:

- An LR B-spline surface is defined on a regular domain. Thus, tiles with rectangular and axis parallel domains reduce the need for surface trimming.
- A regular partitioning of the total domain of the data set makes it trivial to identify the tile a given (x,y) pair belongs to.

A consequence of using regular tiles in the (x,y)-domain is that the number of points for each tile will vary considerably.

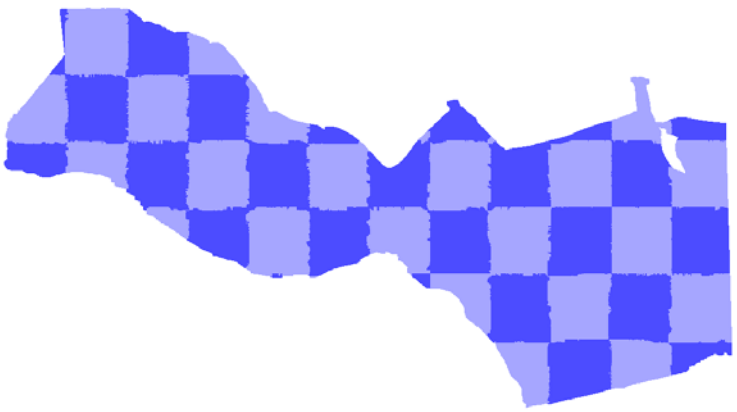


FIGURE 4: A large point cloud split into tiles in a chess board like fashion

Figure 4 shows the tiling of one large point cloud. It consists of about 65.5 million points. The tiles vary in size from 1497 points to slightly less than 2 million points.

The tiling is prepared and performed by the services 136, 128 and 129 described in Section 9. Several services are designed to

operate on one tile without any interference with other tiles. This provides the ground for parallelization on several nodes by the IQmulus infrastructure.

7.3 PARTITIONING

Confidential - See Appendix C

7.4 OUT-OF-CORE COMPUTATION FOR WATERTIGHT SURFACE RECONSTRUCTION

Large data sizes are handled by the following algorithm which is shown schematically in Figure 5.

Algoritm:

- The x,y space is discretized using a 2D grid and each point is aggregated to a chunk.
- A 3D Delaunay triangulation of the points in each 2D grid cell is determined.
- Neighbouring triangulations are stitched; extra points can be added in order to break big tetrahedrons (Service 110).
- Once the triangulation is computed, the score for each for each tetrahedron is computed.
- The segmentation of the triangulation is computed on each chunk followed by taking neighbour's chunks to ensure watertightness.
- Finally, the surface is extracted for each chunk.

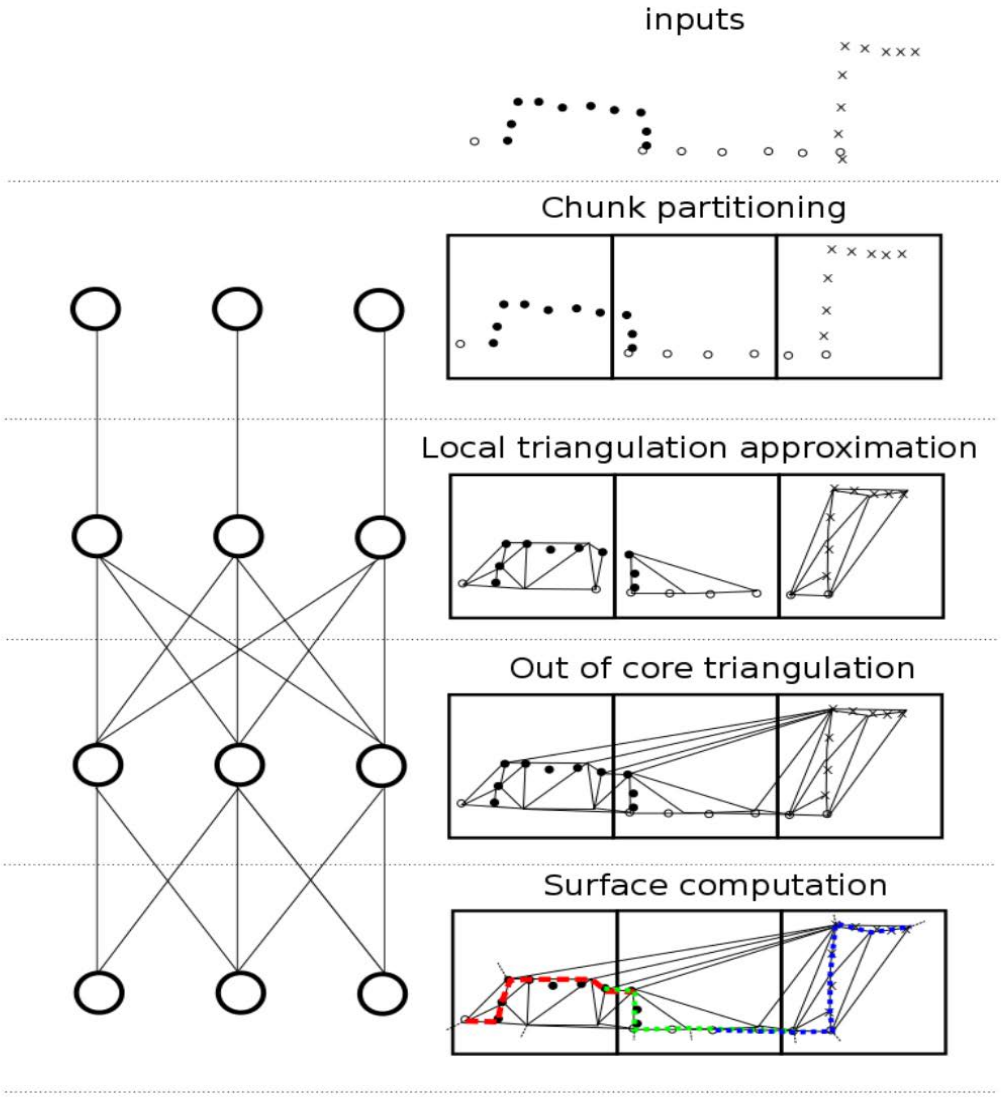


FIGURE 5: The main scheme for the approach to scale the surface reconstruction algorithm for "big data"

7.5 PARALLEL PROCESSING USING APACHE SPARK

A definition of Big Data could be that the data is to big fit on a single hard drive. Even if it does, transferring data from storage to each processing machine in a computing cluster over a network results in a huge performance penalty. A solution to these problems would be to split the data into different partitions, which may be stored across several processing machines. We want the data to be on the processing machine as much as possible, a concept known as data locality, to avoid slow network transfers.

Individual data points may be randomly assigned to different partitions to ensure parallel operations are balanced across processing machines. However, several operations on point clouds, such as extracting normals, require neighbourhoods of points as input. Thus, a method of partitioning point clouds such that points that are geometrically close would be assigned into the same partition is required.

For Service 63, points were assigned into a partition using a *Z*-order space-filling curve, which is a form of locality-sensitive hashing. The *Z*-order, also known as Morton code, of a point p is

assigned as follows. The bounding box is first divided into 2^b partitions per dimension (see Figure 6 for an illustration of a 2D case) and are then sequentially assigned a b-bit number. The number of the partition where p is found for each dimension is then interleaved and this becomes the Z-order code.

In the illustration, two bits (b = 2) were assigned per dimension hence the result is a grid of 2^b partitions × 2^b partitions. The index of the lower leftmost partition is 0000_2 = 0 whilst that of the upper rightmost partition is 1111_2 = 15. The point p is in coordinates (10_2, 01_2). Interleaving the coordinates, we get 0110_2 = 6 and this is the Z-order. Connecting the center of each partition sequentially based on the index would result to a Z-like curve hence the name Z-order.

Service 63 was implemented using the general-purpose Big Data processing engine Apache Spark, which can handle data stored across multiple machines. Users may write code in Java, Scala, Python and R, and call third party dynamically linked libraries using the facilities provided by the programming language of the user. This capability may save significant time and effort by adapting and using previously written code and compiled point cloud libraries. In Service 63, we use the Point Cloud Library (PCL) to extract normals after partitioning the input point cloud by Z-order.

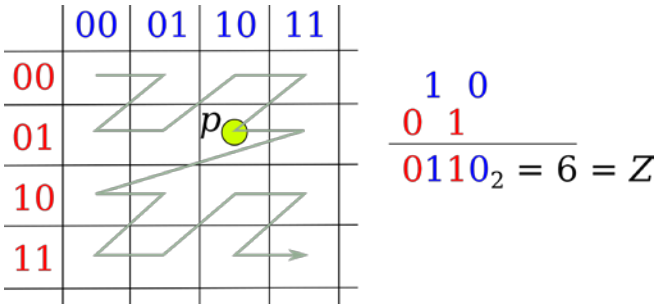


FIGURE 6: Z-order indexing

8 UPDATED SERVICES

Since the delivery D4.4.2, there has been considerable activity in the project on automated service testing and scalability testing. This implies that almost every service have been updated to output information required for this testing. These updates are taken as normal service maintenance and will not be described in this section. Other services have been updated to adapt to the information flow through a work flow. This typically changes the service interface. Service 85 (stitching of LR B-spline surfaces) is an example of such a service. However, the essence of this service is described in Section 5.6 in D4.4.2 and no major algorithmic modifications have taken place. This section is devoted to services that have been significantly modified since D4.4.2.

8.1 #95 WATERTIGHT SURFACE RECONSTRUCTION

8.1.1 Functionality & Algorithm

Surface reconstruction from a point cloud is an important problem that already has been studied extensively. The first reason is the ill-posed characteristic of the problem: for a set of input points, many surface configurations are possible.

The second reason of this interest is the constant increase of both the number of applications that use reconstructed surface (Digital elevation models, computation for flood simulation or path planning) and the data acquisitions sizes (Lidar, images...). These aspects result in multiple and various methods based on different assumptions related to data perturbations. It is against this background that we propose, instead of choosing a given data representation, surface prior and specific output, a method that takes generic input data, and produces a watertight surface of the scene with a confidence metric.

Key contributions are:

- Joint and unified handling of Lidar and photogrammetric range measurements as evidences that the 3D segment between sensor (camera or lidar) and object is free space and that space is occupied immediately behind the measured object
- Handling of heterogeneous sampling rates (airborne, MMS, static, image, lidar...) and location uncertainties of all measurements
- Production of a watertight triangulated surface with a confidence metrics associated to each triangle

Our formulation is based on the segmentation of 3D space. The goal of this approach is to segment the space into volumes that are inside (objects, trees, ground, buildings...) and outside (air). With only the information on the inputs and due to the ill-posed form of the surface reconstruction problem, some areas will be ambiguous (many surfaces can fit a set of point). In urban areas, planar areas must be encouraged. In order to remove the ambiguity and encourage planar surfaces, a global smoothness criterion that penalizes solutions with important surface (and consequently non-planar area) is added as prior. In this case, segmentation with a small interface between inside and outside area will be encouraged. A regularization coefficient is added to the prior term in order to choose how smooth the surface is. This leads to a global optimization formulation that can be solved efficiently with a graph-cut approach. Instead of producing a binary segmentation, the result is a segmentation of the space where each label is a confidence to be inside or outside an object. This gives a quality criterion for each area of the reconstructed surface. The algorithm is decomposed into three steps (see Figure 7):

- First, the scene is segmented in a tetrahedral mesh using the Delaunay triangulation.
- Then for each cell, a score that shows the confidence of the cell to be inside or outside is computed (c).

- The global optimization problem is minimized with the alpha-expansion algorithm (d).

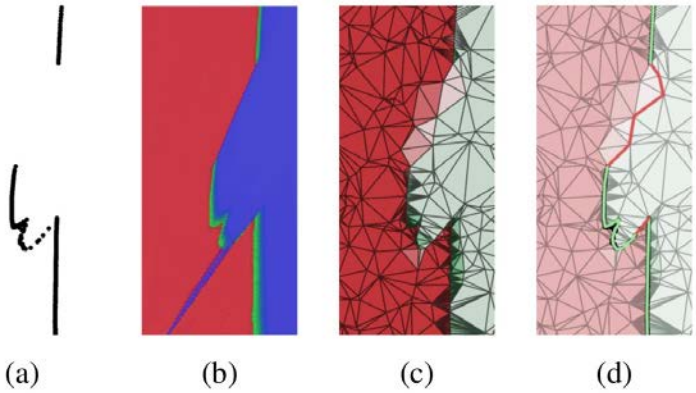
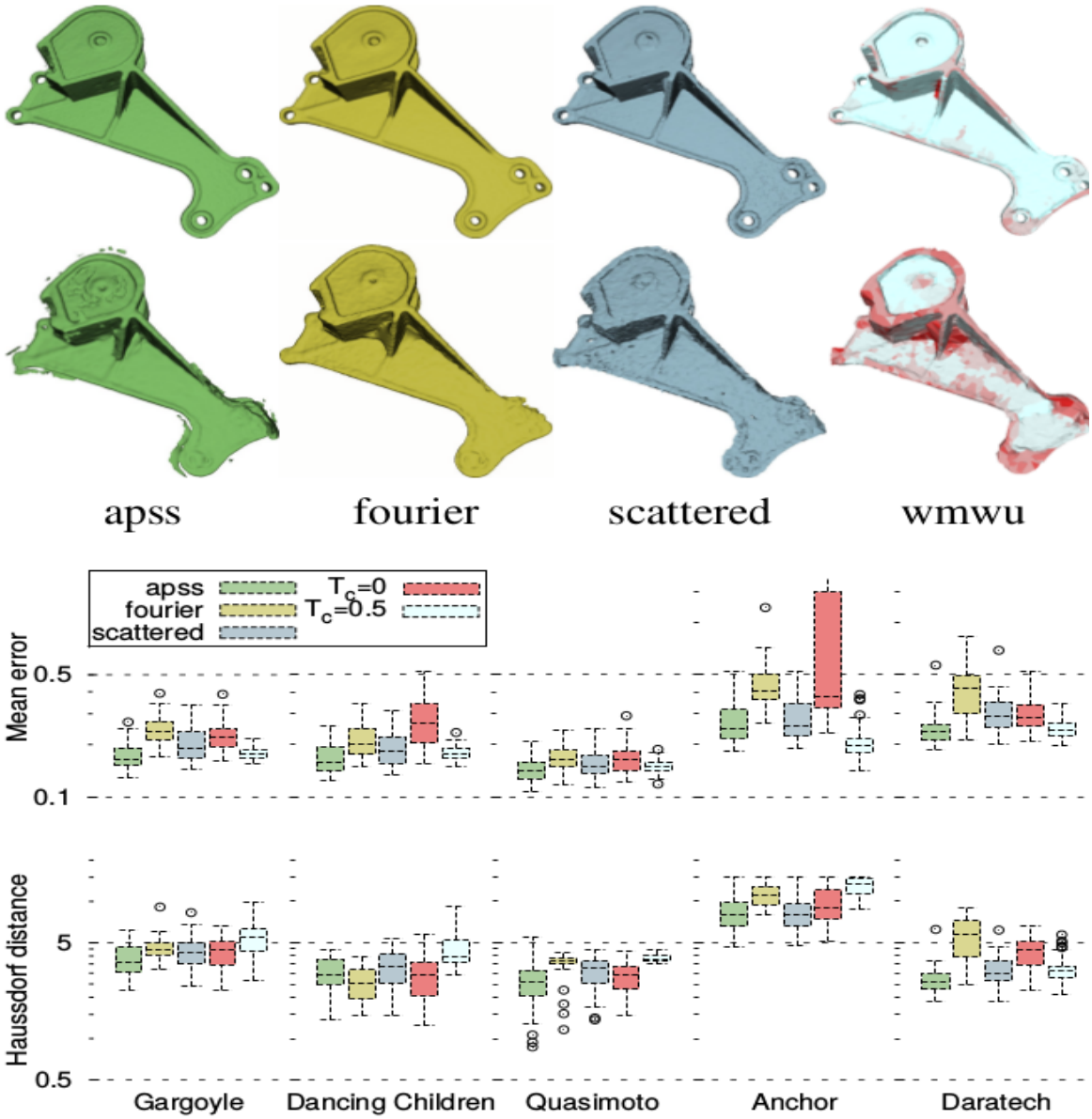


FIGURE 7: Steps in segmenting the space between inside and outside of an object.

8.1.2 Quality

The multilabelled segmentation enables the output of a confidence metrics on each triangle of the reconstructed surface that states to which degree using this triangle in the final mesh was a clear decision.

The method is evaluated on the benchmark data presented in Section 6.3. To enable the comparison to other methods, these data sets are selected even though they are obtained from the measurement of mechanical parts. In order to show the advantage of computing a multi-label segmentation, a single result is computed for each file. For that, we use a confidence threshold T_c. A facet is removed when the difference between two neighbouring tetrahedra's label is lower than T_c. The resulting surface is a non-manifold area. When Tc is small, all the facets are kept. When Tc is large, only facets with either strong confidence to be either inside an object or strong confidence to be outside are kept.



algorithm	comps	bndry	manifold	genu
apss	47.37	140.86	0.50	1.82
fourier	1.54	**0.00**	**1.00**	**0.49**
scattered	1.90	214.21	**1.00**	7.47
wmwu ($T_c = 0$)	1.00	**0**	**1**	0.61
wmwu ($T_c = 0.5$)	2.72	77.08	0.02	12.4

FIGURE 8: Result of 4 algorithms applied to the benchmark data presented in section 6.3: APSS, Fourier, scattered and the proposed approach.

In Figure 8 wmwu refers to our approach. The figure shows the global surface (wmwu t_c=0) and only confident areas (wmwu t_c=0.5). The first line show an easy case, in the second line the input data set is much noisier with occlusions. The error mean and the Hausdorff distance are compared on the five datasets. Black dots shows outliers when the box plot show the median and quartiles. The table shows the average shape properties over the full benchmark: « comps » refers to number of connected components, « bndry » is the length of boundary components, « manifold » is whether or not a mesh is manifold, 1 being a manifold and 0 otherwise, « genus » refers to the deviation from the actual genus. The first column refers to the algorithm being evaluated, state of the art and our.

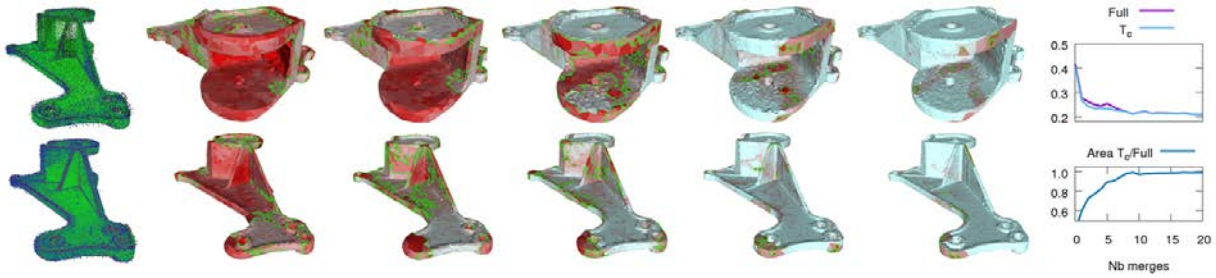


FIGURE 9: Surface reconstruction of many point clouds. The first image shows the fusion of all point clouds. Column 2 to 5 shows the result the surface reconstuction with 1, 2, 5, 10 and 20 point clouds.

The aim of the proposed approach is to merge a significant amount of data originating from different data sets. Figure 9 shows the result up to 20 point clouds processed at the same time.

8.1.3 Scalability

Many watertight surface reconstruction approaches are based on an indoor / outdoor segmentation of a Delaunay triangulation where the resulting surface is represented by the interface between inside and outside areas. When data sizes becomes important and data sets need to be processed separately, the watertightness of the surface cannot be guaranteed anymore. In order to produce a full watertight surface reconstruction, we use the scheme presented in Section 7.4.

As the local Delaunay triangulation approximation and the score computation steps are local to a chunk, is it fully parallelizable with regard to these chunks. The out of core triangulation step can theoretically be applied globally. We add extra points to the triangulation to break big tetrahedrons. Consequently, the out of core triangulation can be processed by taking only the current chunk and the neighbour's chunks into account. Thus, the approach is easily parallelizable. 3D triangulation is performed by Service 110. In this service, a score is computed for each chunk by integration of a mass function, and finally, the surface can be extracted. These computations are performed independently for each chunk thereby enabling a scalable algorithm.

We are currently working on a common representation for 3D triangulations that is adapted to big data. Compare also:
(https://github.com/posseidon/IQLib/blob/master/docs/specification/datamodel.md).

The proposed format is designed to store a set of simplicial meshes (2D and 3D) representing a big simplicial complex tiled according to some spatial or semantic rule.

8.1.4 Degree of Human Intervention

The proposed approach is fully automatic.

8.1.5 Examples

In this example, we show how out approach helps for modeling different types of data. Figure 10 shows an urban scene reconstruction with both airborne and terrestrial Lidar data. Two overlapping airborne Lidar acquisition stripes are for covering a wide area. A terrestrial point cloud is included at the center of the scene for a final 63,000,000 points processed.

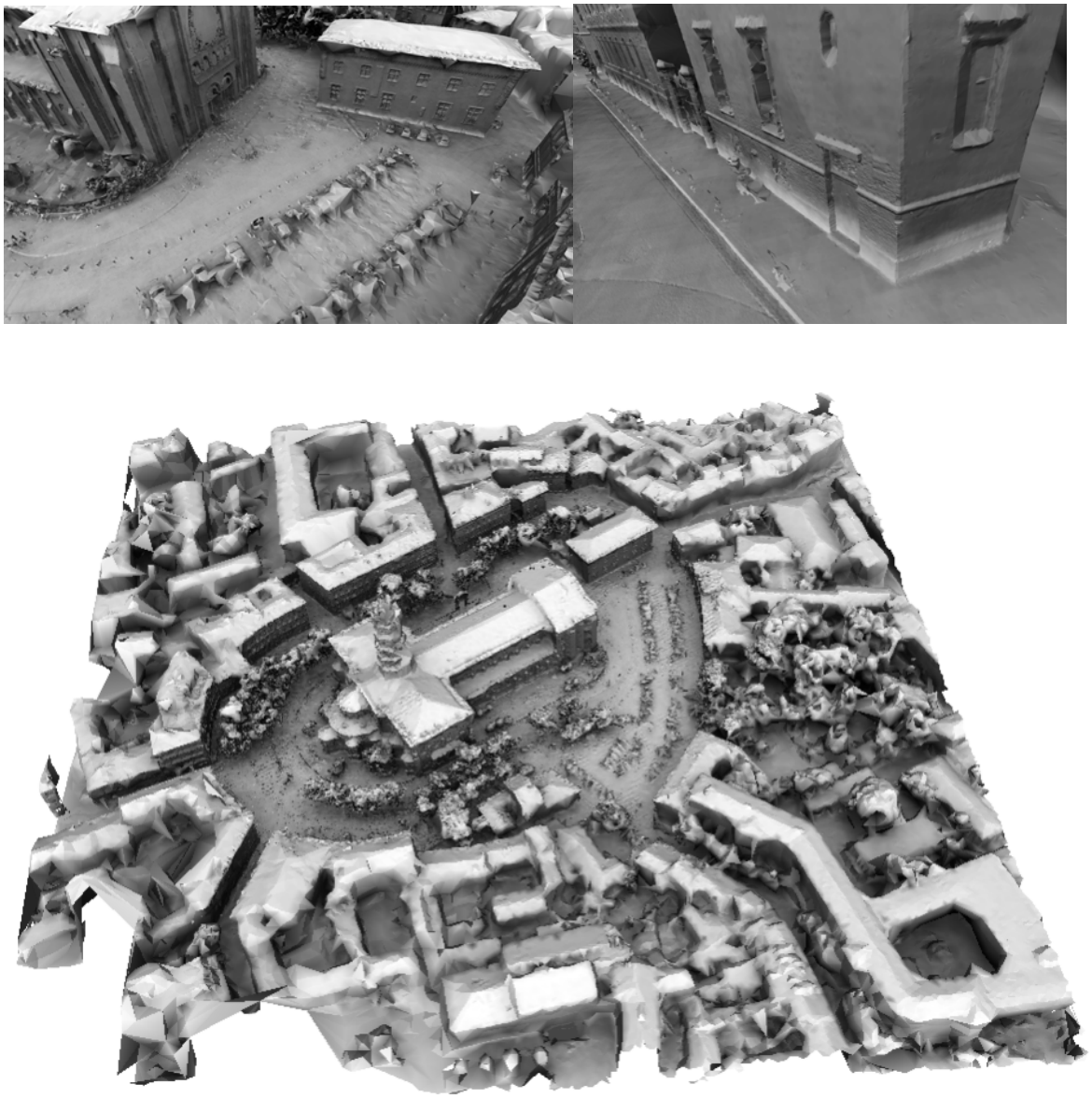


FIGURE 10: Joint reconstruction from airborne (3m points) and terrestrial (mms, 60.000.000 points) lidar of the same scene.

8.2 #110 3D DELAUNAY TRIANGULATION

The aim has been to design a new divide and conquer algorithm for 3D Delaunay triangulations which can be adapted to out-of core cloud computation.

8.2.1 Functionality & Algorithm

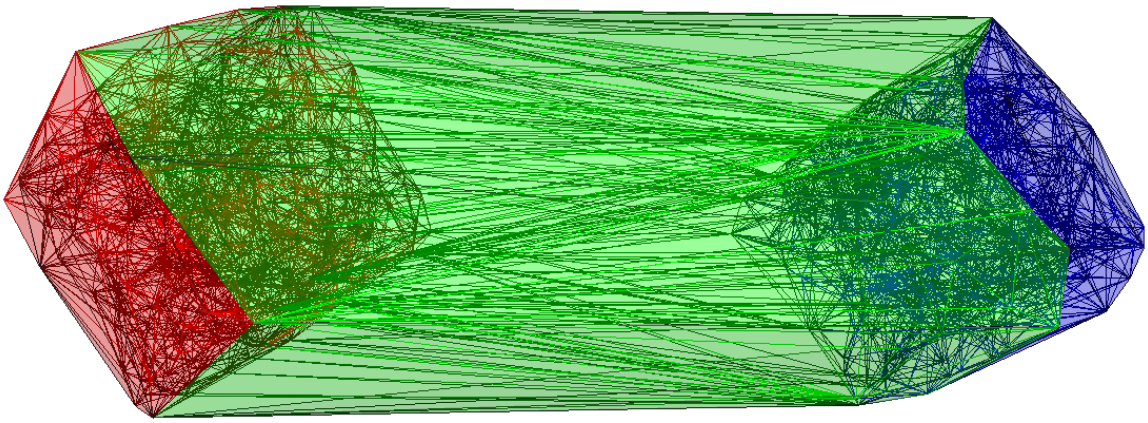


FIGURE 11: Merging of two Delaunay triangulations

Figure 11 shows an example where two Delaunay triangulations have been merged. In the following the algorithm is explained.

The geometry and shape of an object in 3D physical world are generally captured as a set of 3D points by typically scanning its surface. A 3D triangulation decomposes the convex hull of these points into a set of tetrahedra, which is used for further analysis and processing. Among the many possible 3D triangulations of a point set, a special type called the Delaunay triangulation is more popular for practical reasons. For instance, it reduces the containment radius of the tetrahedra (radius of the smallest sphere containing the tetrahedron) which is important in mesh generation for various applications such as finite element analysis (FEM) and surface reconstruction.

The Delaunay triangulation for a set P of points in 3D Euclidean space (R^3) is a triangulation DT(P) such that no point in P is inside the circumsphere of any tetrahedron (cell) in DT(P). Delaunay triangulations have been studied a lot and many different algorithms have been proposed. Recently, researches have started to focus on in core / out of core parallel implementations of the available algorithms.

8.2.1.1 Definitions

Oriented plane: A plane divides R^3 into two open half-spaces: positive and negative side of the plane. The positive side is the half-space pointed to by the normal of the plane.

Visibility of an oriented plane: Let Π be an oriented plane and p be a 3D point in R^3. Π is said to be p-visible if p lies on the positive side of Π.

Convex hull: The convex hull of a point set P is the smallest convex set that contains P. For instance, in 2D Euclidean plane (R^2), the convex hull of a point set is defined by a convex polygon (convex 2-polytope) and in R^3, the convex hull of a point set is defined by a convex polyhedron (convex 3-polytope). Notation: Conv(P).

Facets of a convex hull: The boundary of the convex hull of a d-dimensional point set is defined by (d-1)-dimensional facets. For instance, for a 3D convex hull in R^3, the facets are 2D convex polygons.

Visibility of a facet on the convex hull: A facet of a 3D convex hull is said to be visible from a 3D point p, if the oriented supporting plane of the facet is p-visible. The orientation of the supporting plane is selected such that the positive open half-space does not contain the convex hull.

Touching cell: A cell of a Delaunay triangulation is said to be touching if its circumsphere touches to a given plane Π.

In-sphere test: Given a point p in R^3 and a cell (tetrahedron) of a 3D triangulation, this test returns true if p lies in the interior of the circumsphere of the cell.

Conflicted cell: Let a triangulation T and a point set P are given in R^3. Then, a cell (tetrahedron) of T is said to be conflicted if the in-sphere test returns true for the cell and any point of P.

Delaunay neighbors: For a given Delaunay triangulation, two vertices are said to be Delaunay neighbours if there exists a Delaunay edge between the vertices.

8.2.1.2 Algorithm

Let a 3D point set P in R^3 be given. Also assume that R^3 is split into n convex regions by a set of planes such that in each region there exists at least four non-flat points. Therefore, P can be decomposed into n subsets (called tiles) corresponding to each split-region in R^3. That is, $P = \{P_1 \cup P_2 \dots \cup P_n\}$. Furthermore, let each tile is distributed to a node ($P_i \rightarrow N_i$) in the cloud. The task is the computation of the Delaunay triangulation of the point set (DT(P)). The proposed algorithm is composed of 5 iterations such that after each iteration data exchanges between different nodes are performed.

ITERATION-1: Runs on every node. Iteration steps are described with respect to an arbitrary node N_i in which the tile P_i resides.

1. Compute the DT(P_i).
2. Extract Conv(Pi) from DT(P_i) and save it to disk (File-N_i-1).
3. Send the saved file to node N. Node N is a predefined node.

ITERATION-2: Runs only on the node N. N is a predefined node and $N \notin N_1, N_2 \dots, N_n$.

1. Read the tile convex hulls from the received files (File-N_i-1 for i = {1, 2, 3, …, n}).
2. If the received convex hull points fit into the main memory for computing the 3D Delaunay triangulation, then compute the tile neighbourhood graph from the Delaunay triangulation. Each node of the tile neighbourhood graph corresponds to a tile and an edge between two nodes indicates the existence of at least one pair of points within the corresponding tiles that are Delaunay neighbours. If the convex hull points do not fit into the main memory, then subsample them (w.r.t user defined sampling parameter) to compute the neighbourhood graph.
3. Save the tile neighbourhood graph into disk and send it to every other node (N_i for i = {1, 2, 3, …, n}).

ITERATION-3: Runs on every node. Iteration steps are described with respect to an arbitrary node N_i in which the tile P_i resides.

1. Read the neighbourhood graph that is received from the node N. Assume that P_j denotes a random neighbour of P_i.
2. For each neighbouring tile of P_i perform the following tasks.
    - Compute the touching cells of DT(P_i) with respect to the splitting plane that

separates P_i and P_j. Let's call these touching cells as P_j-touching cells of $DT(P_i)$.

- Compute the vertices of $DT(P_i)$ that are incident to P_j-touching cells of $DT(P_i)$ and save them to disk (File-N_{i-j}-2).

Non-touching cells of $DT(P_i)$ will also be in $DT(P)$.

3. Send the $Conv(P_i)$ to all neighbouring tiles of P_i.
4. Send the vertices that are computed in step-2 (File-N_{i-j}-2) to the corresponding neighboring tiles.

ITERATION-4: Runs on every node. Iteration steps are described with respect to an arbitrary node N_i in which the tile P_i resides. Furthermore, assume that P_j denotes a random neighbour of P_i.

1. For each neighboring tile,
   - Read the $Conv(P_j)$ from the received File-N_j-1.
   - Compute the visible facets of $Conv(P_j)$ with respect to the points of $Conv(P_i)$. Let us call these visible facets as P_i-visible facets of $Conv(P_j)$.
   - Compute the vertices of $Conv(P_j)$ that are incident to P_i-visible facets of $Conv(P_j)$.
2. For each neighbouring tile,
   - Read the vertices of $DT(P_j)$ that are incident to P_i-touching cells of $DT(P_j)$. These vertices are received from $DT(P_j)$ within the File-N_{j-i}-2.
3. For each neighbouring tile,
   - For each P_j-touching cell of $DT(P_i)$ perform the in-sphere test with respect to the vertices that are computed / loaded in steps 1 and 2. If any of the tests return true, then the corresponding P_j-touching cell is classified as conflicted. Let's call these cells as P_j-conflicted cells of $DT(P_i)$.
   - Compute the vertices of $DT(P_i)$ that are incident to P_j-conflicted cells of $DT(P_i)$ and save them to disk (File-N_{i-j}-3).

The non-conflicted cells of $DT(P_i)$ will be in $DT(P)$.

4. Send the vertices that are computed in step-2 (File-N_{i-j}-3) to the corresponding neighboring tiles.

ITERATION-5: Runs on every node. Iteration steps are described with respect to an arbitrary node N_i in which the tile P_i resides. Furthermore, assume that P_j denotes a random neighbour of P_i.

1. For each neighbouring tile,
   - Read the vertices of $DT(P_j)$ that are incident to P_i-conflicted cells of $DT(P_j)$. These vertices are received from $DT(P_j)$ within the File-N_{j-i}-3.
   - Insert the vertices of $DT(P_j)$ into $DT(P_i)$ that are incident to P_i-conflicted cells of $DT(P_j)$ and that are incident to P_i-visible facets of $Conv(P_j)$. These insertions will delete the conflicted cells of $DT(P_i)$ and generates the cells (tetrahedra) that merge $DT(P_i)$ to $DT(P_j)$. However, in addition some redundant cells can also be introduced.
2. The redundant vertices that are inserted in step-1 are removed. Redundant vertices are not incident to non-conflicted cells of $DT(P_i)$ and merging cells.

8.2.2 Quality

For a given 3D point set, the proposed approach computes a 3D triangulation, which is

guaranteed to be Delaunay if the sampling parameter of the algorithm is set to 1.

8.2.3 Scalability

The proposed approach is limited to the computation of the neighbourhood graph between the tiles. See iteration-2 of the algorithm for details. The computations only taking the current and the neighbouring tiles into consideration, thereby enabling triangulation of large data volumes. More details can be found in Section 7.4 and 8.1.3.

8.2.4 Degree of Human Intervention

The proposed approach is fully automatic.

8.2.5 Examples

The proposed algorithm has been tested extensively using three tiles for uniformly generated random points. The cloud nodes are simulated on a single PC in which the files are saved to the local disk that are exchanged between different processes corresponding to the different iterations of the algorithm. The Figure 12 and Figure 13 display a sample result.

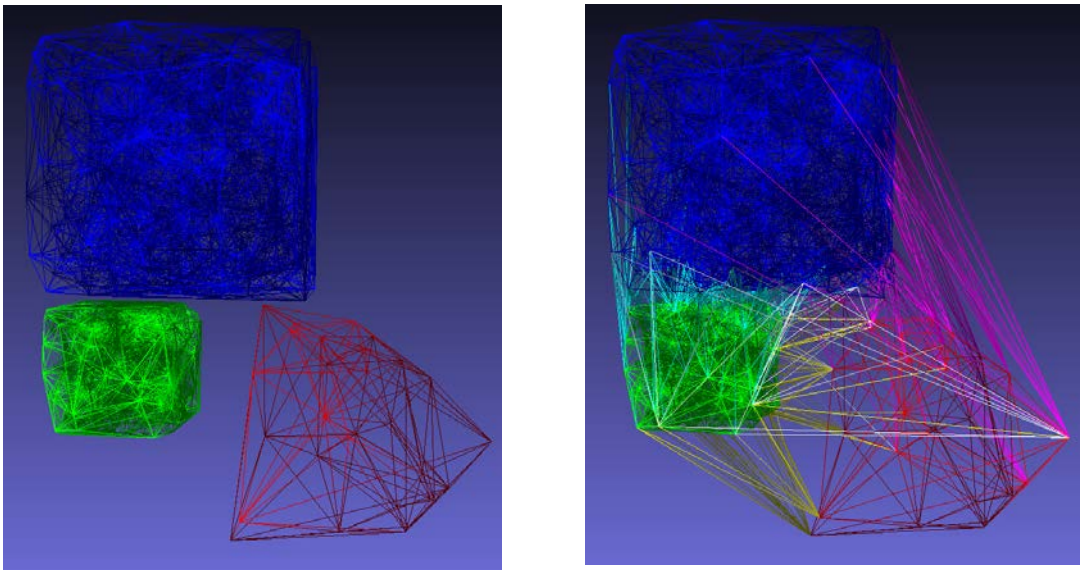


FIGURE 12: Delaunay triangulation of three tiles (left) and the corresponding merged Delaunay triangulation (right)

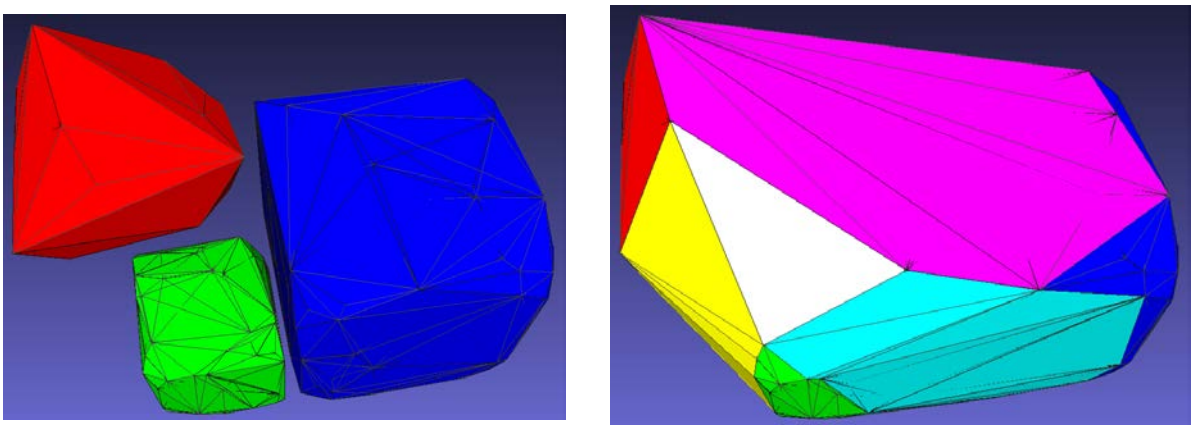


FIGURE 13: The merged Delaunay triangulation of the three tiles given in FIGURE 12. (left) with a different rendering and the corresponding merged triangulation (right). Different colours indicates different merging cells, for instance, the yellow cells merge the red and green tiles and the white cells merge all three tiles

9 NEW SERVICES

This section contains a description of the new services developed in the last period. Most of the services are used in the Marine Scenario work flows, but there are also services belonging to the Land Scenario work flows, see D1.2.3 and D1.2.4. Several of the services are new versions of services delivered in the first and second year that are adapted to multi tile computations and information transfer in a workflow.

9.1 #48

Confidential – see Appendix B

9.2 #63 EXTRACTING SURFACE NORMALS FROM POINT CLOUDS

This service is a utility for extracting surface normals from point clouds and is not tied up with any showcase. However, the current implementation assumes that the normals are pointing upwards (+*z* direction) so it may not be suitable for mobile mapping applications. Built with the parallel processing engine Apache Spark, this is a completely parallelized service.

9.2.1 Functionality & Algorithm

This service extracts surface normals by running PCA on the *k* nearest neighbours of each point. The `NormalEstimation` class of the point cloud library (PCL) performs the actual computation. The current implementation assumes normals to be pointing upwards. Any computed normal that points downwards is reversed.

The service is built using the parallel processing engine Apache Spark, which carries out parallelism over chunks of data known as partitions. Points are assigned into rectangular partitions by dividing the bounding box of the point cloud into a grid. The points in each partition are then passed on to PCL to extract the normal of each point.

9.2.2 Quality

The accuracy of the method employed, PCA, is affected by the neighbourhood size *k*. However, it is more critical to investigate the effect of partitioning on the accuracy of the results. Due to partitioning, points may be too near a partition boundary such that some of the points in their neighbourhood are excluded because they are not within the same partitioning.

Figure 14 shows the accuracy of normals extracted from a hemispherical point cloud with 16000 points. The angle θ is between the normal and the +*z*-axis and the angle φ is between the projection of the normal on the *x*-*y* plane and the +*x*-axis. The figure clearly shows that the accuracy is very high with the mean absolute error in θ being 0.0029°, and 0.49° for φ.

The accuracy of normals extracted from a triangular wave point cloud with 16000 points is shown in Figure 15. In this example, aimed to investigate the effect of a less smooth geometry, the wave has three peaks parallel to the *y*-axis. There is a slight hint of four bluish lines in the upper left panel, which correspond to relatively larger $\Delta\varphi$. The spread of $\Delta\varphi$ is noticeably larger in the upper right panel and this observation is supported by a higher mean absolute error φ_{MAE} of 2.63°. The error in θ remains very low as evidenced by a mean absolute error θ_{MAE} of 2.6°. Both observations imply that the extracted normal slightly rotates about the *z*-axis only.

To quantify the statement above about points near a partition boundary, we introduce the quantity δ_x and δ_y, which describe how close a point is from the x or y partition boundary, respectively. We define δ_x as

$$\delta_x = \frac{\min\big(\text{rank}(x), n - \text{rank}(x)\big)}{n},$$

where rank(x) is the index, starting from 0, of a point when the points in a partition are sorted by their x-coordinates, and n is the number of points in the partition. The quantity δ_y is defined similarly with y-coordinate instead of x-coordinate. The range of δ_x is [0, 0.5] with a smaller value implying it is closer to the partition boundary.

There is a peak in the mean absolute error (MAE) for smaller values of δ_x and δ_y for a hemispherical point cloud as shown in Figure 17 and this peak is expected. There is also a small unexpected peak at around $\delta_y = 0.1$. However, MAE remains small for all δ_x and δ_y with the largest MAE being 3.1°. The value of θ_{MAE} is always below 0.006°.

The expected peak at small values of δ is only seen in φ_{MAE} for a triangular wave point cloud (Figure 16) with φ_{MAE} = 8.9° for δ_y near zero. Although the maximum MAE is larger than that for a hemispherical point cloud, the peak is not at small values of δ_x, which implies that the error is less due to partitioning than due to the normals extraction method itself.

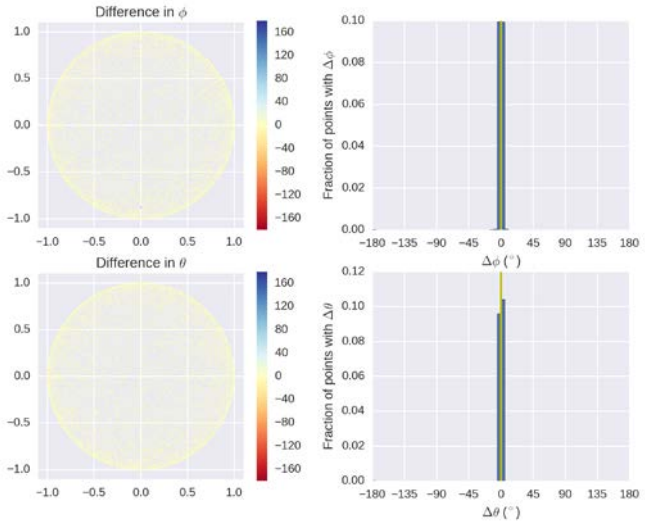


FIGURE 14: Accuracy of normals of a hemispherical point cloud

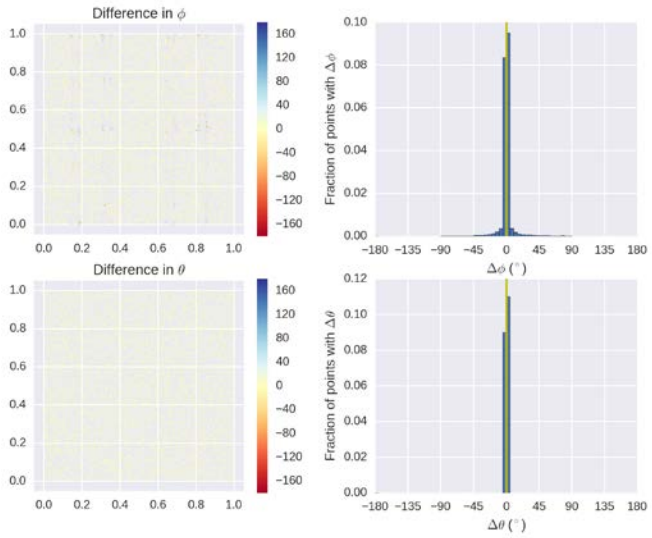


FIGURE 15: Accuracy of normals of a triangular wave point cloud

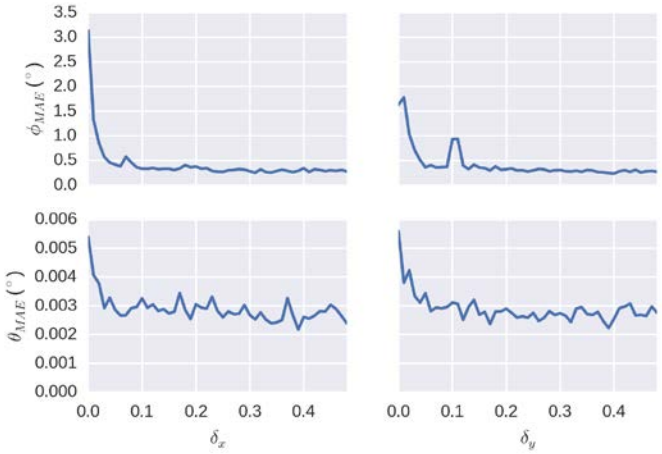


FIGURE 16: Mean absolute error of normals of a hemispherical point cloud as a function of distance from the partition boundary

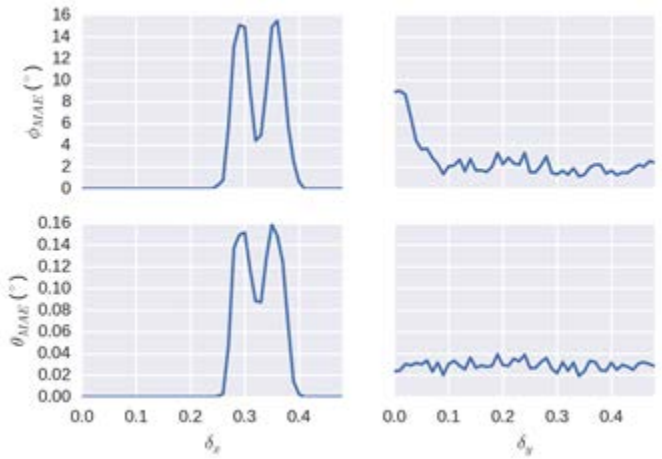


FIGURE 17: Mean absolute error of normals of a triangular wave point cloud as a function of distance from the partition boundary

9.2.3 Scalability

The scalability of the service was measured using hemispherical point clouds consisting of 10^5, 10^6 and 10^7 points stored in 10 files having 10^4, 10^5 and 10^6 points each, respectively. The number of nodes was varied from 1 to 5 nodes as shown in Figure 18. As expected, the effect of parallelism is more apparent for larger point clouds. For smaller point clouds, the overhead of parallelism and Apache Spark, in particular, is too large so it is no longer advisable to parallelize. The minimum running time occurs when there are four nodes. This may be due to having 16 partitions in the computation. Please see deliverable 6.5 for the complete scalability test results.

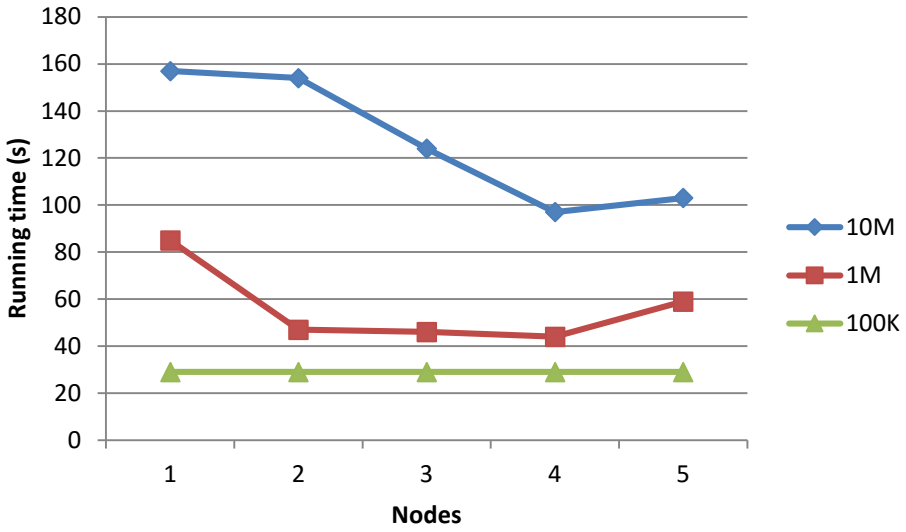


FIGURE 18: Scalability of service 63. Each line is for a different number of points in the test point cloud.

9.2.4 Degree of Human Intervention

No human intervention is required.

9.3 #107

Confidential – see Appendix B

9.4 #117 LR B-SPLINE RAINFALL FIELD TO TRIANGULATION

The service belongs to LS2 and is connected to approximation of rainfall data with LR B-spline surfaces. This service combined with service 58 constitutes a functionality that provides an alternative to service 40 or service 67 in the work flow.

9.4.1 Functionality & Algorithm

Prior to the execution of this service, service 58 has approximated the rainfall data with an LR B-spline function f(x,y) parameterized on the x- and y- coordinates of the geographical data points associated to the rainfall data. This function is evaluated in the x- and y- coordinates of a given triangulation representing the same area resulting in an enhanced triangulation. To improve the performance, the function is expanded to a tensor product spline surface during the computations giving an intermediate data size expansion.

9.4.2 Quality

Evaluation of a spline function is an exact operation.

9.4.3 Scalability

Service 58 approximates one or more time samples of rainfall data with LR B-spline functions. This service is applied to one rainfall time sample. The operation is independent on other time samples which allows for parallelization with respect to the time.

The scalability of this service is tested with data sizes 8.75 MB, 185.27 MB, 760.95 MB, 1213.07 MB and 5044.96 MB, see Figure 19. The data size is computed by the input LR B-spline field and a triangulated surface representing the area over which the rainfall field is computed. The LR B-spline field is kept constant throughout this test while the size of the triangulated surface is varied. The service is executed single threaded on a single node.

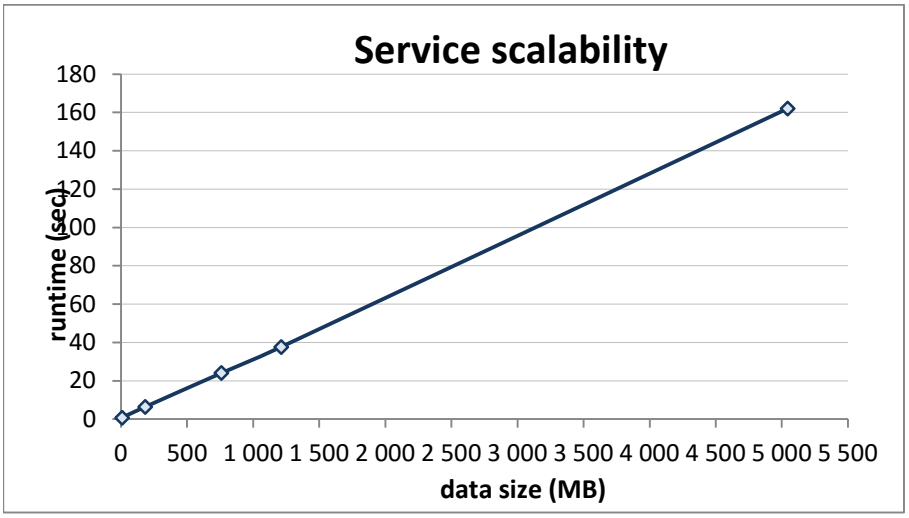


FIGURE 19: Scalability with respect to data size

As a pre-processing step, the input LR B-spline field is represented in a tensor-product structure. The effort required for this will be the same for all test cases. The main effort is spent in evaluating the field. This part of the process is linear with respect to the number of nodes in the triangulation.

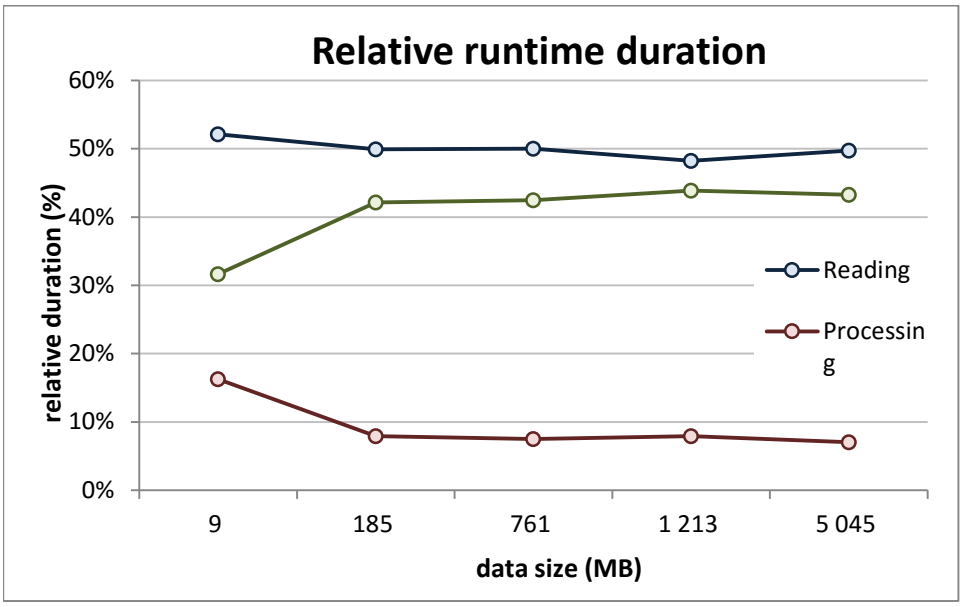


FIGURE 20: Runtime divided into time spent for reading, processing and writing with respect to data size

The majority of the time is spent in reading and writing the triangulation as shown in Figure 20. For smaller data size, the pre processing step, which represents the LR B-spline field in a tensor-product structure, is significant. The effort spend in pre-processing does not depend on the size of the triangulated surface. This implies that the processing effort is relatively larger for the smallest data size.

9.4.4 Degree of Human Intervention

No human intervention is required.

9.4.5 Examples

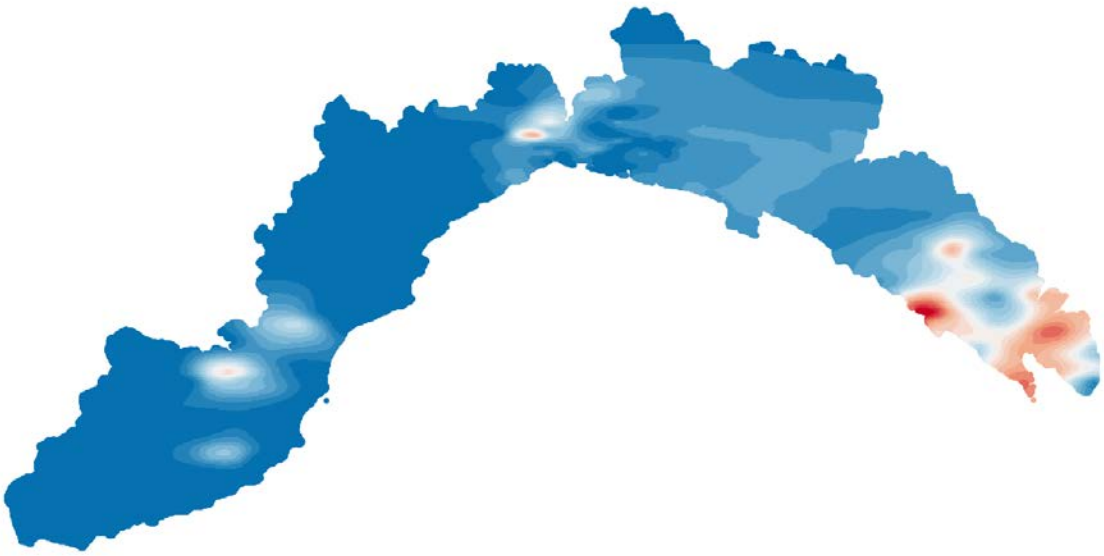


FIGURE 21: Rainfall field over Liguria computed from rainfall data obtained September 29 2013. Low rain values are coloured blue and high values are red

Figure 21 shows a rainfall field represented as an LR B-spline surface evaluated at the nodes of a triangulated surface representing the Regione Liguria in Italy. This example belongs to a study comparing three methods for approximation of rainfall data and computing persistent rainfall maxima [1].

9.5 #122 APPROXIMATE TILED POINT CLOUD WITH LR B-SPLINE SURFACE

The service is essential in the Marine Scenario work flows MS1 and MS2. It corresponds to Service 9 (Approximate point cloud with LR B-spline surface, see D4.4.1 and D4.4.2) and is adapted for use on tiled data and utilizing parallelization by the IQmulus infrastructure. The service is multi threaded.

9.5.1 Functionality & Algorithm

The approximation algorithm itself is fully explained in reports D4.4.1 and D4.4.2 and details can be found there. The service creates a surface approximating the data belonging to one tile extended with nearby data points from adjacent tiles.

The input data to the work flow consists of one or more data surveys that can be overlapping. The data set presented in Section 6.1 is a typical example. Preparations for the current service are performed by the services 136, 128 and 129 resulting in a regular tiling for each data survey. Service 122 is executed independently for each tile. The actual data set is collected from the corresponding tile for each given survey and extended with a fraction of the adjacent tiles. The generated surface is initially larger than the corresponding tile and is cut back to the tile boundaries. The motivation for creating these extended surfaces is to get good consistence between surfaces corresponding to adjacent tiles. The surfaces will after creation by this service meet with almost C^1 continuity and they can be modified to become exactly C^1 continuous by

applying Service 85. In this way, we achieve a seamless transition between individual tile surfaces.

The approximation itself is an iterative process starting from an initial spline surface with a lean and regular polynomial structure. At each step in the iteration, the point cloud is approximated with maximum accuracy given the current data size. The resulting approximation errors are computed and the surface is refined in areas where a prescribed accuracy is not met. This process is continued until this accuracy is obtained or a given number of iteration steps is performed. At each step the approximation is performed using least squares approximation or multilevel B-spline approximation applied to an LR B-spline surface. The two approximation methods are described in reports D4.4.1 and D4.4.2, respectively.

9.5.2 Quality

As for Service 9, quality refers to approximation accuracy and surface quality as measured by surface curvature. The reflections on service quality in Section 6.1.4 in D4.4.2 are applicable also for this service.

9.5.3 Scalability

The service has been tested with respect to scalability on file collections of sizes 0.1 GB, 0.7 GB, 2.3 GB, 4.8 GB and 9.5 GB. The tests have been run on configurations with 1, 6 and 12 nodes with the exception of the smallest test case which is run on 1 and 2 nodes. The runtime in minutes are shown in Figure 22. The runtime decreases rapidly from 1 to 6 nodes, but then the reduction is slowed down. The relative difference between 1 and 6 nodes is much larger than between 6 and 12 nodes.

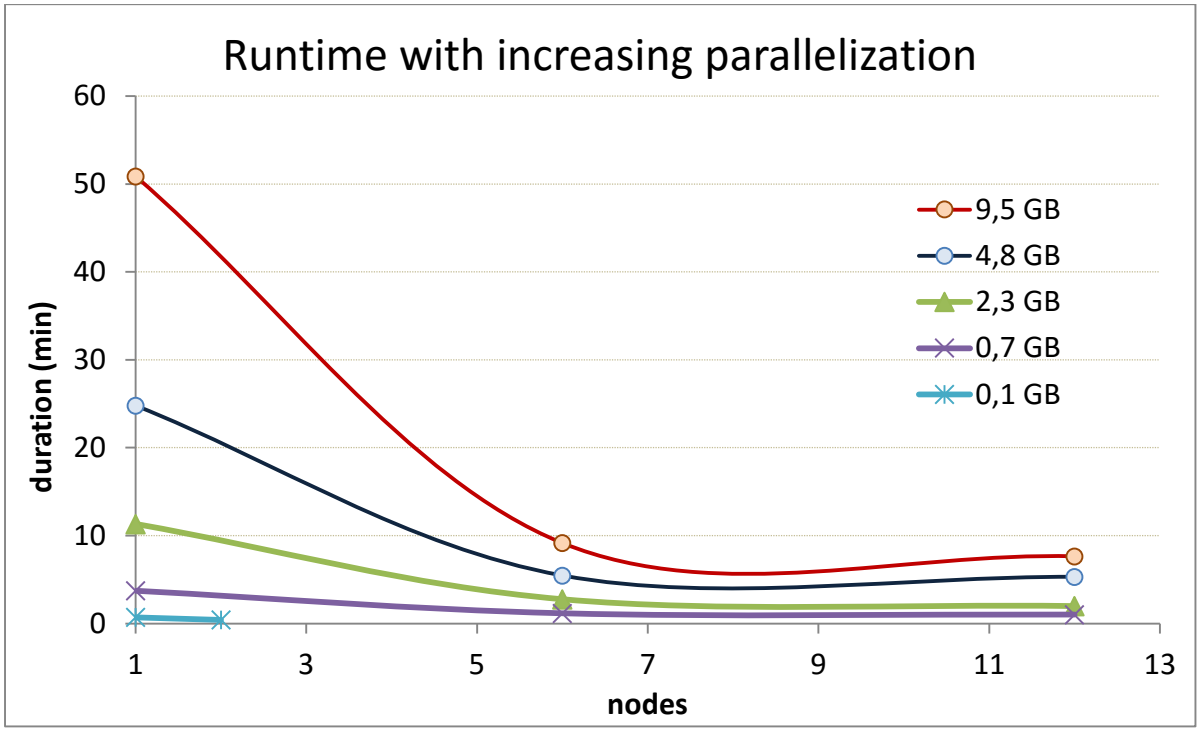


FIGURE 22: Parallelization on up to 12 nodes in the Fraunhofer Cloud

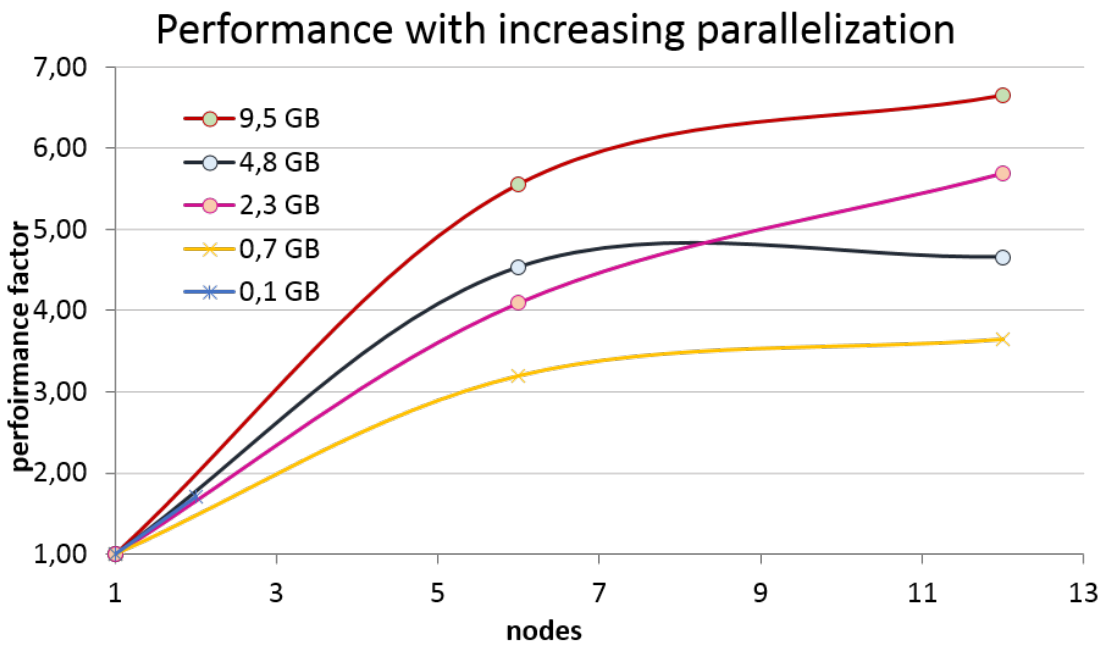


FIGURE 23: Performance with respect to the number of nodes

In Figure 23 we again find the parallelization effect from Figure 22 but we see that the data sets behave differently. The service is parallelized with regard to the surfaces that are produced and not the input data directly.

Figure 24: Data processing speed measured in MB per seconds

This implies that the number of points being handled by each service execution can vary with a large extent. Thus, arbitrary variations regarding how the executions are distributed among the nodes can have an effect on the scalability result. The general trend is still that the performance is improved with the number of nodes.

The most dominant factor for the processing speed shown in Figure 24 is the input data size and the general behaviour is that each data point is processed a number of times giving a constant data processing speed. However, the shape of the terrain influences both how often each data point is processed and the data size of the produced surface, which again influences the performance. Thus, the relation between data size and execution time is more complex than just linear. We also see the parallelization effect for the second largest data set.

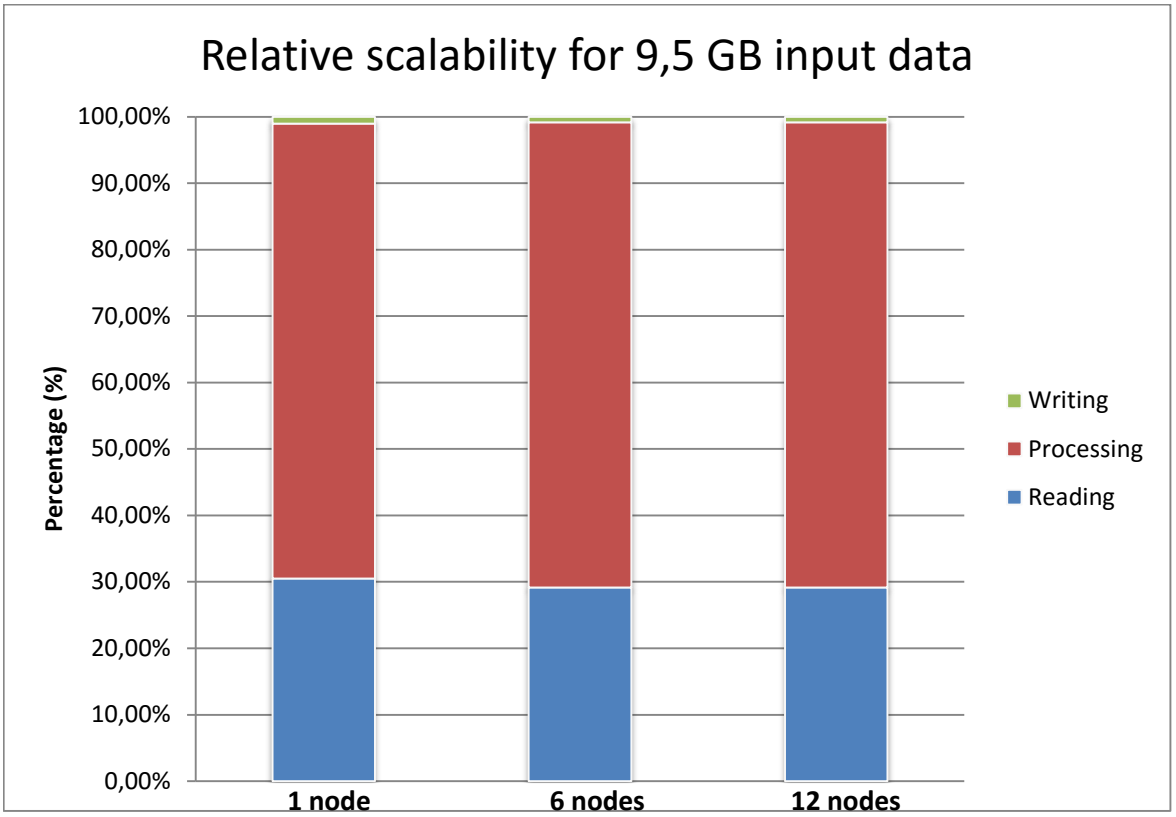


FIGURE 25: Execution effort split into reading the data sets, processing and writing of the surfaces

The proportion of the effort between reading, processing and writing is relatively constant regardless of the number of nodes as can be seen from Figure 25. The algorithm is quite complex so the processing time is the most dominant factor as expected. The point cloud is tiled and the service approximates one tile with a surface, but extends the tile with additional points from adjacent tiles to get an overlap. This implies that the much more points than indicated by the data size are actually read. The surfaces created by the service are small compared to the input data size so writing time is almost neglectable.

9.5.4 Degree of Human Intervention

Similar to Service 9, the service has two parameters, which influence the result, the tolerance and the maximum number of iterations, and several optional ones. The mandatory parameters do not have any default values. The number of iterations is advised to lie in the range [3,7] depending on the wanted accuracy. The tolerance should reflect the smoothness of the point cloud and the purpose of applying this service. However, even for an accurate reproduction, the tolerances should not be set lower than ~0.25 m. A study on the impact of these two parameters can be found in D4.4.2.

9.5.5 Examples

Figure 26 shows the result of applying Service 122 on 16 regular tiles defined from the data set in Figure 1. The polynomial patches of the surfaces are emphasized. After surface approximation, the set of surfaces has been subject to trimming by Service 125, see Section 9.8 and stitching by Service 85, see Section 5.6 of D4.4.2. The increased density of knot lines giving

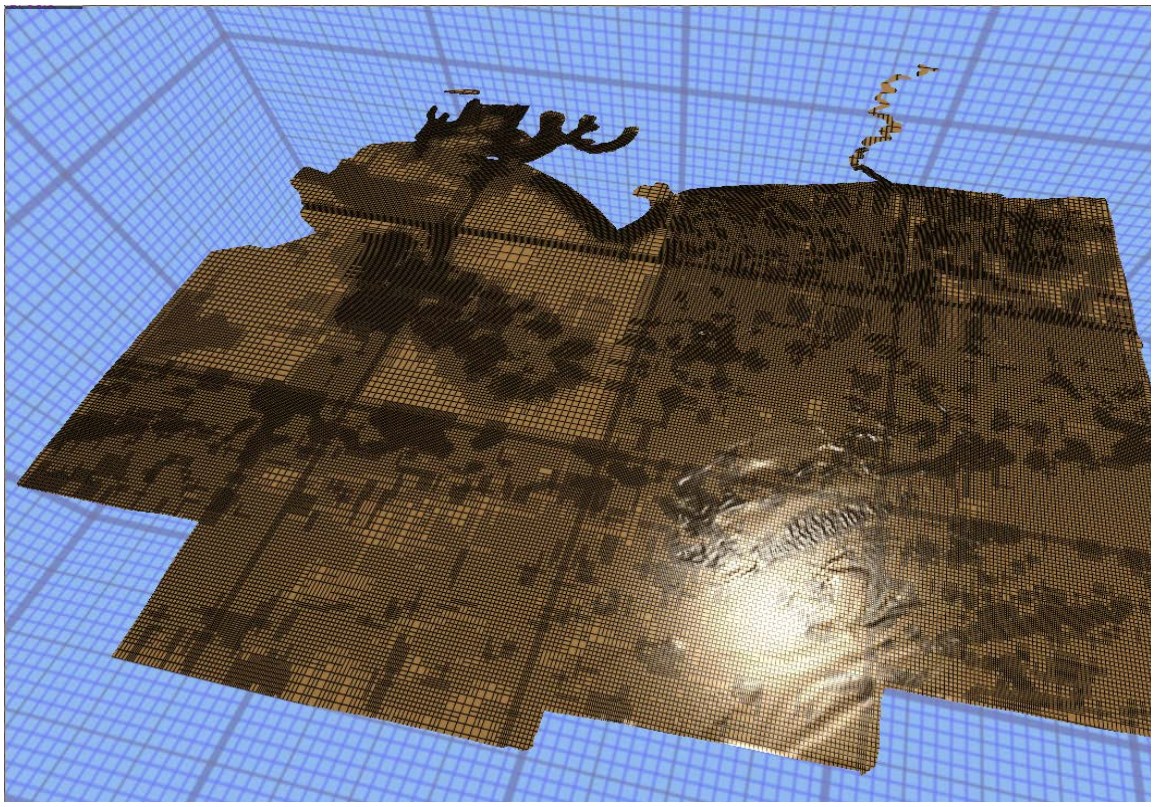


FIGURE 26: A set of LR B-spline surfaces approximating the data collection shown in Figure 1. The polynomial patches from which the surfaces are defined, are shown as black rectangles.

many small polynomial patches along the surface boundaries are caused by Service 85.

The initial data surveys may potentially contain inconsistencies as they are not yet deconflicted, see Section 9.6, so only four iterations are applied in the adaptive approximation algorithm. This implies that the result will not be very accurate, but could fit well as a reference surface that can serve as a tool in the deconfliction service (123).

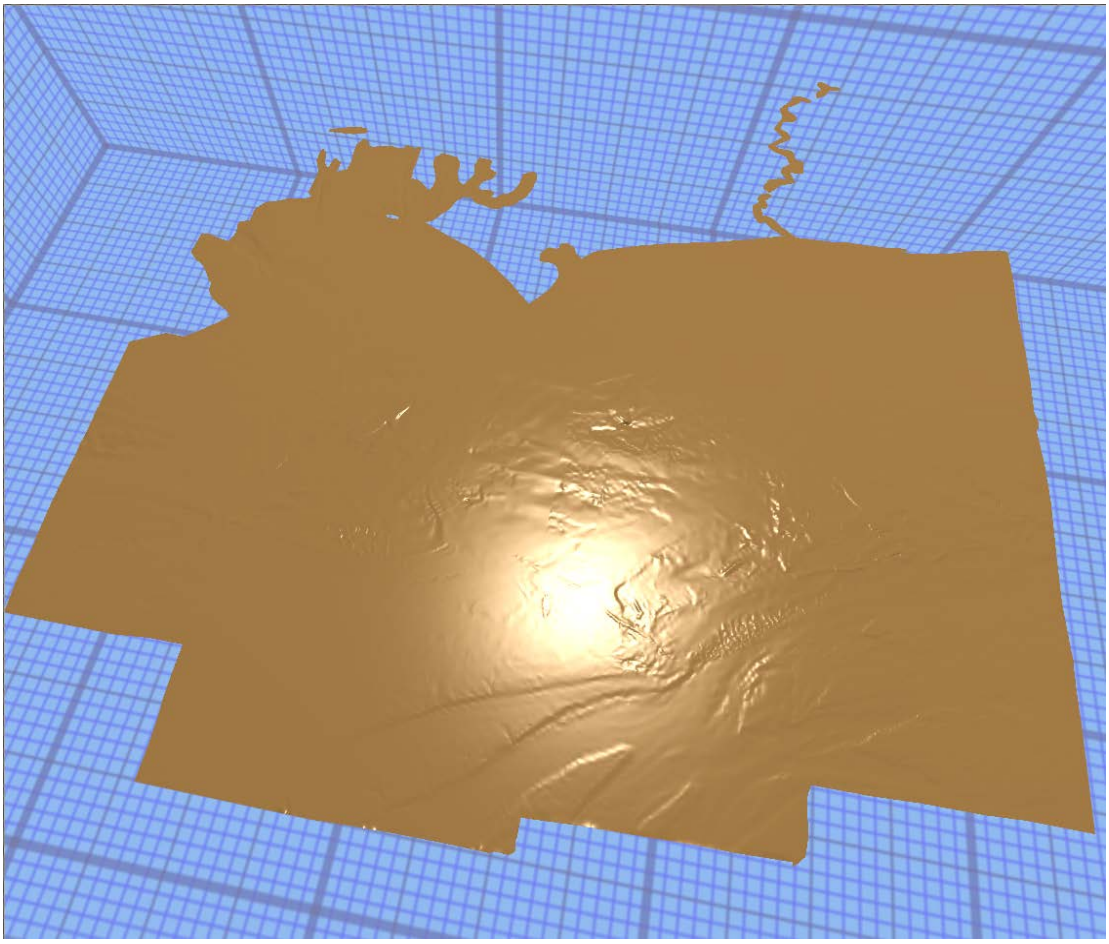


FIGURE 27: The same surfaces as in FIGURE 26 shown as one unit.

9.6 #123 DECONFLICTION

This service represents the core of MS1, the Marine Scenario deconfliction work flow. It is applied to tiled data and is designed for multi node execution by the IQmulus infrastructure.

9.6.1 Functionality & Algorithm

The service is applied for one tile and receives tiled data from a number of overlapping data surveys and a reference surface created by Service 122 as input. The data surveys are equipped with priority scores.

A data survey is equipped with a set of attributes, for instance

- METADATA_ID / BLOCK_ID
- SURSTA – survey start date
- SUREND – survey end date
- TECSOU – technique of sounding

- POINTS – number of points
- MAX_EDGE_LENGTH_95PERCENTILE – Point density in a survey block given as distance between neighbouring soundings

The various attributes are given as input to an algorithm that determines on a single score for each survey, a score that can be extended to be associated with individual points. The priority scores are computed prior to the deconfliction work flow, and are utilized in the current service to rank the data surveys with respect to importance.

The deconfliction algorithm is described as follows:

- For each polynomial element of the reference surface
    - For each overlapping data survey
        - Fetch the associated data points
        - Compute signed distances between the data points and the reference surface
- Group elements with respect to the data surveys present for each element
- Split large element groups into smaller entities
- For each element group
    - Compute statistics on the distance fields corresponding to the data surveys
    - If the element group statistic clearly identifies which surveys are consistent with the survey having the highest priority score
        - Divide surveys into those that will be kept and used for further surface creation and those that will be removed
    - Otherwise
        - For each element
            - Compute statistics on the distance fields corresponding to the data surveys restricted to the element
            - Classify local survey points as *consistent*, *not consistent*, or *with unclear relation* to the survey with the highest priority score
        - Modify the classification of points that have an *unclear relation* according to the main tendency of the element group
        - Dependent on the classification, divide surveys points into those that will be kept and used for further surface creation and those that will be removed
- Check if small groups of survey points having a deviant classification compared to their neighbours should be reclassified

All points in the survey with the largest priority score will be kept and used for surface generation. In areas where only one data survey exists, the points of this survey will be kept even if they are associated with a low priority score. Data surveys or groups of points belonging to a data survey, which are consistent with the survey with the highest score, will be kept for further use, other points will be removed. This implies that some points in a data survey may be kept while other points are removed. The percentage of kept points tends to be higher if the survey has a high priority score, but this depends on the relation between the current survey and the one with the highest score.

The key point in this algorithm is how to decide whether a set of points from one data survey is consistent with nearby points from the survey with highest score. Note that the points typically lie in different positions and the pattern of points in the various surveys differs.

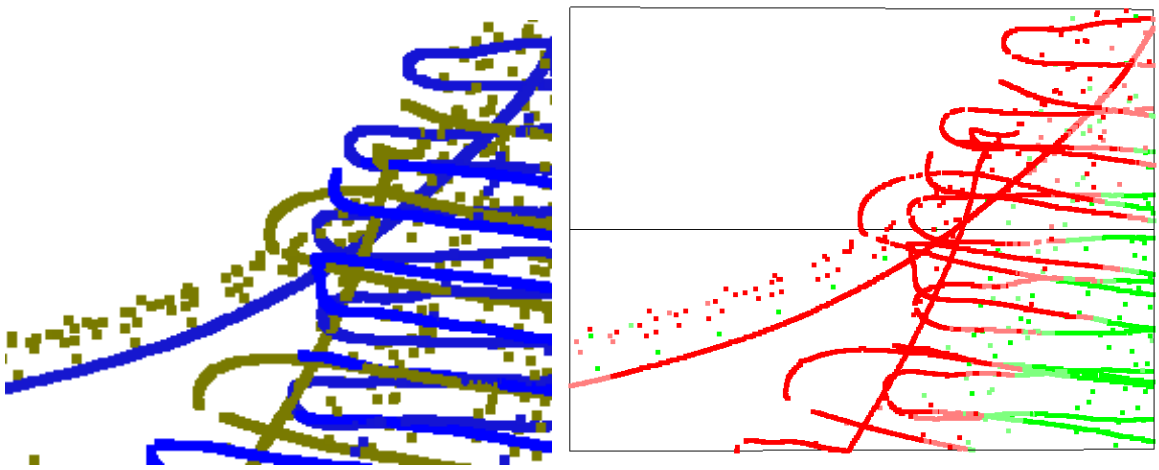


FIGURE 28: A detail with 9 surveys.

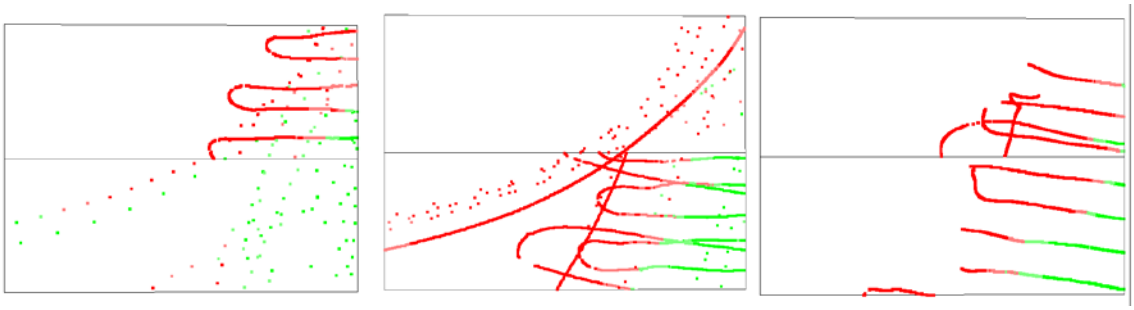


FIGURE 29: Result after initial deconfliction. The first picture shows points that are classified as in conflict with the survey with highest score, the second picture shows the points where no decision is made while the last picture shows points that are found to be consistent.

The survey configuration shown in Figure 28 is a part of the setting showed in 6.2. Surveys with a low priority score are green while high scores are shown as blue points. A higher score gives a clearer blue colour. This configuration is present in two polynomial elements of the reference surface only. The points in the second picture are coloured according to their distance to the surface. Red points lie above the surface and green points below. Clear read and green indicates a larger distance than a fainter colour.

Surveys that lie on opposite sides of the reference surface and with a significant distance to the surface give inconsistent information. This is, however, not the typical situation. Points from one survey may lie on both sides of the surface and the same is the case for another survey. Still, the distance ranges with respect to the surface may differ and so may the average distances. The relative positioning of the points within the polynomial element is also significant. The closer the

points, the more significant is a difference in distance to the surface. A first classification result, computed individually for each element is shown in Figure 29. The distance information is for many groups of survey points found not to be sufficiently significant resulting in many unclassified points. We can also see that a scan line of low priority score (green in the first picture of Figure 28) is found to be consistent compared to the survey of highest score in the upper element, but insecure in the lower.

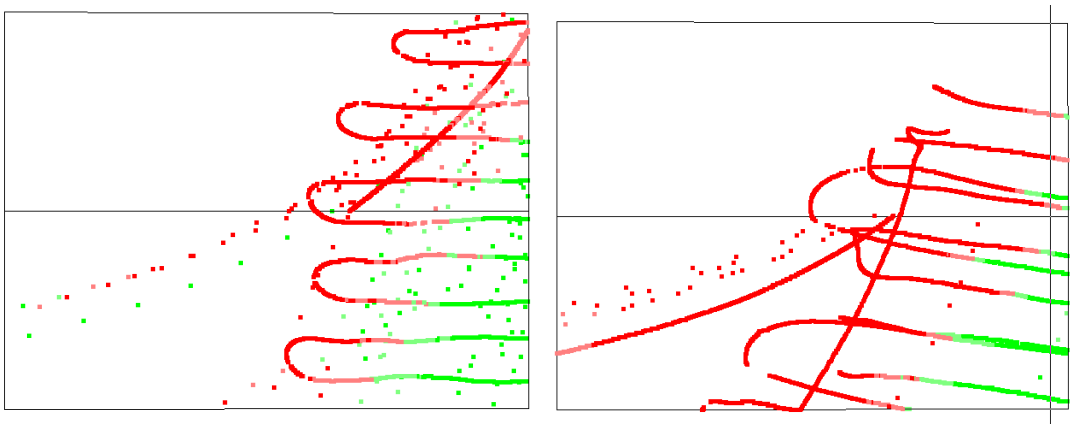


FIGURE 30: Conflicting (first picture) and consistent (second picture) surveys shown after a second decision stage

In Figure 30 information from the neighbouring element has been taken into consideration and a more tight decision procedure has been applied to split the survey points into those to dismiss (first picture) and those to keep (second picture). The same procedure is applied to all polygonal elements in the reference surface.

9.6.2 Quality

Good quality of deconfliction is characterized by resulting point clouds, which give a consistent and sufficient input for surface generation leading to good quality surfaces. The deconflicted point clouds should have the following properties:

- Nearby points from different data surveys should be consistent with respect to the reference surface. That is, they should lie either close to the surface or at the same side with approximately the same distances.
- Removal of points due to conflict should not lead to large areas without any points.
- A safety distance between groups of points from conflicting data surveys must be ensured to avoid inconsistent input to surface generation.
- Small isolated groups of points from low prioritized data surveys should be avoided.

These characteristics are qualitative rather than quantitative. Quality testing of the service is currently ongoing with the means of services 124 and 126 in combination with IQmulusViz, see D5.1.3.

9.6.3 Scalability

The service is prepared for parallelization on different nodes with respect to tiles. Until now, the service has been tested on relatively small data collections only and a preliminary scalability test gives little information. It is clear that the execution time of the service is only to a little extent related to data size. After a pre-processing step where the configuration of the overlapping data

sets is identified, the performance is more related to the complexity of this configuration than the data size itself. If no data sets overlap, the deconfliction problem is trivially solved, and the execution is stopped. Many overlapping data sets in an area having a non-trivial relation with respect to whether or not they are consistent lead to high execution times. The service is still under development and scalability results will be used to identify bottlenecks.

9.6.4 Degree of Human Intervention

The data surveys provided for this service must be equipped with priority scales to enable the prioritization of overlapping surveys in areas with conflict. A threshold value indicating when two data surveys lie close enough to be regarded as consistent, must be provided by the user. This value should correspond to the tolerance used in surface generation (Service 122 and Service 124).

9.6.5 Examples

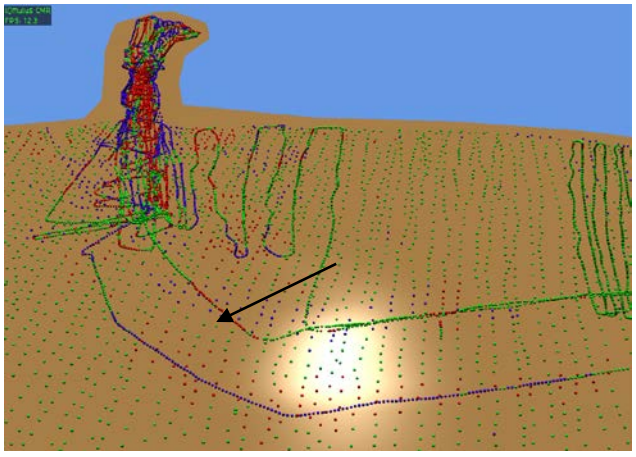


FIGURE 31: Reference surface and a number of overlapping data surveys

Figure 31 shows a detail from Figure 2 along with the corresponding reference surface used in the deconflicton algorithm. The points are visualized as follows: Green points lie within a distance to surface smaller than a threshold of 0.5 meters. Blue points lie above the surface at a distance above this threshold while red points lie below. There are areas where the different points are clearly inconsistent with regard to the reference surface as indicated by the arrow.

Figure 32 shows the result after deconfliction. In the specified area conflicting data points belonging to low priority surveys are removed and the same exercise has been performed all

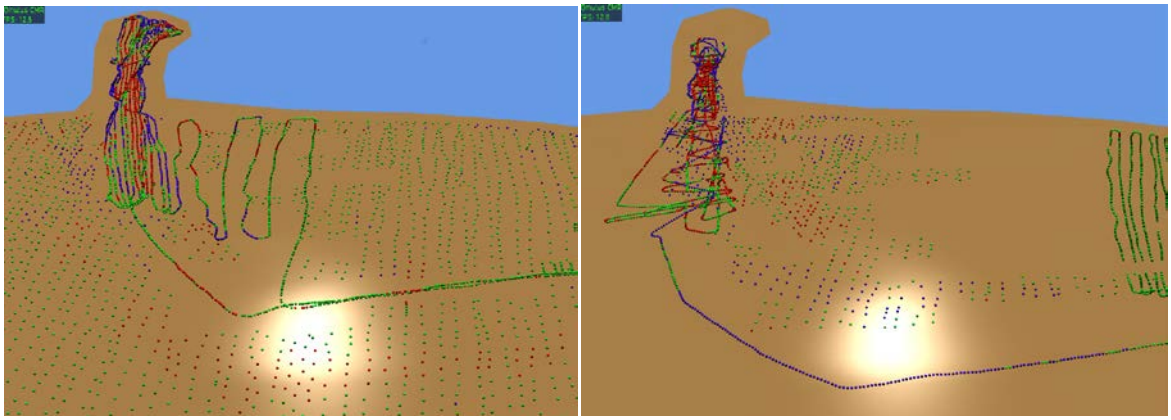


FIGURE 32: Rresult after deconfliction, the left picture shows the points that are classified as consistent with the high score surveys while the right picture shows the dismissed points.

over. The distance between the points and the surface is generally larger than the threshold also for the deconflicted point clouds as the reference surface is created with low accuracy.

9.7 #124 UPDATE SPLINE SURFACE WITH POINT TILE

The service belongs to MS1 and corresponds to Service 57. Service 57 is applied to one point cloud and an initial surface corresponding to this point cloud while Service 124 is applied to one tile from a data set and the corresponding surface. This implies that the current service can be executed in parallel on several nodes. Both services are multi threaded.

9.7.1 Functionality & Algorithm

An initial LR B-spline surface and a corresponding point cloud tile are given and the service updates the surface with the point cloud to improve the approximation accuracy. In essence, the algorithm is the same as for Service 9 and Service 122, but the execution starts after the adaptive approximation algorithm used in these services has been run for a few steps. In this service, a current surface is given as input together with a point cloud and some additional steps in the approximation algorithm are run.

9.7.2 Quality

As for the other services belonging to the same category (9, 57 and 122) quality refers to approximation accuracy and surface quality as measured by surface curvature. Since applying this service is similar to restarting Service 122 at a point in time, also the quality performance will be similar.

9.7.3 Scalability

Scalability testing of this service is not yet performed, but it is expected that the result will resemble that of Service 122.

9.7.4 Degree of Human Intervention

The service has two parameters that influence the result, the tolerance and the maximum number of iterations. The number of iterations is advised to lie in the range [1,3] depending on the wanted accuracy and the size of the initial surface. The tolerance should reflect the smoothness of the point cloud. However, even for an accurate reproduction, the tolerances should not be set lower than ~0.25 m.

9.7.5 Examples

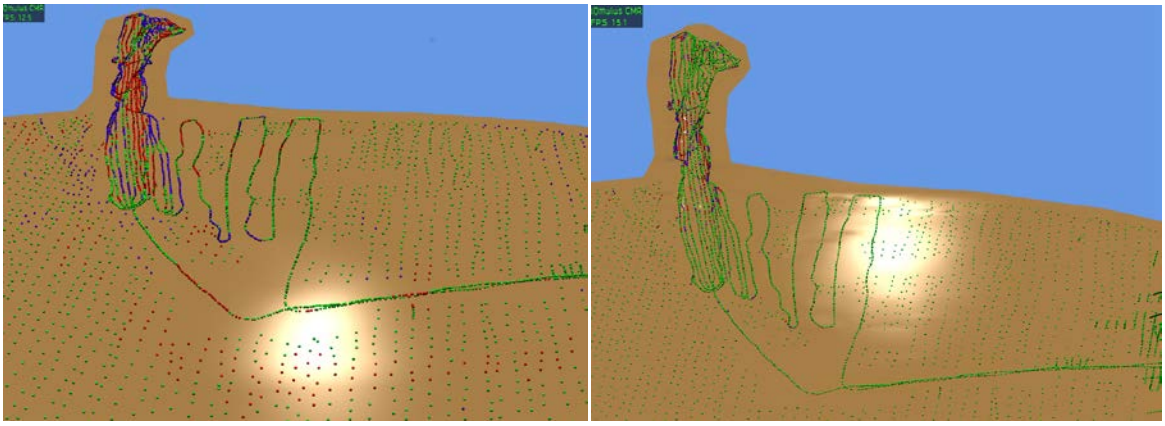


FIGURE 33: Accuracy of point before and after updating the surface with service 124.

The reference surface from the deconfliction example (Section 9.6.5) is updated with three iteration steps and the result can be seen in Figure 33. To the left the accuracy of the deconflicted points before the update of the surface is visualized. The colour coding is the same as in Figure 31 and Figure 32. To the right, we see the result after updating the surface. Now most of the points are within the threshold.

9.8 #125 TRIM LR B-SPLINE SURFACE FOR ONE TILE

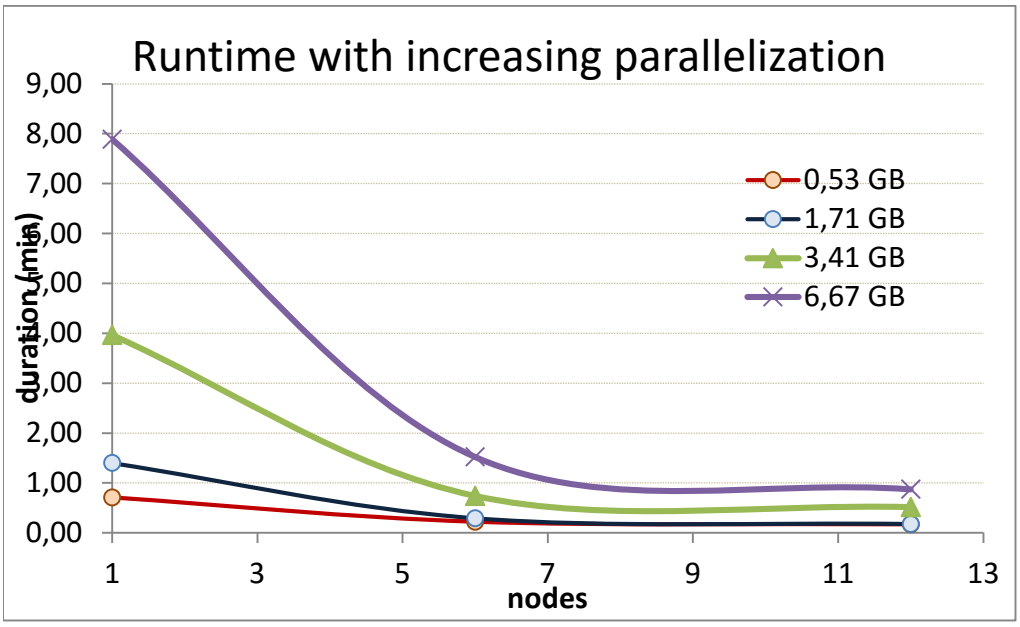


FIGURE 34: The duration of the execution with respect to the number of nodes for all data sizes

The service is used in MS1 and MS2 and corresponds to Service 90. It is designed for use in a tiled environment. The service can, for each tile, be executed in parallel.

9.8.1 Functionality & Algorithm

The service is applied to the surface corresponding to one tile in a regular tiling limiting the domain of the surface to the domain covered by the point tile. The regularity of the tiling implies that many tiles will have a rectangular domain, but tiles at the boundary of the total domain are expected to have a non-rectangular domain and this will also be the case if there are large areas without points in the interior of the point cloud. In these cases, trimming will be applied.

The trimming algorithm is the same as for Service 90, see D4.4.2. In a recursive approach, sub domains of the parameter domain of the surface are inspected to check if the lie on the boundary of the point cloud domain. After a given number of recursion levels, the boundary of those sub domains positioned at the boundary of the point cloud are used to define a polygon surrounding the point cloud domain. This polygon is again approximated by a spline curve to create a trimming curve in the parameter domain of the surface.

9.8.2 Quality

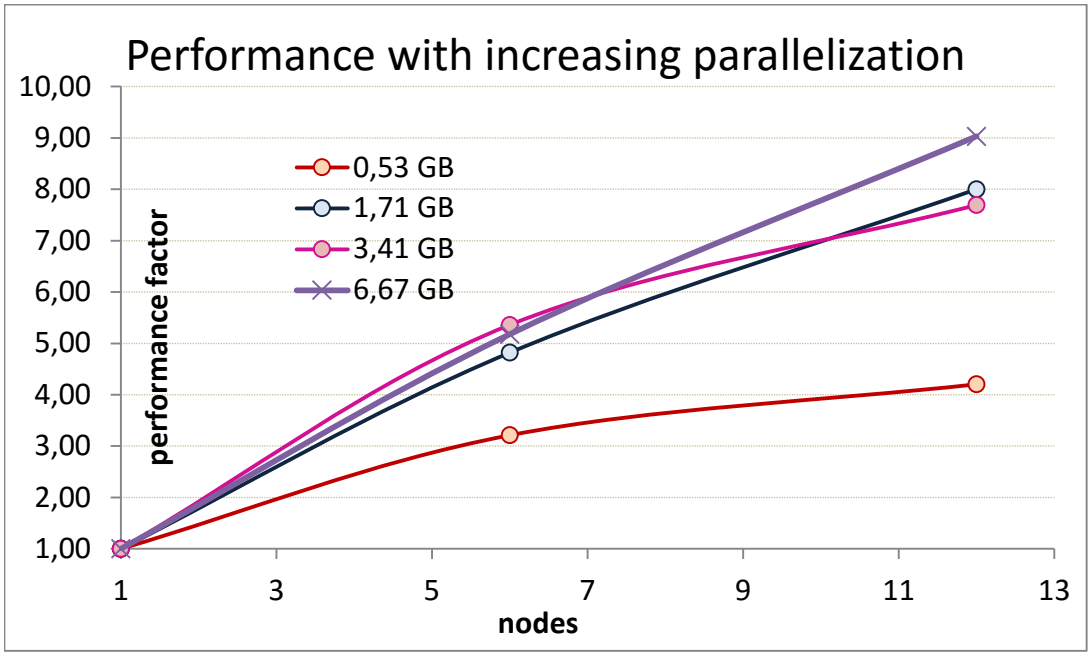


FIGURE 35: The development of the performance when the number of nodes is increased, node count 1, 6 and 12

In the context of trimming, quality is related to how well the trimming curve follows the outline of the point cloud. In D4.4.2, Section 5.4.3 a test was performed to check how well the current algorithm followed this outline. The tightness of the trimming curve is dependent on an input parameter, but if a very tight trimming was specified, a few points were excluded from the trimming domain of the surface. This may still occur with the current version of the service.

A data set obtained by combining several overlapping data surveys with or without a prior deconfliction step, can have a very non-uniform pattern. This is particularly prominent if some of the data surveys can be categorized as scan lines. Figure 2 illustrates this problem. In some parts of the data set, the trimming curve cannot follow the point cloud too closely as this will result in a curve following individual scan lines. This implies that in other areas, the trimming becomes very rough as can be seen in for instance Figure 31. The current algorithm uses the same tightness to the point cloud in the entire domain.

Point clouds containing different point clusters either originating from the initial data surveys or being induced by the tiling, should lead to several trimmed surfaces, each representing an "island" with data points. Currently, only the largest "island" is represented. Thus, there are some quality issues with the current trimming approach when applied to a typical data set being input to MS1 and MS2. These work flows are important for exploitation so an effort to improve the quality of this service will be maintained even after D4.4.3 and the finalization of Task 4.4.

9.8.3 Scalability

The service has been tested on 1, 6 and 12 nodes with collections of data surveys adding up to 538 MB, 1753 MB, 3491 MB and 6826 MB. Included in these data sizes is the input surface, which constitutes a small fraction of the total data size.

The execution time decreases rapidly from 1 to 6 nodes, see Figure 34, but only slightly when the number of nodes is increased to 12. The relative difference between 1 and 6 nodes is much higher than between 6 and 12 nodes. For the smaller data sets, it is also likely that the potential of the number of nodes is not fully exploited.

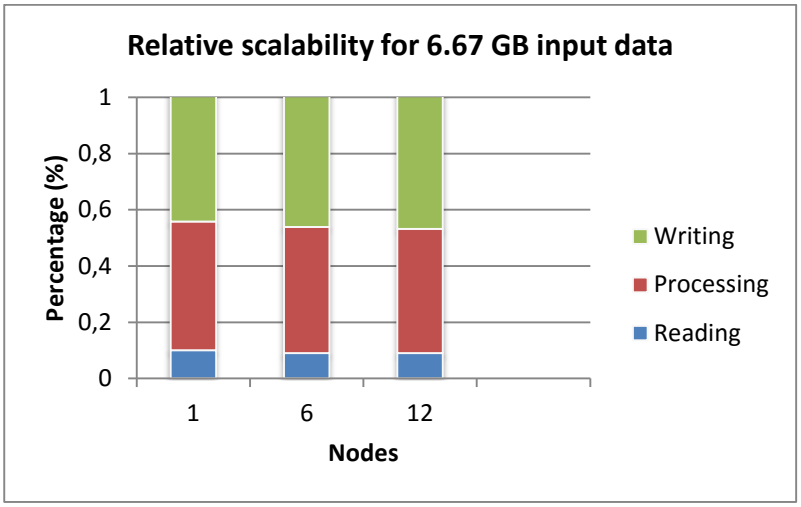


FIGURE 36: Relative effort distribution between reading, processing and writing for the 6.67 GB data set

The service is parallelized with respect to the number of tiles which is linked to the data size, but which also depends on the configuration of data points. The number of points in each tile also depends on how the points are distributed. The data sets included in this test have different configurations, which implies that a non-uniform shape of the graphs corresponding to the data sets is expected. As shown in Figure 35, the overall trend is that the performance increases with the number of nodes, but that this increase is slowed down when the node count is increased.

The service is performed on tiled data and the execution is independent for each tile. Thus, the relative scalability is not affected by the number of nodes as we can see in Figure 36.

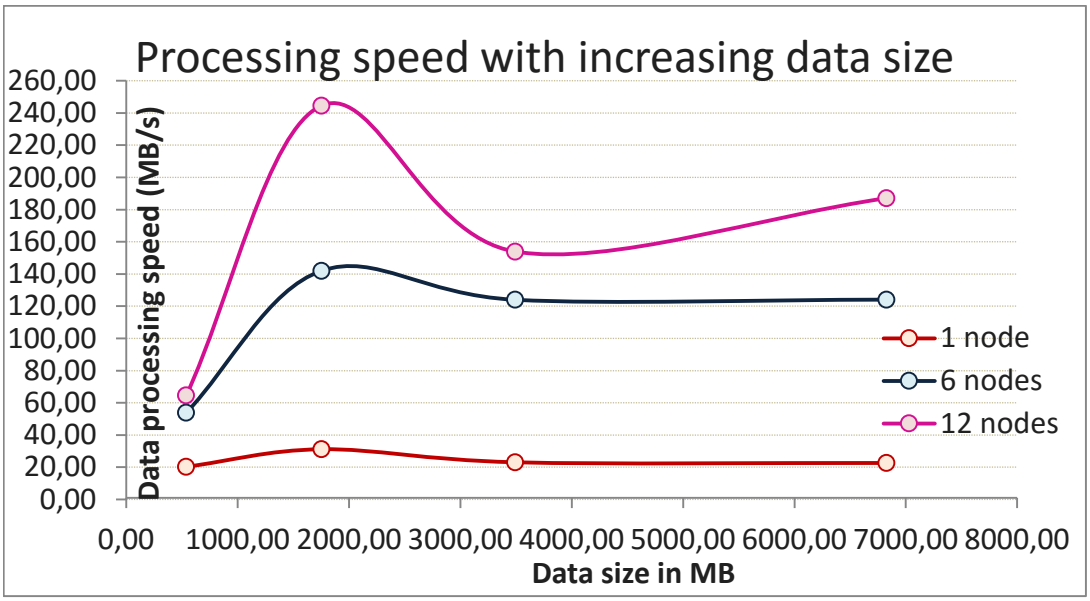


FIGURE 37: Number of megabytes processed per second with respect to data size and number of nodes

The performance, see Figure 37, depends partly on the number of data points, but also on the configuration of data points. When the domain corresponding to a point tile is rectangular and corresponding to the surface domain, little effort is spent on trimming, but if the point cloud domain is complex, more work is required to bound this domain. Thus, a graph with large variation that converges to constant value for large data sizes is expected.

9.8.4 Degree of Human Intervention

The service has one sensibility parameter influencing the density with which the trimming curve bounds the point cloud. This parameter is translated into a maximum recursion level for the computations and information on splitting the point cloud into sub clouds. An example showing the significance of this parameter is given in Section 5.4.5 in D4.4.2.

9.8.5 Examples

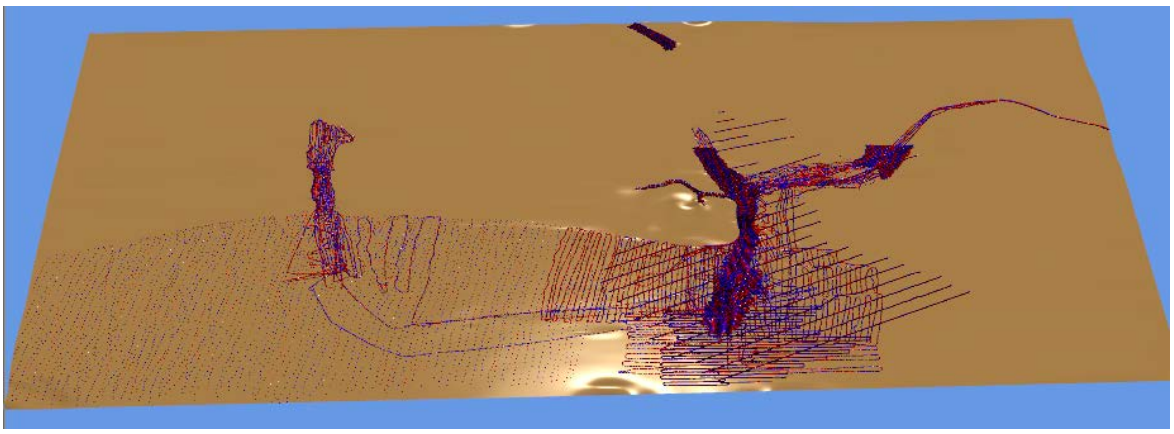


FIGURE 38: The point clouds from Figure 2 and an approximating LR B-spline surface

Figure 38 shows the point cloud collected from all data surveys together with the approximating surface. Due to the rectangular domain of the surface, it exceeds the point cloud considerably.

Figure 39 shows the same situation after the surface has been trimmed to limit the surface area corresponding to the point cloud. The challenges with the current algorithm are also obvious in this example.

9.9 #126 DISTANCE FIELD SURFACE VERSUS POINT CLOUD TILE

The service is essential in MS2 to create input for quality control. It is also useful in MS1 to ensure that the deconfliction quality is sufficient. The service corresponds to Service 88 for non-tiled data sets.

9.9.1 Functionality & Algorithm

A set of metadata files keep track on the connection between tiled point clouds and corresponding surfaces. Given a point cloud and surface pair, the distance field is obtained by evaluation. During the service execution, intermediate links into the data structure of the surface

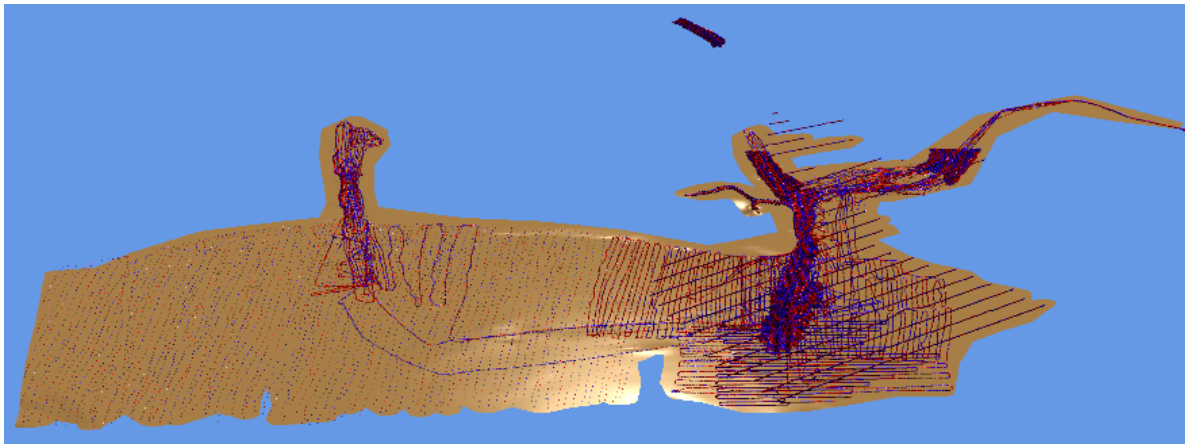


FIGURE 39: Point cloud and trimmed approximating surface

is created to improve performance, see also Section 5.2.1 in D4.4.2.

9.9.2 Quality

Spline surface evaluation is an exact operation. Thus, no quality issues are expected.

9.9.3 Scalability

The service is multi threaded and is prepared to be distributed on different nodes with respect to pairs of point sets and surfaces.

The service has been tested on 1, 6 and 12 nodes with collections of data surveys adding up to 107 MB, 565 MB, 1759 MB, 3491 MB and 7017 MB. Included in these data sizes is the surface involved in the distance computations. The point sets substitutes the major part of the data size.

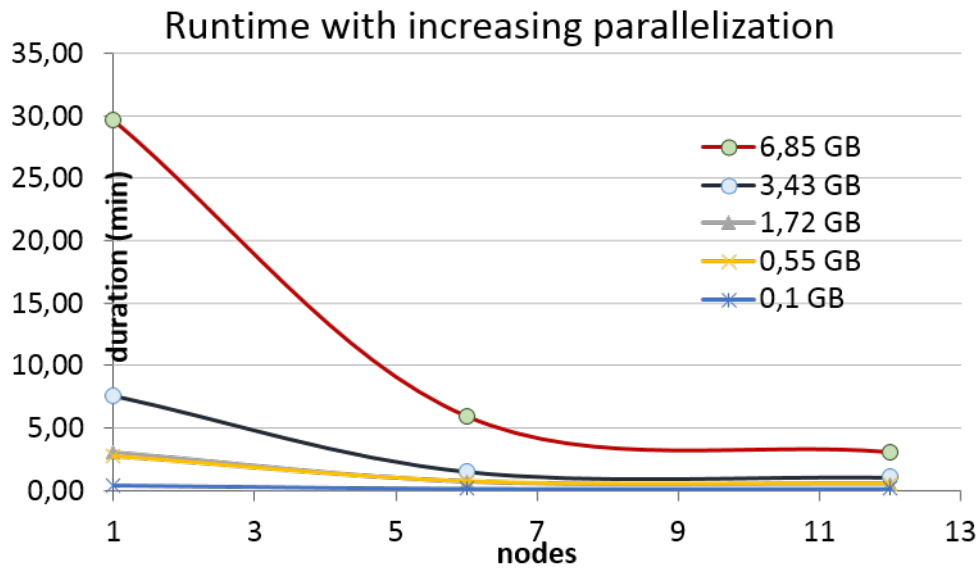


FIGURE 40: The duration of the execution with an increasing number of nodes

The execution time, see Figure 40, decreases rapidly from 1 to 6 nodes, but only slightly when the number of nodes is increased to 12. The relative difference between 1 and 6 nodes is much

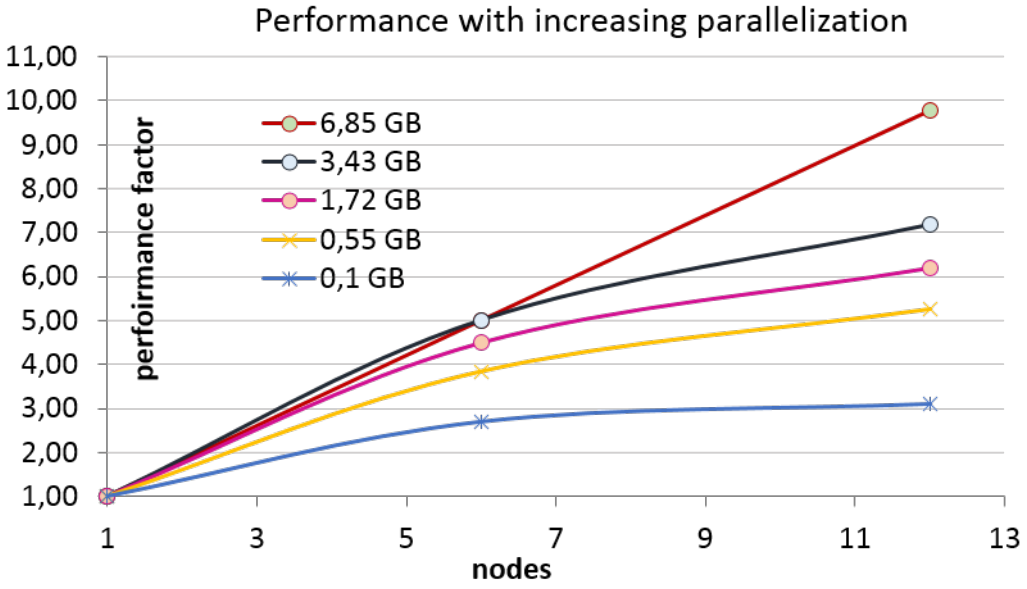


FIGURE 41: The development of the performance for nodes 1, 6 and 12

higher than between 6 and 12 nodes.

As expected, the performance shown in Figure 41 increases with the number of nodes. For the smaller data set, the effect of adding more nodes is decreasing.

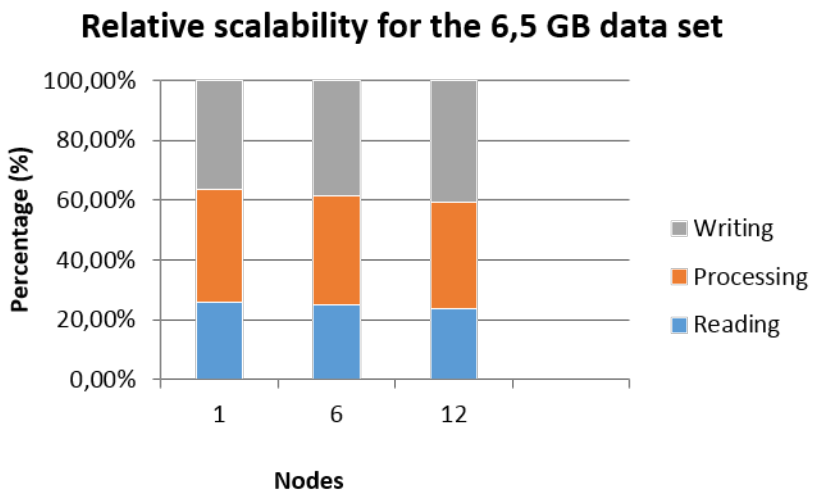


FIGURE 42: Distribution of execution time for reading, processing and writing for the 6,5 GB data set

The relative scalability (Figure 42) is quite constant regardless of the number of nodes. The

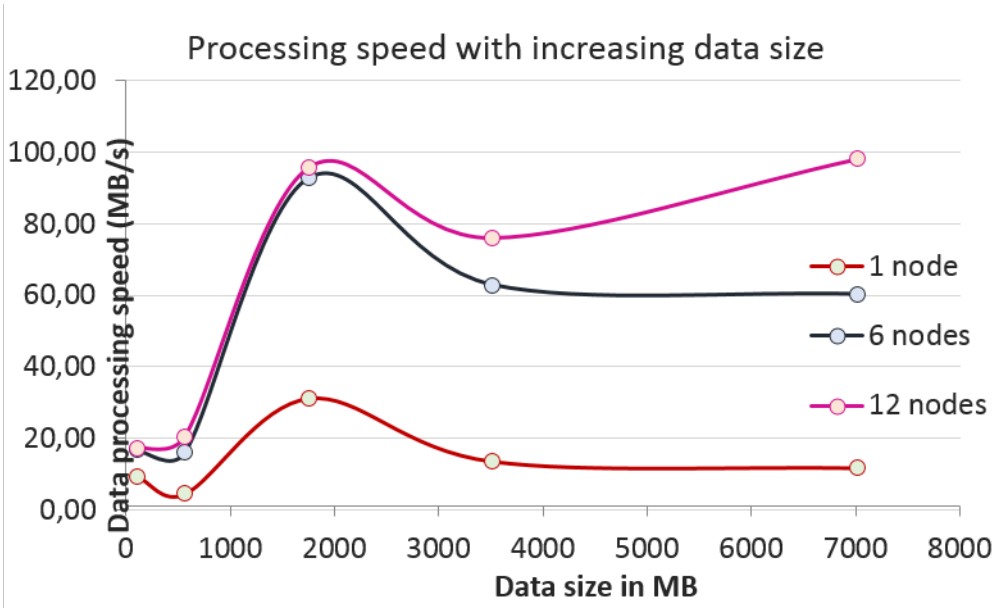


FIGURE 43: The number of MB processed in a second for varying data sizes and node

processing for each data point is quite simple in this service, just evaluation, so relatively much time is spent on reading and writing.

The measured processing speed of Service 126 is shown in Figure 43. The service consists of two steps a pre-processing step setting up a local data structure to simplify the navigation in the data structure of the LR B-spline surface and the distance field evaluation. For the latter, the data processing speed is expected to be constant, while the first step depends on the number and size of the surfaces and not the number of data points.

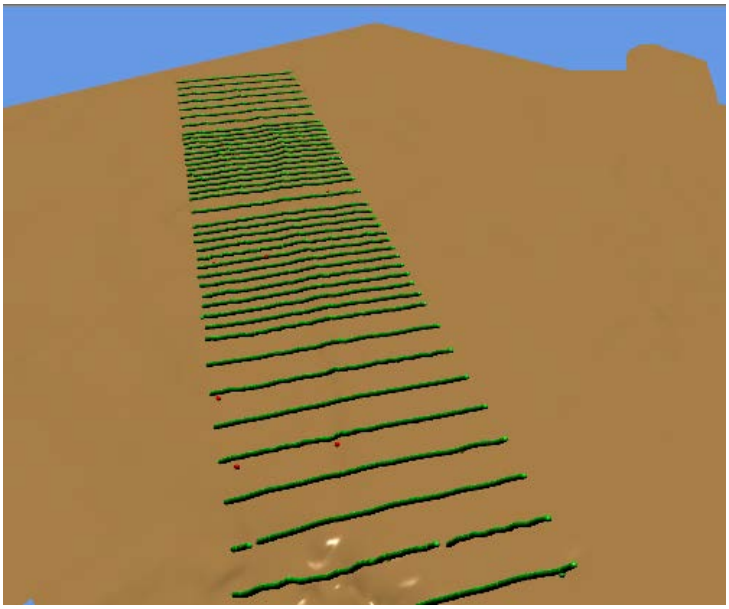


FIGURE 44: Point cloud coloured according to the distance between the points and an approximating surface.

9.9.4 Degree of Human Intervention

The service does not require any user input.

9.9.5 Examples

One use of this service is to detect outliers. Consider Figure 44. One data survey from the collection of test data set for deconfliction presented in Section 6.2 is shown along with the surface approximating the total data set. The survey has the highest priority score in the area so all points are kept. The points are coloured according to the distance to the surface and in this case, all points with a distance less than 26 meters are green. However, some points are still red. They lie below the surface at a considerable distance. These points also fail to fall into the scanline pattern followed by most points in the data set. The identified points are clearly outliers. Despite the huge distance between the surface and the outliers, the average distance between all points and the surface is less than 0.2 meters.

9.10 #128 DEFINE REGULAR TILING

This service is essential in preparation for the chain of services in MS1 and MS2. It defines the regular tiling providing an appropriate number of input points for surface generation in the two work flows.

9.10.1 Functionality & Algorithm

The service works entirely on metadata. Based on default values for a typical and a maximum number of points for surface generation, the number of tiles in x and y direction is defined by an iterative procedure taking the number of points and the bounding boxes of each data survey into account.

9.10.2 Quality

The number of points in a tile is important for the performance of Service 122. Too many points lead to poor performance and inconveniently large surfaces. If Service 128 is too careful avoiding large tiles, the number of surfaces produced in the work flow (MS1 or MS2) gets very high. The quality measure for this service is related to finding the right balance between tiled point cloud sizes and number of surfaces. The test cases run so far, indicate that the current balance is acceptable.

9.10.3 Scalability

The service is single node and single threaded, but is, as it only considers metadata, not subject to large data sizes. The service has been tested with data sizes between 0.002 MB and 0.064 MB. The execution on one node of the Fraunhofer cloud takes between 0.66 and 5.13 seconds and neither the data sizes nor the result give any valid input for scalability analysis.

9.10.4 Degree of Human Intervention

No human intervention is required.

9.10.5 Examples

Section 7.2 gives an example of a tiled data set. For one data survey of about 65.5 million points, a regular tiling with 12 times 6 tiles is defined. A more typical setting is shown in Figure 1. For this data collection, 60 data sets with a total of about 16.8 million points are divided into 16 tiles.

9.11 #129 PERFORM REGULAR TILING

This service represents the next step in preparation for MS1 and MS2 after Service 128.

9.11.1 Functionality & Algorithm

Service 129 distributes the points from one data survey into tiles according to the tile distribution defined by Service 128. The data points are read from file, sorted and written to separate tile files.

9.11.2 Quality

This service is not subject to any quality measures.

9.11.3 Scalability

The service is prepared to be distributed to different nodes with respect to given data surveys. It has been with collections of data surveys adding up to 105 MB, 544 MB, 1753 MB, 3491 MB and 6826 MB divided into 10, 60, 9, 31 and 317 data surveys, respectively. The collections are tested on a varying number of nodes, dependent on the survey configuration.

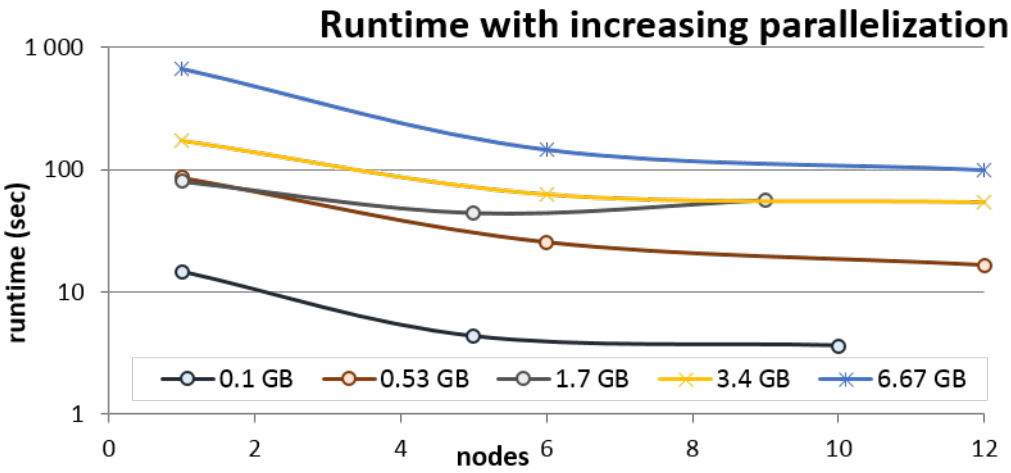


Figure 45: Runtime for each data collection on an increasing number of nodes

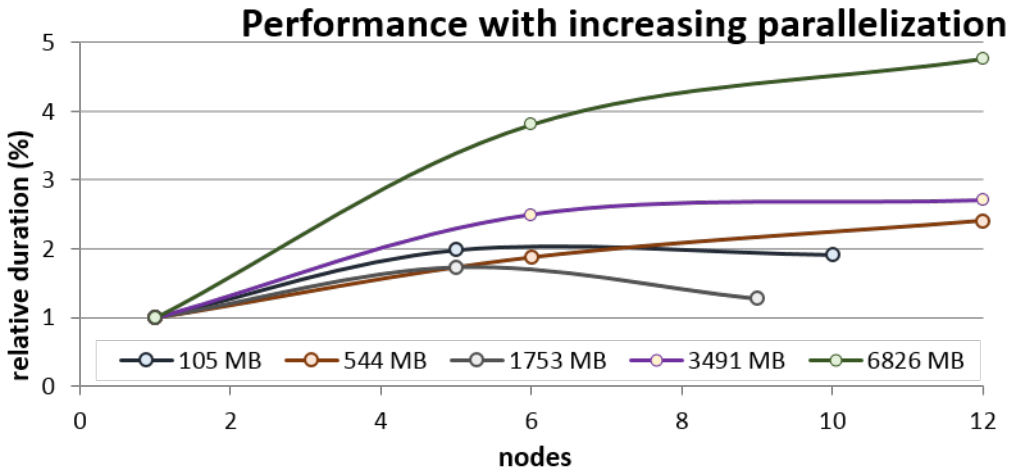


Figure 46: Performance with increasing parallelization for all data sets compared to a linear scalability

As shown in Figure 46 the runtime is reduced with an increasing number of nodes, but possibly less than what we would expect. The service is parallelized with regard to data surveys and not the number of data points and the number of points for each survey varies for the test data set. Few data surveys clearly reduces the effect on parallelization.

The performance in general improves with increasing parallelization, see Figure 45, but less than a linear behaviour as indicated by the dotted line. The service is parallelized on data surveys, not on data points, and the number of data points varies for each survey. The 1753 MB collection is for instance assembled from a few large data surveys, 9 in total. This removes all flexibility when the collection is parallelized on 9 nodes and the parallelization factor is lost. The same effect can be recognized especially for the smallest collection with 10 data surveys while the largest one has 317 data surveys and performs better with parallelization.

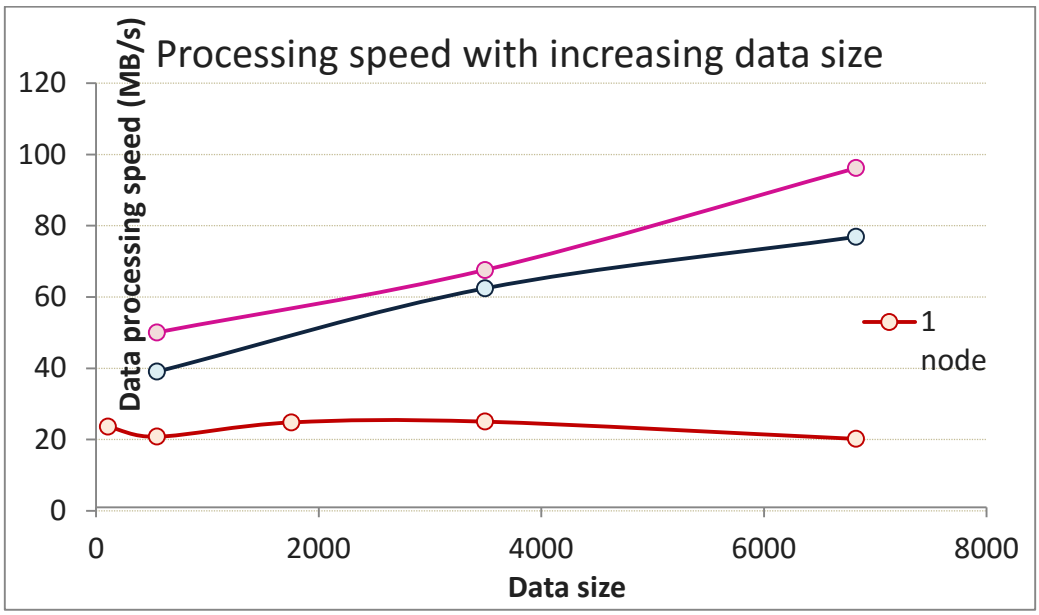


Figure 47: Processing speed measured in MB per seconds for increasing data size and 1, 6 and 12 nodes

The scenarios with fewest data survey files compared to parallelization nodes are omitted from the graph in Figure 47. The remaining shows a close to constant behaviour for 1 node, but an increasing data processing speed for higher data sizes with 6 and 12 nodes. The relation between data size and number of files is a probable cause for this behaviour, but more studies are required to draw any conclusions.

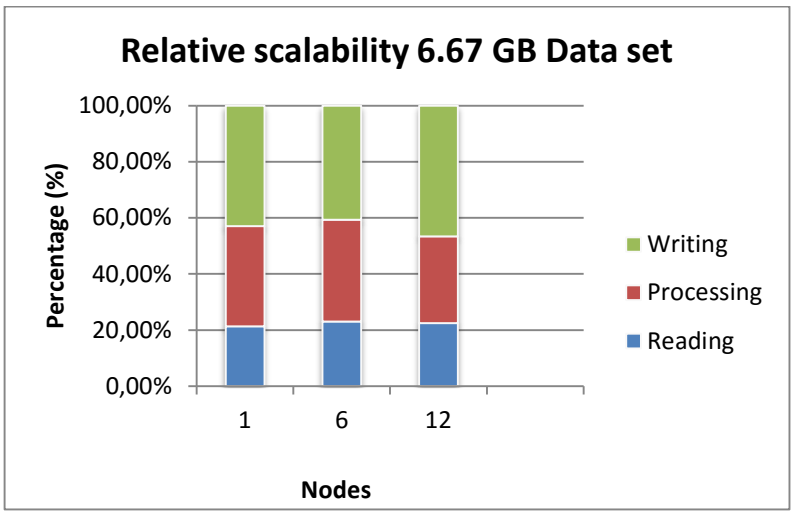


FIGURE 48: Relative distribution of effort for reading, processing and writing for the largest data set

For each survey, the data is read, points are sorted according to their x- and y-coordinates and written to separate files for each tile in a regular tiling defined by Service 128. None of these tasks is dominant and the distribution of the execution into several nodes does not have a large effect as can be seen from Figure 48. The service reads and writes equally many points, but the written points are distributed into many more files. Furthermore, one metadata file will be created for each tile data file.

9.11.4 Degree of Human Intervention

No human intervention is required.

9.11.5 Examples

Figure 4 in Section 7.2 provides an example of a tiled data set.

9.12 #136 METADATA FOR DATA SURVEYS

This is the first service in the MS1 and MS2 workflows.

9.12.1 Functionality & Algorithm

The service defines the metadata structure for the workflows. All given data surveys are read from file, the number of points for each survey is counted and the associated bounding boxes are computed. This information is stored as metadata for use by subsequent services.

9.12.2 Quality

This service is not subject to any quality measures.

9.12.3 Scalability

The services is single threaded and runs on one node. It is tested for data collections with size 105 MB, 544 MB, 1753 MB, 3491 MB and 6826 MB data which are distributed into 10, 60, 9, 31 and 317 data surveys, respectively.

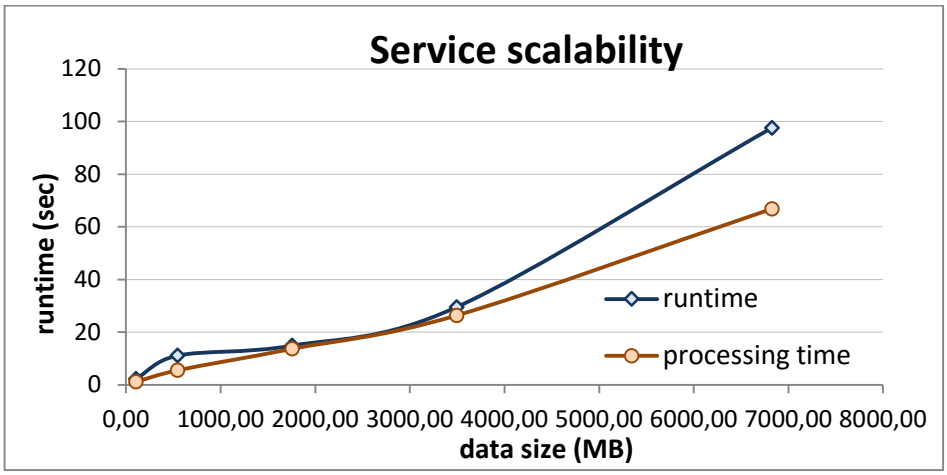


FIGURE 49: Runtime and processing time, i.e. Runtime minus system overhead, for increasing data sizes

The task performed by this service is very simple and in essence consists of reading a number of files for each test scenario and do some statistics. The deviation from linearity seen in Figure 49 is probably caused by how the points are distributed into the initial data surveys.

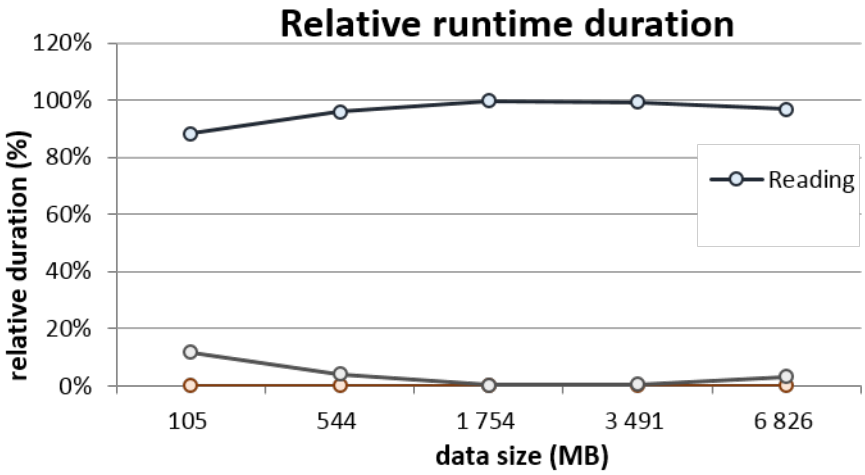


FIGURE 50: The runtime divided into reading, processing and writing for increasing data sizes

The main bulk of the job is done during reading (Figure 51) when the number of points are counted and the bounding box related to the point set is updated. Processing is mainly concerned with preparing for output and only metadata with a much smaller data size than the input point clouds, is written.

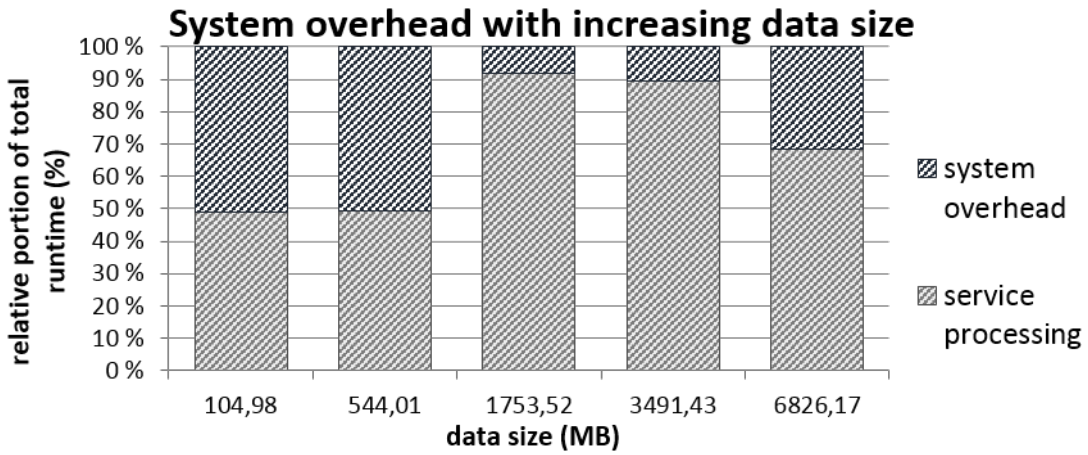


FIGURE 51: Percentage of system overhead for all data collection

It is no clear relation between how the data points are distributed into few or many data files and the system overhead, but there is a tendency that the handling of many small data files gives rice to a larger overhead than fewer and larger files.

9.12.4 Degree of Human Intervention

The service requires no human intervention.

9.13 POST PROCESSING OF THE MS1 AND MS2 WORKFLOWS

Two services intended for preparing the output of the MS1 and MS2 workflows for further processing by applications that are not familiar with the concept of LR B-splines are currently under development. The services are not yet stored in artifactory.

9.13.1 Prepare for DEM generation

This service ensures that LR B-spline surfaces generated by the workflow are defined in the correct coordinate system. If the data is initially given in WGS84, the points are scaled to get better correspondence between x-, y- and z-coordinates during the computations. This service reparametrizes the surfaces and creates metadata for the surface collection. Furthermore, distance field information related to the point cloud tiles may be collected for each data survey.

9.13.2 LR B-spline surface collection to raster

A unified raster is created from a regular collection of LR B-spline surfaces. The raster is stored in the GeoTIFF format.

10 REFERENCES

1. G. Patane, A. Cerri, V. Skytt, S. Pittaluga, S. Biasotti, D. Sobrero, T.Dokken, M. Spagnuolo. Comparing Methods for the Approximation of Rainfall Fields in Environmental Applications. Submitted to Elsevier (2016)
2. Jonathan Richard Shewchuk, Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator, in ``Applied Computational Geometry: Towards Geometric Engineering'' (Ming C. Lin and Dinesh Manocha, editors), volume 1148 of Lecture Notes in Computer Science, pages 203-222, Springer-Verlag, Berlin, May 1996.
3. Michael Garland and Paul S. Heckbert. 1997. Surface simplification using quadric error metrics. In Proceedings of the 24th annual conference on Computer graphics and interactive techniques (SIGGRAPH '97). ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 209-216. DOI=http://dx.doi.org/10.1145/258734.258849
4. T. Dokken, K.F. Pettersen, T. Lyche. Polynomial splines over locally refined box-partitions. Computer Aided Geometric Design 2013; 30(3): 331-356; http://dx.doi.org/10.1016/j.cagd.2012.12.005
5. Skytt, Vibeke, Oliver Barrowclough, and Tor Dokken. "Locally refined spline surfaces for representation of terrain data." Computers & Graphics 49 (2015): 58-68. Special Issue on Processing of Large Geospatial Data
6. Dokken, T., Skytt, V., & Barrowclough, O. (2015). Locally Refined Splines Representation for Geospatial Big Data. The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, 40(3), 565.

11 APPENDIX A - SERVICE TABLES

IQmulus Service #63			
Name of the metadata	**Content expected**		**Motivation/comments**
Service Acronym	ExtractingSurfaceNormalsFromPointClouds		*Unique identifier of the service; necessary to call it within user-defined workflows*
Description	Extracting surface normals from point clouds using PCA. The current implementation assumes that the normals are pointing upwards (+z direction) so it may not be suitable for mobile mapping applications. Built with the parallel processing engine Apache Spark, this is a completely parallelized service.		*Brief textual description of the service: what it provides, what can be used for. This text could be used as a short "help text" in the User Interfaces of IQmulus*
Service functionality	**Input: <data representation and format>**	**Point cloud in las format**	*Include here the input/output of the service, the name of the functionality (e.g., registration, fusion), and input parameters (if any). In the future, we may consider defining a taxonomy of the functionalities to harmonize the terminology used to fill this field.*
	Input parameters: *[optional]*	**Number of nearest neighbors** **Bits per dimension in index**	
	Output: <data representation and format>	**Enriched point cloud in ply format**	
	Functionality of the service: <text>	Extracting surface normals	
Algorithm	Apply PCA on neighbourhood of each point. The third principal component is the normal.		*The same functionality may in principle be implemented by different algorithms.*
Implementation details	**Implementation language**	Python	*Include information related to the implementation of the service, such as language (e.g., C, C++, etc); dependencies with other libraries (e.g., ANN library); constraints on the operating systems, and visualization modalities of the output.*
	Dependencies with other libraries	PCL	
	Operating system	Linux	
	Visualization modalities of the output		
IQmulus Data	**Available IQmulus input data**:		*If there are examples of data that could serve as an example, then include here*

		their identifiers as in the data table in eRoom (useful to test the service]
Service characteristics	**Accuracy:** Very high accuracy for smooth point clouds.	*These fields are necessary to document all the characteristics of the services that are important to assess the quality of the results. Also, these fields could be used to select a specific service among more services that implement the same functionality but with different characteristics. See the following box.*
	Robustness: Results depend on number of points per partition and neighbourhood size.	
	Computational time in relation to data size:	
	Locality/globality of the algorithm: Local	
Alternatives		*List other services (if any) that have the same functionality, have the same input/output but use a different algorithm or have different features*
Related use cases		*Mention the user stories related to the usage of the service (see D4.1.1) and related use cases uploaded on Red Mine*
Responsible Partner	Christian Alis, UCL (c.alis@ucl.ac.uk)	*Partner ID and responsible person (include email)*
Involved Partners	UCL	

IQmulus Service #110			
Name of the metadata	**Content expected**		Motivation/comments
Service Acronym	3DDelaunayTriangulation		
Description	Computing the 3D Delaunay triangulation of a given point set in R^3.		
Service functionality	**Input: <data representation and format>**	Point cloud in ply format	
	Input parameters:	Sampling rate (0, 1]	Convex hull points of

			the tiles are sampled according to this parameter.
	Output: <data representation and format>	3D Delaunay triangulation in CGAL 3D triangulation file format	
	Functionality of the service: <text>	Compute 3D Delaunay triangulation	
Algorithm	Compute the Delaunay triangulation of each tile separately and then merge the neighbouring tiles.		
Implementation details	**Implementation language**	C++	
	Dependencies with other libraries	CGAL	
	Operating system	Linux	
	Visualization modalities of the output		
IQmulus Data			
Service characteristics	**Accuracy:** The accuracy depends on the sampling parameter. If the sampling parameter is set to 1, then the computed triangulations are guaranteed to be Delaunay.		
	Robustness: The algorithm uses the Exact predicates inexact constructions kernel of the CGAL library.		Since no new points are computed, the utilized geometry kernel is enough to guarantee robustness.
	Computational time in relation to data size: Will be analyzed.		
	Locality/globality of the algorithm: Global		
Alternatives			
Related use cases	Watertight surface reconstruction, US		
Responsible Partner	IGN		
Involved Partners			

IQmulus Service #117			
Name of the metadata	**Content expected**		**Motivation/comments**
Service Acronym	LRField2Triang		*Unique identifier of the service; necessary to call it within user-defined workflows*
Description	Evaluate an LR B-spline field parameterized on the x- and y-values of a corresponding terrain at the vertices of a triangulated surface representing the same terrain. The result is stored as an enhanced triangulation.		*Brief textual description of the service: what it provides, what can be used for. This text could be used as a short "help text" in the User Interfaces of IQmulus*
Service functionality	**Input: <data representation and format>**	LR B-spline surface in g2 format Triangulated surface in ply format	*Include here the input/output of the service, the name of the functionality (e.g., registration, fusion), and input parameters (if any). In the future, we may consider defining a taxonomy of the functionalities to harmonize the terminology used to fill this field.*
	Input parameters: *[optional]*		
	Output: <data representation and format>	Enhanced triangulation in ply format	
	Functionality of the service: <text>	Evaluate LR B-spline field	
Algorithm	Expand the surface to tensor-product format and evaluate in the triangle vertices.		*The same functionality may in principle be implemented by different algorithms.*
Implementation details	**Implementation language**	C++	*Include information related to the implementation of the service, such as language (e.g., C, C++, etc); dependencies with other libraries (e.g., ANN library); constraints on the operating systems, and visualization modalities of the output.*
	Dependencies with other libraries	GoTools and Rply integrated in the executable	
	Operating system	Linux	
	Visualization modalities of the output	3D visualization of the rainfall map with different colours	
IQmulus Data	**Available IQmulus input data**: Dataset 22, Dataset 27, Dataset 26, Dataset 42, 43 The data sets need to be processed by other services before appliying Service #117		*If there are examples of data that could serve as an example, then include here their identifiers as in the data table in eRoom (useful to test the service]*

Service characteristics	**Accuracy:** Spline evaluation is an exact operation	*These fields are necessary to document all the characteristics of the services that are important to assess the quality of the results. Also, these fields could be used to select a specific service among more services that implement the same functionality but with different characteristics. See the following box.*
	Robustness: No particular robustness issues exists	
	Computational time in relation to data size: After some time for preprocessing, the spline evaluation is linear in the number of data points	
	Locality/globality of the algorithm: Local. The basis function of the spline surface have local support.	
Alternatives	<List of Service IDs> Service #40 and #67	*List other services (if any) that have the same functionality, have the same input/output but use a different algorithm or have different features*
Related use cases	Use case (IQmulus) #1107	*Mention the user stories related to the usage of the service (see D4.1.1) and related use cases uploaded on Red Mine*
Responsible Partner	Vibeke Skytt, SINTEF (vibeke.skytt@sintef.no)	*Partner ID and responsible person (include email)*
Involved Partners	CNR-IMATI	

IQmulus Service #122			
Name of the metadata	**Content expected**		**Motivation/comments**
Service Acronym	OneTile2Surf Generate spline surfaces from parameterized data		*Unique identifier of the service; necessary to call it within user-defined workflows*
Description	Compute one LR B-spline surface for one tile in a data set preprocessed by services 128 and service 129 to generate a regular tiling, by approximating a point cloud with parameter information.		*Brief textual description of the service: what it provides, what can be used for. This text could be used as a short "help text" in the User Interfaces of IQmulus*
Service	**Input: <data**	Tiled point cloud (.las, .xyz or	*Include here the input/output*

<table>
<tr><td rowspan="4">functionality</td><td>representation and format></td><td>.g2), corresponding metadata</td><td rowspan="4">of the service, the name of the functionality (e.g., registration, fusion), and input parameters (if any). In the future, we may consider defining a taxonomy of the functionalities to harmonize the terminology used to fill this field.</td></tr>
<tr><td>Input parameters: [optional]</td><td>• Approximation tolerance
• Number of iterations in adaptive procedure</td></tr>
<tr><td>Output: <data representation and format></td><td>• LR B-spline surface in g2-format
• Accuracy information
• Update metadata</td></tr>
<tr><td>Functionality of the service: <text></td><td>Surface approximation</td></tr>
<tr><td>Algorithm</td><td colspan="2">An adaptive procedure is applied, where at each step the current surface is refined in areas where a prescribed tolerance is not met, then an updated approximating surface is computed. Two different approximation methods are used depending on the stage of the computation and optional input parameters: least squares approximation and multi-level B-spline approximation applied on LR B-spline surfaces (LR-MBA)</td><td>The same functionality may in principle be implemented by different algorithms.</td></tr>
<tr><td rowspan="4">Implementation details</td><td>Implementation language</td><td>C++</td><td rowspan="4">Include information related to the implementation of the service, such as language (e.g., C, C++, etc); dependencies with other libraries (e.g., ANN library); constraints on the operating systems, and visualization modalities of the output.</td></tr>
<tr><td>Dependencies with other libraries</td><td>liblas, rply, OpenMP</td></tr>
<tr><td>Operating system</td><td>Linux</td></tr>
<tr><td>Visualization modalities of the output</td><td>Visualization of LR B-spline surface. IQmulusViz</td></tr>
<tr><td>IQmulus Data</td><td colspan="2">Available IQmulus input data:

Test data set made available by HR Wallingford</td><td>If there are examples of data that could serve as an example, then include here their identifiers as in the data table in eRoom (useful to test the service]</td></tr>
<tr><td rowspan="3">Service characteristics</td><td colspan="2">Accuracy: Depends on the characteristics of the input points and the number of iterations in the adaptive procedure. In most cases 98-99% of the points will lie closer to the surface than 0.5m.</td><td rowspan="3">These fields are necessary to document all the characteristics of the services that are important to assess the quality of the results. Also, these fields could be used to select a specific service among more services that implement the same functionality but with different characteristics. See</td></tr>
<tr><td colspan="2">Robustness: Small perturbations in the point set will give small changes in the surface.</td></tr>
<tr><td colspan="2">Computational time in relation to data size:
Keeping the tolerance and the number of iterations fixed, the service scales roughly linear, but the performance is affected by the amount of detail in the</td></tr>
</table>

	data set. **Locality/globality of the algorithm:** Global, but some operations are local	*the following box.*
Alternatives	<List of Service IDs> Service #9 for data sets that can be processed on one node	*List other services (if any) that have the same functionality, have the same input/output but use a different algorithm or have different features*
Related use cases	Marine scenario MS1 and MS2	*Mention the user stories related to the usage of the service (see D4.1.1) and related use cases uploaded on Red Mine*
Responsible Partner	SINTEF Vibeke Skytt, Vibeke.Skytt@sintef.no	*Partner ID and responsible person (include email)*
Involved Partners	HR Wallingford	

IQmulus Service #123			
Name of the metadata	**Content expected**		**Motivation/comments**
Service Acronym	deconflictionOneTile		*Unique identifier of the service; necessary to call it within user-defined workflows*
Description	Apply deconfliction for one tile in a regularly tiled collection of data surveys. Groups of points being found to be in conflict with the data survey with highest priority are dismissed for further processing.		*Brief textual description of the service: what it provides, what can be used for. This text could be used as a short "help text" in the User Interfaces of IQmulus*
Service functionality	**Input: <data representation and format>**	Tiled data surveys (.las, .ply, .txt) LR B-spline reference surface (.g2) Metadata including priority score	*Include here the input/output of the service, the name of the functionality (e.g., registration, fusion), and input parameters (if any). In the future, we may consider defining a taxonomy of the functionalities to harmonize the terminology used to fill this field.*
	Input parameters: *[optional]*	Distance threshold	
	Output: <data	Kept and dismissed points	

	representation and format>	from the input data surveys (.las, .ply, .txt) Updated metadata	
	Functionality of the service: <text>	Deconfliction	
Algorithm	Compute distance between data surveys and the reference surface. Compare distance information from different surveys and decide if subsets of data points are in conflict with the locally highest prioritized survey.		*The same functionality may in principle be implemented by different algorithms.*
Implementation details	**Implementation language**	C++	*Include information related to the implementation of the service, such as language (e.g., C, C++, etc); dependencies with other libraries (e.g., ANN library); constraints on the operating systems, and visualization modalities of the output.*
	Dependencies with other libraries	liblas, rply	
	Operating system	Linux	
	Visualization modalities of the output	Visualization of LR B-spline surface and distance fields. IQmulusViz	
IQmulus Data	**Available IQmulus input data**: Test data sets made available by HR Wallingford		*If there are examples of data that could serve as an example, then include here their identifiers as in the data table in eRoom (useful to test the service]*
Service characteristics	**Accuracy:** The quality is under investigation. Accuracy is not an adequate measure in this context		*These fields are necessary to document all the characteristics of the services that are important to assess the quality of the results. Also, these fields could be used to select a specific service among more services that implement the same functionality but with different characteristics. See the following box.*
	Robustness: Must be investigated		
	Computational time in relation to data size: Scalability testing will be performed		
	Locality/globality of the algorithm: Global within each data tile. Local with respect to the tile.		
Alternatives	<List of Service IDs>		*List other services (if any) that have the same functionality, have the same input/output but use a different algorithm or have*

		different features
Related use cases	Marine Scenario MS1	*Mention the user stories related to the usage of the service (see D4.1.1) and related use cases uploaded on Red Mine*
Responsible Partner	SINTEF Vibeke Skytt, Vibeke.Skytt@sintef.no	*Partner ID and responsible person (include email)*
Involved Partners	HR Wallingford	

IQmulus Service #124			
Name of the metadata	**Content expected**		**Motivation/comments**
Service Acronym	UpdateSplineOneTile		*Unique identifier of the service; necessary to call it within user-defined workflows*
Description	Update LR B-spline surface with respect to data points belonging to one tile in a regular tiling		*Brief textual description of the service: what it provides, what can be used for. This text could be used as a short "help text" in the User Interfaces of IQmulus*
Service functionality	**Input: <data representation and format>**	LR B-spline surface (.g2) Tile data surveys (.las, .txt, .ply)	*Include here the input/output of the service, the name of the functionality (e.g., registration, fusion), and input parameters (if any). In the future, we may consider defining a taxonomy of the functionalities to harmonize the terminology used to fill this field.*
	Input parameters: *[optional]*	Approximation tolerance Number of iterations in adaptive procedure	
	Output: <data representation and format>	Updated LR B-spline surface (.g2)	
	Functionality of the service: <text>	Spline approximation	
Algorithm	Starting from the input surface an adaptive procedure is applied, where at each step the current surface is refined in areas where a prescribed tolerance is not met, then an updated approximating surface is computed.		*The same functionality may in principle be implemented by different algorithms.*

<table>
<tr><td rowspan="4">Implementation details</td><td>Implementation language</td><td>C++</td><td rowspan="4">Include information related to the implementation of the service, such as language (e.g., C, C++, etc); dependencies with other libraries (e.g., ANN library); constraints on the operating systems, and visualization modalities of the output.</td></tr>
<tr><td>Dependencies with other libraries</td><td>liblas, rply, OpenMP</td></tr>
<tr><td>Operating system</td><td>Linux</td></tr>
<tr><td>Visualization modalities of the output</td><td>Visualization of LR B-spline surface. IQmulusViz</td></tr>
<tr><td>IQmulus Data</td><td colspan="2">Available IQmulus input data:

Test data set made available by HR Wallingford</td><td>If there are examples of data that could serve as an example, then include here their identifiers as in the data table in eRoom (useful to test the service]</td></tr>
<tr><td rowspan="4">Service characteristics</td><td colspan="2">Accuracy: Depends on the characteristics of the input points and the number of iterations in the adaptive procedure.</td><td rowspan="4">These fields are necessary to document all the characteristics of the services that are important to assess the quality of the results. Also, these fields could be used to select a specific service among more services that implement the same functionality but with different characteristics. See the following box.</td></tr>
<tr><td colspan="2">Robustness: Small perturbations in the point set will give small changes in the surface.</td></tr>
<tr><td colspan="2">Computational time in relation to data size:
Keeping the tolerance and the number of iterations fixed, the service scales roughly linear, but the performance is affected by the amount of detail in the data set.</td></tr>
<tr><td colspan="2">Locality/globality of the algorithm:

Global, but some operations are local</td></tr>
<tr><td>Alternatives</td><td colspan="2"><List of Service IDs></td><td>List other services (if any) that have the same functionality, have the same input/output but use a different algorithm or have different features</td></tr>
<tr><td>Related use cases</td><td colspan="2">Marine scenario MS1</td><td>Mention the user stories related to the usage of the service (see D4.1.1) and related use cases uploaded on Red Mine</td></tr>
</table>

Responsible Partner	SINTEF Vibeke Skytt, Vibeke.Skytt@sintef.no	*Partner ID and responsible person (include email)*
Involved Partners	HR Wallingford	

IQmulus Service #125			
Name of the metadata	**Content expected**		**Motivation/comments**
Service Acronym	TrimSurfOneTile		*Unique identifier of the service; necessary to call it within user-defined workflows*
Description	Bound an LR B-spline surface with one or more loops describing the extent of the point cloud tiles from which the surface was generated. The result will be that the domain of the trimmed surface will correspond to the area covered by the point clouds.		*Brief textual description of the service: what it provides, what can be used for. This text could be used as a short "help text" in the User Interfaces of IQmulus*
Service functionality	**Input: <data representation and format>**	LR B-spline surface: g2 format Point cloud: las, xyz, ply or g2-format Metadata	*Include here the input/output of the service, the name of the functionality (e.g., registration, fusion), and input parameters (if any). In the future, we may consider defining a taxonomy of the functionalities to harmonize the terminology used to fill this field.*
	Input parameters: *[optional]*	Tightness parameter. How close to the point cloud should the trimming curve go? A number in the interval [1,7].	
	Output: <data representation and format>	Trimmed B-spline surface: g2-format Updated metadata	
	Functionality of the service: <text>	Trimming of LR B-spline surface	
Algorithm	The point cloud is recursively divided into sub clouds. Each cloud is bounded by a rectangle and the outer boundaries of these rectangles are collected into inner and outer trimming loops. These loops are approximated by parameter domain spline curves which are used to limit the surface.		*The same functionality may in principle be implemented by different algorithms.*
Implementation details	**Implementation language**	C++	*Include information related to the implementation of the service, such as language (e.g., C, C++, etc); dependencies with other libraries (e.g., ANN library); constraints on the*
	Dependencies with	liblas, rply	

	other libraries		*operating systems, and visualization modalities of the output.*
	Operating system	Linux	
	Visualization modalities of the output	Visualization of the trimmed LR B-spline surface, IQmulusViz	
IQmulus Data	**Available IQmulus input data**: Test data set made available by HR Wallingford		*If there are examples of data that could serve as an example, then include here their identifiers as in the data table in eRoom (useful to test the service]*
Service characteristics	**Accuracy:** The domain is bounded by the constructed trimming loop. It is <u>not</u> interpolated. Accuracy depends on the tightness parameter. An accurate bound can be obtained if the input points are dense and have a regular pattern, otherwise a rougher trimming must be applied.		*These fields are necessary to document all the characteristics of the services that are important to assess the quality of the results. Also, these fields could be used to select a specific service among more services that implement the same functionality but with different characteristics. See the following box.*
	Robustness: Unexpected results may occur for very scattered data sets. Small changes in the data points will induce none or small changes in the result.		
	Computational time in relation to data size: The computation is linear with respect to data size provided that the tightness parameter and the complexity of the domain represented by the point cloud are kept fixed.		
	Locality/globality of the algorithm: In essence global. Local results are merged in a recursive procedure.		
Alternatives	<List of Service IDs> Service #90 for data sets of a size where tiling is not required		*List other services (if any) that have the same functionality, have the same input/output but use a different algorithm or have different features*
Related use cases	Marine Scenario MS1 and MS2		*Mention the user stories related to the usage of the service (see D4.1.1) and related use cases uploaded on Red Mine*
Responsible Partner	SINTEF Vibeke Skytt: Vibeke.Skytt@sintef.no		*Partner ID and responsible person (include email)*

Involved Partners	HR Wallingford	

IQmulus Service #126			
Name of the metadata	**Content expected**		**Motivation/comments**
Service Acronym	DistanceFieldOneTileSurf		*Unique identifier of the service; necessary to call it within user-defined workflows*
Description	Compute the distance field between a point cloud tile and a corresponding LR B-spline surface		*Brief textual description of the service: what it provides, what can be used for. This text could be used as a short "help text" in the User Interfaces of IQmulus*
Service functionality	**Input: <data representation and format>**	LR B-spline surface, possibly trimmed (.g2) Point cloud (.las, .txt, .ply, .g2)	*Include here the input/output of the service, the name of the functionality (e.g., registration, fusion), and input parameters (if any). In the future, we may consider defining a taxonomy of the functionalities to harmonize the terminology used to fill this field.*
	Input parameters: *[optional]*		
	Output: <data representation and format>	Summary of distance information Distance field given as x, y, z and distance (.txt or .ply)	
	Functionality of the service: <text>	Distance field computation	
Algorithm	Define an intermediate data structure to enable fast access to the basis functions of the LR B-spline surface. Evaluate the surface in the x- and y-values of all data points and compute the distance.		*The same functionality may in principle be implemented by different algorithms.*
Implementation details	**Implementation language**	C++	*Include information related to the implementation of the service, such as language (e.g., C, C++, etc); dependencies with other libraries (e.g., ANN library); constraints on the operating systems, and visualization modalities of the output.*
	Dependencies with other libraries	liblas, OpenMP rply and GoTools are integrated in the executable	
	Operating system	Linux	
	Visualization modalities of the	Visualization of distance	

	output	field, IQmulusViz	
IQmulus Data	**Available IQmulus input data**: Test data set made available by HR Wallingford		*If there are examples of data that could serve as an example, then include here their identifiers as in the data table in eRoom (useful to test the service]*
Service characteristics	**Accuracy:** Evaluation according to machine precision		*These fields are necessary to document all the characteristics of the services that are important to assess the quality of the results. Also, these fields could be used to select a specific service among more services that implement the same functionality but with different characteristics. See the following box.*
	Robustness: Small changes in the surface or point cloud will be exactly reflected in the distance field.		
	Computational time in relation to data size: The computation is linear in the number of data points. Pre-processing time depends on the surface size.		
	Locality/globality of the algorithm: The evaluation is local, the pre-processing is global		
Alternatives	<List of Service IDs> Service #88 for data sets of a size where tiling is not required		*List other services (if any) that have the same functionality, have the same input/output but use a different algorithm or have different features*
Related use cases	Marine scenario MS1 and MS2		*Mention the user stories related to the usage of the service (see D4.1.1) and related use cases uploaded on Red Mine*
Responsible Partner	SINTEF Vibeke Skytt: Vibeke.Skytt@sintef.no		*Partner ID and responsible person (include email)*
Involved Partners			

IQmulus Service #128		
Name of the metadata	**Content expected**	**Motivation/comments**
Service Acronym	DefineRegularTiling	*Unique identifier of the service; necessary to call it within user-defined workflows*

Description	Define dimension of a regular tiling of a set of data surveys based on the number of points and spatial extent for each data set,, utility		*Brief textual description of the service: what it provides, what can be used for. This text could be used as a short "help text" in the User Interfaces of IQmulus*
Service functionality	**Input: <data representation and format>**	Metadata .json	*Include here the input/output of the service, the name of the functionality (e.g., registration, fusion), and input parameters (if any). In the future, we may consider defining a taxonomy of the functionalities to harmonize the terminology used to fill this field.*
	Input parameters: *[optional]*		
	Output: <data representation and format>	Updated set of metadata .json	
	Functionality of the service: <text>	Define regular tiling	
Algorithm	Iteratively set the tiling dimension and estimate the number of points for each tile		*The same functionality may in principle be implemented by different algorithms.*
Implementation details	**Implementation language**	C++	*Include information related to the implementation of the service, such as language (e.g., C, C++, etc); dependencies with other libraries (e.g., ANN library); constraints on the operating systems, and visualization modalities of the output.*
	Dependencies with other libraries		
	Operating system	Linux	
	Visualization modalities of the output		
IQmulus Data	**Available IQmulus input data**: Test data set made available by HR Wallingford		*If there are examples of data that could serve as an example, then include here their identifiers as in the data table in eRoom (useful to test the service]*
Service characteristics	**Accuracy:** Not an issue		*These fields are necessary to document all the characteristics of the services that are important to assess the quality of the results. Also, these fields could be used to select a specific service among more services that implement the same functionality but with different characteristics.*
	Robustness: The number of points for each tile depends on the initial distribution of points		
	Computational time in relation to data size: The data set is very small (metadata) and		

	computation time is not an issue	*See the following box.*
	Locality/globality of the algorithm: Global	
Alternatives	<List of Service IDs>	*List other services (if any) that have the same functionality, have the same input/output but use a different algorithm or have different features*
Related use cases	Marine Scenario MS1 and MS2	*Mention the user stories related to the usage of the service (see D4.1.1) and related use cases uploaded on Red Mine*
Responsible Partner	SINTEF Vibeke Skytt: Vibeke.Skytt@sintef.no	*Partner ID and responsible person (include email)*
Involved Partners		

IQmulus Service #129			
Name of the metadata	**Content expected**		**Motivation/comments**
Service Acronym	RegularTiling		*Unique identifier of the service; necessary to call it within user-defined workflows*
Description	Perform regular tiling for a collection of data surveys, utility		*Brief textual description of the service: what it provides, what can be used for. This text could be used as a short "help text" in the User Interfaces of IQmulus*
Service functionality	**Input: <data representation and format>**	Data surveys (.csv, .las, .txt, .ply) Metadata (.json)	*Include here the input/output of the service, the name of the functionality (e.g., registration, fusion), and input parameters (if any). In the future, we may consider defining a taxonomy of the functionalities to harmonize the terminology used to fill this field.*
	Input parameters: *[optional]*		
	Output: <data representation and format>	Tiled data surveys (.csv, .las, .txt, .ply) Updated metadata (.json)	
	Functionality of		

	the service: <text>		
Algorithm	For each data set, sort the points according to x- and y- coordinates and extract the tiles according to given regular discretization of the domain		*The same functionality may in principle be implemented by different algorithms.*
Implementation details	**Implementation language**	C++	*Include information related to the implementation of the service, such as language (e.g., C, C++, etc); dependencies with other libraries (e.g., ANN library); constraints on the operating systems, and visualization modalities of the output.*
	Dependencies with other libraries	Liblas, rply	
	Operating system	Linux	
	Visualization modalities of the output		
IQmulus Data	**Available IQmulus input data**: Test data set made available by HR Wallingford		*If there are examples of data that could serve as an example, then include here their identifiers as in the data table in eRoom (useful to test the service]*
Service characteristics	**Accuracy:** Not an issue		*These fields are necessary to document all the characteristics of the services that are important to assess the quality of the results. Also, these fields could be used to select a specific service among more services that implement the same functionality but with different characteristics. See the following box.*
	Robustness: Not an issue		
	Computational time in relation to data size: Approximately linear		
	Locality/globality of the algorithm: Global for each data survey		
Alternatives	<List of Service IDs>		*List other services (if any) that have the same functionality, have the same input/output but use a different algorithm or have different features*
Related use cases	Marine Scenario, MS1 and MS2		*Mention the user stories related to the usage of the service (see D4.1.1) and related use cases uploaded on Red Mine*
Responsible Partner	SINTEF Vibeke Skytt, Vibeke.Skytt@sintef.no		*Partner ID and responsible person (include email)*

Involved Partners		

IQmulus Service #136			
Name of the metadata	**Content expected**		**Motivation/comments**
Service Acronym	MakeMetaData		*Unique identifier of the service; necessary to call it within user-defined workflows*
Description	Create metadata for a set of data surveys, utility		*Brief textual description of the service: what it provides, what can be used for. This text could be used as a short "help text" in the User Interfaces of IQmulus*
Service functionality	**Input: <data representation and format>**	Data surveys (.csv, .las, .txt, .ply)	*Include here the input/output of the service, the name of the functionality (e.g., registration, fusion), and input parameters (if any). In the future, we may consider defining a taxonomy of the functionalities to harmonize the terminology used to fill this field.*
	Input parameters: *[optional]*		
	Output: <data representation and format>	Metadata (.json)	
	Functionality of the service: <text>	Extract information	
Algorithm	Count the number of points and compute the bounding box for each data set		*The same functionality may in principle be implemented by different algorithms.*
Implementation details	**Implementation language**	C++	*Include information related to the implementation of the service, such as language (e.g., C, C++, etc); dependencies with other libraries (e.g., ANN library); constraints on the operating systems, and visualization modalities of the output.*
	Dependencies with other libraries	Liblas, rply	
	Operating system	Linux	
	Visualization modalities of the output		
IQmulus Data	**Available IQmulus input data**:		*If there are examples of data that could serve as an example, then include here their identifiers as in the data*

	Test data set made available by HR Wallingford	*table in eRoom (useful to test the service]*
Service characteristics	**Accuracy:** Not an issue	*These fields are necessary to document all the characteristics of the services that are important to assess the quality of the results. Also, these fields could be used to select a specific service among more services that implement the same functionality but with different characteristics. See the following box.*
	Robustness: Not an issue	
	Computational time in relation to data size: Linear	
	Locality/globality of the algorithm: Points are traversed one-by-one	
Alternatives	<List of Service IDs>	*List other services (if any) that have the same functionality, have the same input/output but use a different algorithm or have different features*
Related use cases	Marine Scenario, MS1 and MS2	*Mention the user stories related to the usage of the service (see D4.1.1) and related use cases uploaded on Red Mine*
Responsible Partner	SINTEF Vibeke Skytt, Vibeke.Skytt@sintef.no	*Partner ID and responsible person (include email)*
Involved Partners		

1 APPENDIX B - CONFIDENTIAL CONTENT

From page 12, Table 1

48	Multiresolution triangulation	CNR IMATI	LS1	Essential in work flow

107	Lod Extractor	CNR IMATI	LS1	Essential in work flow

From page 15, Table 4

48	Multiresolution triangulation	CNR IMATI	LS1	D4.4.3

107	LOD Extractor	CNR IMATI	LS1	D4.4.3

Multi resolution triangulations offer a balance between data size and level of detail appropriate to the current application.

1.1 #48 MULTI RESOLUTION TRIANGULATION

This service belongs to LS1 and relies on the output of 94. Once points belonging to each basin are identified, 48 will organize them into a multiresolution structure, as detailed in this section. Then, service 107 will take care of extracting in real time only the points that are relevant at a specified level of detail, which can be finally triangulated in a constrained fashion using service 49.

Practically speaking, Service 94 re-orders points belonging to a basin (dispersed in several LAS files) into blocks of consecutive records. The indexes of the first and last record of each block and the LAS file of provenance are saved in the metadata file for that basin. Service 48 then orders points within each block according to their "relevance", i.e., how much they contribute to characterize the appearance of the terrain. Following this ordering, points can be rendered from the most salient on, limiting the resolution visualization according to performance constraints.

1.1.1 Functionality & Algorithm

As a result of the previous processing done by Service 94, we have points organized into basins and blocks, where each block spans contiguous records in the same LAS file. Service 48 proceeds by processing blocks separately, thus allowing to manage big data as separated chunks.

Since the final aim of LS1 is computing a triangulation of the terrain and visualize it at variable resolution, we are going to compute point saliency in a basin and partition basin points into a

number of level of details (10 in our implementation) on the base of their saliency. This computation is performed for each block of the basin separately.

As a measure of point relevance for the terrain appearance, we chose the error that would be introduced into the triangulation of the terrain by deleting that point and its incident triangles and re-triangulate its neighbourhood. The higher the error, the more relevant the point is. Therefore, points are re-ordered within their block in descending relevance order. Other points in the LAS file that do not belong to the block will be kept unchanged. A threshold is set to split the ordered sequence of points into 10 LODs. An example is given in Figure 1.

The algorithm will then produce:

- new LAS files, where blocks of points belonging to basin ID are re-ordered in descending relevance order (LAS files are overwritten)
- a new metadata file, which specifies the point records of the new LAS files belonging to the basin *at a specific level of detail*.

The algorithm proceeds as follows.

For each block of points in a LAS file, we load the points and build a triangulation constrained to the basin boundary using the "liftedDelaunay" algorithm by Shewchuk[A1][A2] [2]. Then, we simplify the mesh according to the error introduced by each removal; points whose removal introduces the minimum error corresponds to points with minimum relevance and are removed first, while boundary points are given infinite error and will be preserved. We use the approach described by Garland in [3] to simplify the triangular mesh with the half-edge collapse primitive. Most salient points (including boundary points) will belong to the lowest LOD (LOD #1) and as such will be present in the triangulation of the basin at every resolution (see Figure 1).

FIGURE 1 – FROM LEFT TO RIGHT, BASIN 81 OF THE LIGURIA REGION IS DEPICTED AT LOD1, LOD5 AND LOD10, RESPECTIVELY.

Processing a basin will in general produce a new collection of sorted LAS files replacing the original ones. Once ordered the block points with respect to decreasing relevance, the service has to identify cut-off points in the sequence to split points into different LODs.

The K-means clustering method is used to group points with similar error (since the deletion error is quadratic, a regular subdivision based on cluster cardinality would be highly unbalanced). As initialization, the block is equally partitioned into ten clusters of equal size, and the representative error for each cluster is set as the average error in the set. Then, the optimization iterates moving points to the adjacent clusters in case their error is closer to another representative. The implementation is particularly efficient since points are ordered by error. At each iteration, the representative error of each cluster is re-computed. A maximum number of iterations is set to 10 to guarantee convergence; also, a check over the minimum size of each sets is introduced to avoid empty clusters. However, an empty cluster would not imply to have and empty LOD since each LOD indeed includes points belonging to the lower levels (see Figure 2).

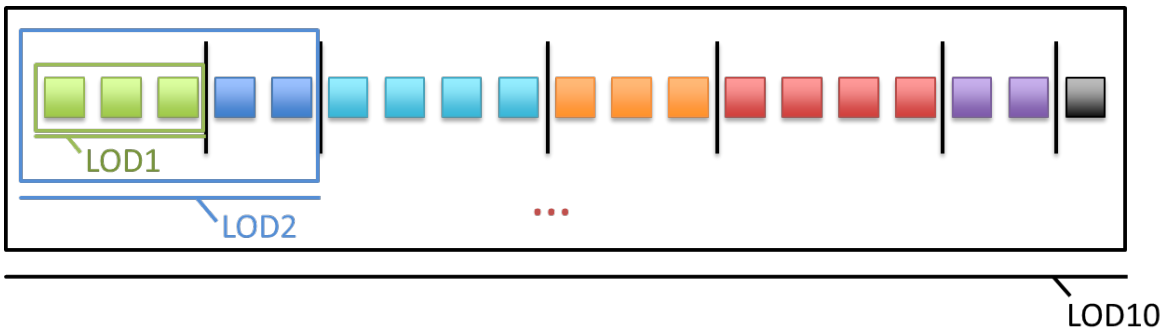


FIGURE 2 – DEPICTION OF THE SEPARATION OF POINTS INTO LODS AND LOD INCLUSION.

Eventually, the optimization stops when no point is moved to another cluster, or a maximum number of iterations is reached. Finally, information related to the extension of the different LODs are stored in a metadata file (.json) associated to the new (ordered) LAS file.

As an example, basin #11 is composed of two blocks of points stored in file 0_grouped.LAS and 1_grouped.LAS, respectively. Information about the two blocks are stored in file 11.json. The output is the new LAS files 0_grouped_ordered.LAS, 1_grouped_ordered.LAS and the new 11lods.json (see Figure 3).


```
{ "ID": 11,
"description": "",
"las_blocks": [
  {
   "file": "1_grouped.las",
   "start_record": 0,
   "num_records": 226325
  },
  {
   "file": "0_grouped.las",
   "start_record": 0,
   "num_records": 1680508
  } ]
}
```

```
{   "ID": 11,
  "description": "lod version",
  "las_blocks": [
    {
      "file": "1_grouped.las",
      "start_record": 0,
      "num_records": 226325,
      "lods": [
        34,
        887,
        30570,
        79925,
        104512,
        ...,
        226325
      ]
    },
    {
      "file": "0_grouped.las",
      "start_record": 0,
      "num_records": 1680508,
      "lods": [
        51,
        1671,
        .... ,
       1680508
      ]
    }  ]
}
```


FIGURE 3 – LEFT: JSON FILE CONTAINING THE METADATA OF BASIN #11 AS COMPUTED BY SERVICE 94. RIGHT: RESULTING JSON FILE AFTER LAUNCHING SERVICE 48. THE TWO BLOCKS OF POINTS HAVE BEEN DIVEDED INTO 10 LODS.

1.1.2 Quality

The visual quality of results mainly depends on the the saliency computation and how much such saliency has an intuitive visual meaning. We used the approach proposed by Garland, defining the saliency of a point as the error introduced in the mesh built over the point cloud if that point is removed. Therefore, the measure is local and does not take into account neighbourhoods of points nor “semantic” features, like ridges, ravines, to mention the most intuitive. This is an improvement that could be inserted in the future.

Secondly, our implementation of the simplification algorithm preserves the boundaries. This because we want adjacent basins to stich nicely together; therefore boundary points must be present in all the LODs (i.e., they belong to LOD1). However, since Service 48 works on blocks (sets of points in the basin stored in the same LAS) also block boundary points (including part of the basin boundary) are preserved. This can lead to a higher resolution of points on boundary blocks, even if the effect is not visually perceived in our tests.

1.1.3 Scalability

Time execution depends on the size of the input basin. More specifically, the algorithm works block-wise, i.e., performs the same operations (triangulation, simplification and lod division) for each basin block. Therefore, the driving factor in computational complexity is the block size (multiplied by the number of blocks).

To estimate the execution time of the service, we performed a simple test on blocks of increasing size, belonging to the same basin (Entella, #140). In TABLE 1 and Figure 4, the timings devoted to reading the points belonging to a block, triangulate them, simplify the mesh and write the whole LAS file are depicted. The table show how reading time, triangulation time and simplification time are proportional to the block size. Instead, there is no correlation with writing time, since the whole LAS file is re-written, no matter the dimension of the input block. These timings are related to the execution of the service on the IMATI cluster (Intel Xeon E5-2650) on a single node.

TABLE 1: TIMINGS OF SERVICE 48 ON BLOCKS OF INCREASING SIZE

BLOCK #	BLOCK POINTs	reading time	triang time	simpl. time	writing time
124	46839	5	68	351	5
122	189269	22	332	2544	53
120	235338	23	380	3980	74
116	656921	126	1407	20551	84
115	1231094	294	3769	40417	62
113	1612262	248	3326	78045	75

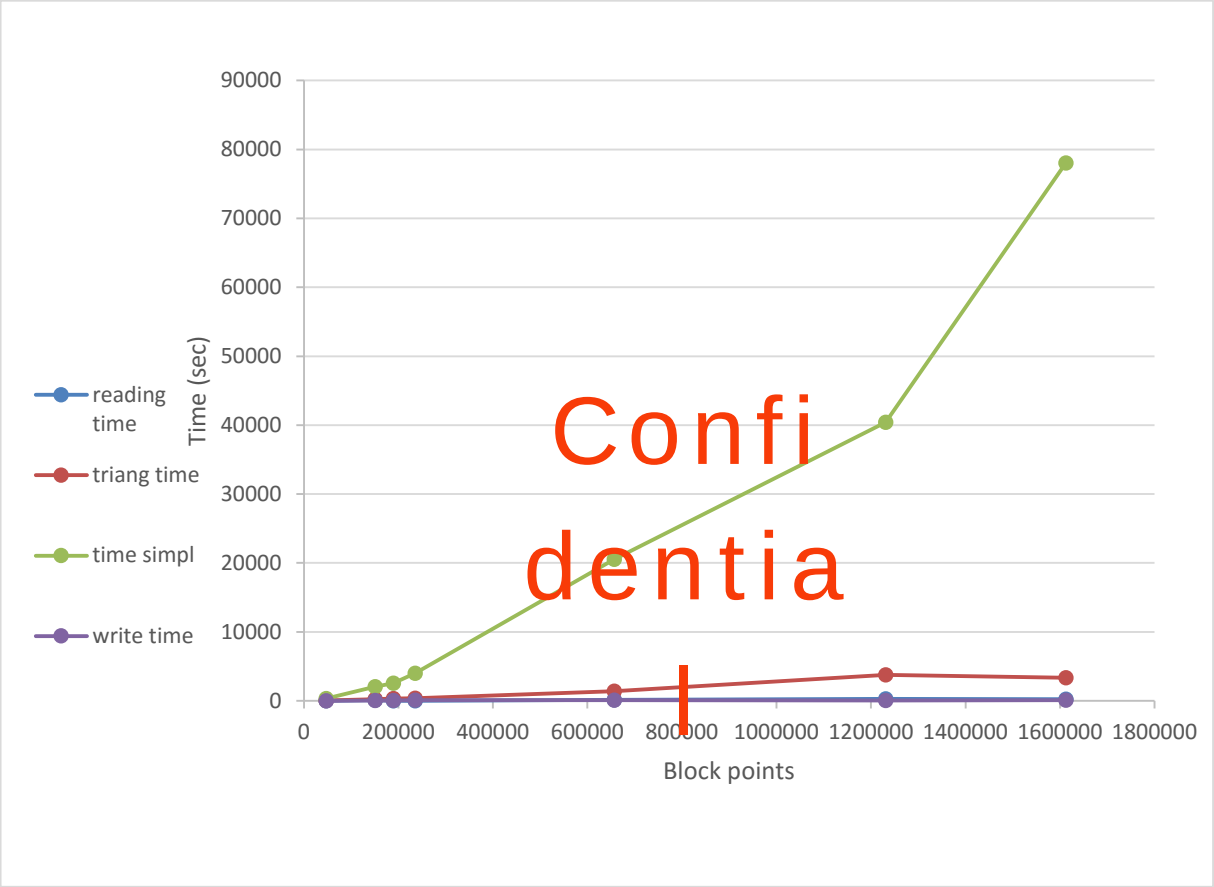


FIGURE 4 - TIMINGS (IN SECONDS) FOR THE MAIN PHASES OF SERVICE 48 AT INCREASING BLOCK SIZES

Scalability tests were performed for service 48 also on the Franunhofer cloud. 5 Scenarios were taken into account, of increasing input size. The scenarios were as follows:

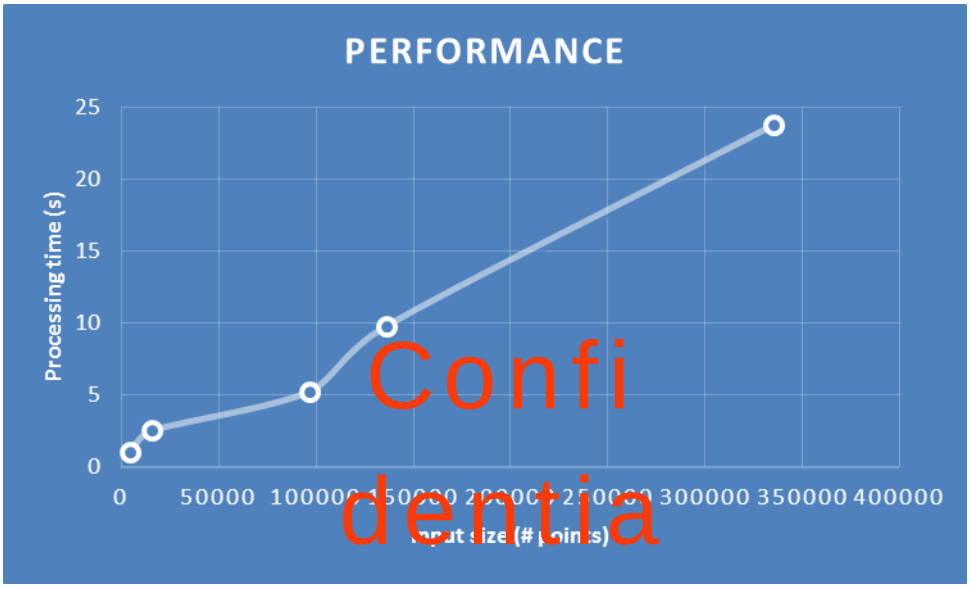
1. Basin #638
   composed of 2 Blocks of **4557 points** in total;
   input size **151,74 MB** approximatively.

2. Basin #626
   composed of 4 Blocks of **15588 points** in total;
   input size **772,43 MB** approximatively

3. Basin #999
   composed of 4 Blocks of **96928 points** in total;
   input size **3219,17 MB** approximatively.

4. Basin #318
   composed of 4 Blocks of **136388 points** in total;
   input size **4529,36 MB** approximatively

5. Basin #11
   composed of 5 Blocks of **335125 points** in total;
   input size **11128,43 MB** approximatively.



Below, the processing time of Service 48 with respect to the input size of the 5 scenarios.

1.1.4 Degree of Human Intervention

No interaction with the user is required; the service does not need any parameter. The service can be launched on a specific input file (a basin) or on all the basins in the dataset in sequence (via a script).

1.1.5 Examples

Entella basin: Data reduction

To give an idea of the data reduction of this service, we describe the biggest basin in Liguria: the Entella basin (basin #140) composed of 125 Blocks. Of these, block #113 has 1.612.262 points. After the LOD computation, the same block has the following point cardinality:

LOD1 16.125 (approximatively 1/100 of the full set)

LOD2 26.350

LOD3 52.362

LOD4 124.008

LOD5 302.207

LOD6 606.345

LOD7 916.863

LOD8 1.156.142

LOD9 1.359.841

LOD10 1.612.262 (the whole set)

MultiLOD stitching: the Sori Area

Service 48 is envisaged to produce multiLOD representation of basins that can be stitched together at different resolution. This allows managing big data by using higher resolution only on the locus of interest. As a small example, we show the Sori area in Liguria, composed of 7 basins (#76-#82). Of these, basin #82 is much bigger than the others; it has more than 3 million points (3032219) so that standard visualization systems do not manage to render it. In the example below, basin #76 is depicted at full resolution (LOD10), basins #77-#81 at medium scale (LOD5) while basin #82 at lowest resolution (LOD1). In this way, it was possible to load the whole area and visualize it on a standard desktop.

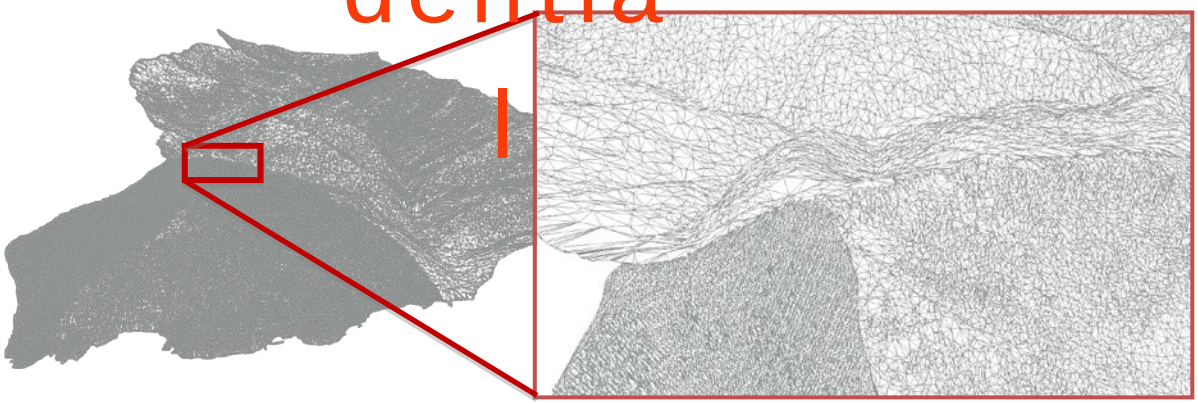


FIGURE 5 - DEPICTION OF THE SORI AREA. DIFFERENT BASINS ARE REPRESENTED AT DIFFERENT RESOLUTION AND THEN STITCHED TOGETHER. IN THE RED BOX, A JUNCTION OF BASINS AT THREE DIFFERENT LODS.

1.2 #107 LOD EXTRACTOR

The LOD extraction algorithm produces a single resolution representation (at the desired LOD) out of the multi-resolution description of a basin described in the previous section. The algorithm works directly on the metadata file and produces a new one describing the same basin at a specific LOD; practically speaking, only the block of points at LOD #i (where LOD #i is an input parameter) will be copied. The results is then a description of the basin at LOD #i.

The new .json file can be then served to the Constrained Delaunay Triangulation (service 49) to produce the triangulation of the basin at the required resolution.

```
{   "ID": 11,
   "description": "lod version",
   "las_blocks": [
      {
         "file": "1_grouped.las",
         "start_record": 0,
         "num_records": 226325,
         "lods": [
            34,
            887,
            30570,
            ...,
            226325
         ]
      },
      {
         "file": "0_grouped.las",
         "start_record": 0,
         "num_records": 1680508,
         "lods": [
            51,
            1671,
            .... ,
           1680508
         ]
      }  ]
}
```

```
{   "ID": 11,
   "description": "lod 2 version",
   "las_blocks": [
      {
         "file": "1_grouped.las",
         "start_record": 0,
         "num_records": 887
      },
      {
         "file": "0_grouped.las",
         "start_record": 0,
         "num_records": 1671
      }  ]
}
```


FIGURE 6: LEFT: INPUT JSON FILE TO THE EXTRACTOR; RIGHT: OUTPUT JSON FILE CORRESPONDING TO POINTS AT LOD #2

1.2.1 Scalability

Since the service works on the metadata file, the time complexity does not depend on the block dimension and can be considered irrelevant in the workflow. No scalability test has therefore been done.

1.2.2 Degree of human interaction

The service is fast and therefore is expected to work real-time. In this case, it needs the lod number at which the user wants to represent the basin. In the workflow, this input data can either be given by the user or been guessed by the system according to a set of constraints (e.g., position of viewpoint, lookAt direction, computational power, available memory).

1.2.3 Example

Examples shown in Figure 1 and Figure 5 are based on service 107 to extract the basin representation at the required LOD (and on 49 to triangulate the selected points).

1.3 SERVICE METADATA #48

IQmulus Service #48			
Name of the metadata	**Content expected**		**Motivation/comments**
Service Acronym	#48 Multiresolution Triangulation		*Unique identifier of the service; necessary to call it within user-defined workflows*
Description	The service follows service #94. It orders points of the input basin according to their relevance and divides them in 10 level of details (LODs). It modifies the LAS files where points belonging to the basin are stored (just their order in the file). The output is a new basin metadata file describing blocks of the basin at the 10 LODs.		*Brief textual description of the service: what it provides, what can be used for. This text could be used as a short "help text" in the User Interfaces of IQmulus*
Service functionality	**Input: <data representation and format>**	**A basin metadata file (e. the json file describing that basin, in the format id.json**	*Include here the input/output of the service, the name of the functionality (e.g., registration, fusion), and input parameters (if any). In the future, we may consider defining a taxonomy of the functionalities to harmonize the terminology used to fill this field.*
	Input parameters: *[optional]*	**Name of the input json file (id.json)** **Name of the output json file (e.g., id_lod.json)**	
	Output: <data representation and format>	**The new basin metadata file, e.g. id_lod.json**	
	Functionality of the service: <text>	Multiresolution triangulation	
Algorithm	For each block -load block -simplify block -reorder LAS points in inverse simplification order -find start/end points of the 10 LODs -write output json file		*The same functionality may in principle be implemented by different algorithms.*
Implementation details	**Implementation language**	C++	*Include information related to the implementation of the service, such as language (e.g., C, C++, etc); dependencies with other libraries (e.g., ANN library); constraints on the operating systems, and visualization modalities of the output.*
	Dependencies with other libraries	liftedDelaunay rapidjson libLas	
	Operating system	Linux (Ubuntu)	
	Visualization modalities of the output	none	
IQmulus Data	**Available IQmulus input data**: LIGURIA		*If there are examples of data that could serve as an example, then include here their identifiers as in the data table in eRoom (useful to test the service]*
Service characteristics	**Accuracy:** **Points are not changed – however the division into LODs could be improved by imposing more constraints, e.g. on the terrain feature lines**		*These fields are necessary to document all the characteristics of the services that are important to assess the quality of the results. Also, these fields could be used to select a specific service among more services that implement*
	Robustness: **It is robust**		
	Computational time in relation to data size: Most time consuming phase is Simplification in n log n (n		

	#points of a block) **Locality/globality of the algorithm:** **Local – acts on blocks of points of a basin stored within the same LAS**	*the same functionality but with different characteristics. See the following box.*
Alternatives	<List of Service IDs>	*List other services (if any) that have the same functionality, have the same input/output but use a different algorithm or have different features*
Related use cases	LS1	*Mention the user stories related to the usage of the service (see D4.1.1) and related use cases uploaded on Red Mine*
Responsible Partner	CNR IMATI Michela Mortara michela.mortara@ge.imati.cnr.it	*Partner ID and responsible person (include email)*
Involved Partners	CNR IMATI	

1.4 SERVICE METADATA #107

IQmulus Service #107			
Name of the metadata	**Content expected**		**Motivation/comments**
Service Acronym	#107 LOD Extractor		*Unique identifier of the service; necessary to call it within user-defined workflows*
Description	The service follows service #48. Given a multiresolution description (at 10 LODs) of the input basin and the level of detail to be extracted, returns the metadata of that basin at the required level of detail.		*Brief textual description of the service: what it provides, what can be used for. This text could be used as a short "help text" in the User Interfaces of IQmulus*
Service functionality	**Input: <data representation and format>**	**A basin metadata file i.e., the json file describing that basin at different LODs as produced by service #48**	*Include here the input/output of the service, the name of the functionality (e.g., registration, fusion), and input parameters (if any). In the future, we may consider defining a taxonomy of the functionalities to harmonize the terminology used to fill this field.*
	Input parameters: *[optional]*	**Name of the input json file (e.g., id_lod.json)** **LOD to be extracted – an integer in 1..10** **Name of the output json file (e.g., id_lod_3.json)**	
	Output: <data representation and format>	**The new basin metadata file at the required LOD (a json metadata file)**	
	Functionality of the service: <text>	LOD extractor	
Algorithm	The algorithm do not access the terrain data bu works on the metadata only. It reads the input json file and writes a new output json file.		*The same functionality may in principle be implemented by different algorithms.*

Implementation details	**Implementation language**	C++	*Include information related to the implementation of the service, such as language (e.g., C, C++, etc); dependencies with other libraries (e.g., ANN library); constraints on the operating systems, and visualization modalities of the output.*
	Dependencies with other libraries	rapidjson	
	Operating system	Linux (Ubuntu)	
	Visualization modalities of the output	none	
IQmulus Data	**Available IQmulus input data**: LIGURIA		*If there are examples of data that could serve as an example, then include here their identifiers as in the data table in eRoom (useful to test the service]*
Service characteristics	**Accuracy:** **It is totally accurate**		*These fields are necessary to document all the characteristics of the services that are important to assess the quality of the results. Also, these fields could be used to select a specific service among more services that implement the same functionality but with different characteristics. See the following box.*
	Robustness: **It is robust**		
	Computational time in relation to data size: Constant time – does not depend on the size of the terrain		
	Locality/globality of the algorithm: **Not applicable**		
Alternatives	<List of Service IDs>		*List other services (if any) that have the same functionality, have the same input/output but use a different algorithm or have different features*
Related use cases	LS1		*Mention the user stories related to the usage of the service (see D4.1.1) and related use cases uploaded on Red Mine*
Responsible Partner	CNR IMATI Michela Mortara michela.mortara@ge.imati.cnr.it		*Partner ID and responsible person (include email)*
Involved Partners	CNR IMATI		

2 APPENDIX C – BIG DATA STRATEGY, CONFIDENTIAL

2.1 BIG DATA STRATEGY: PARTITIONING

The LiDAR technology allows to capture hundreds of square kilometers in a single day. Due to the acquisition setting, raw data come as separate point clouds, each representing a strip of terrain (along the airplane route), so basically unstructured. Conversely, such a large amount of data calls for structured storage suitable to efficiently implement distributed and parallel computation mechanisms. Current methods to store point clouds are not suited when turning to *big data*.

The point clouds (encoded as binary LAS files, for instance) are collected and stored in folder structures in the filesystem, which resemble little, if any, spatial structuring. Accessing the data can be therefore cumbersome, and the raw data are often re-organized using regular quadrangular tiles indexed according to typical geographical localization of their bounding boxes. However, the processing of big point clouds or triangle meshes in a distributed computation setting requires partitioning the data according to rules that are imposed by the specific algorithm, as reflected by the algorithm data access pattern. The processing-oriented tiling should be able to read an input big data (collection) and write a re-organization of the input into tiles that might have or not overlapping boundaries to support the processing at the border of neighboring tiles. This structuring also calls for services for accessing partitioned data sets and direct processing services to the needed portion of the data file.

The approach of Vector layer partitioning (implemented by Service 94) addresses the challenge of devising appropriate tiling mechanisms that allow to distribute a single big data file, which usually comes in the form of a collection, to several computing nodes using the constraints/features of the underlying computing architecture. Specifically, we aim at organizing points with respect to the semantic unit of "basin".

Input data comes as a dense point cloud acquired by airborne Lidar technology and, as such, basically unstructured. However, the ordering of the data samples follows the airplane trajectory, resulting is a strip whose width is determined by the range of acquisition of the sensor (see Figure 7(b)). Each strip - saved as a LAS binary file - is likely to span several adjacent basins (Figure 7 (c)); in turn, it is also likely that points of a basin are scattered among more than one strip (see Figure 7(d)). The proposed methodology is then devoted to the identification and indexing of points of each basin saved across different LAS files.

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

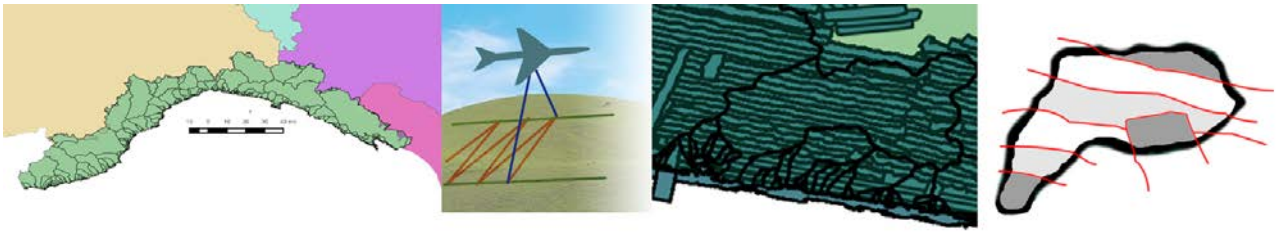


FIGURE 7 – (A) REGIONE LIGURIA, THE DRAINAGE BASINS ARE DEPICTED; (B) LIDAR ACQUISITION EXEMPLIFICATION; (C) OVERLAPPING OF STRIPES (EACH SAVED TO A LAS FILE) WITH BASINS; (D) POINTS OF A BASIN BELONG TO DIFFERENT LAS FILES: WE ARE GOING TO EXPLICIT THIS INDEXING.

Practically speaking, the vector layer partitioning re-orders points belonging to a basin (dispersed in several LAS files) into blocks of consecutive records. The indexes of the first and last record of each block and the LAS file of provenance are saved in the metadata file for that basin.

The algorithm, developed as a service, performs the classification by testing the points in the point cloud against the polygons representing the boundaries of the basins and defined in the decomposition files (Shapefile). The service returns a basin-wise structured point cloud. In particular, the output of the procedure is twofold: i) a re-ordered LAS file where points belonging to the same basin are stored in a group of adjacent records, and ii) a metadata file in JSON format, which associates every basin to the range of records to be read in the corresponding LAS files. Note that the original LAS file is overwritten. However, points from the original file are not added nor deleted, but just re-ordered.

The point classification is based on the point-in-polygon test, a basic geometry operation widely adopted in GIS applications. For a detailed discussion on the point-in-polygon problem and different approaches, see (E. Haines, 1994). Our implementation is based on the point-in-polygon test with the ray crossing method, as described by W. Randolph Franklin in (PtInPoly, 2014).

The computational cost of the service is $O(nf e)$, where n is the number of points to be tested and e is the total number of edges of the polygons defining the space subdivision. The cost is however lower in the average case: being acquired in sequence, it is likely that a point belongs to the same basin of its predecessor; testing that basin first slightly improves the performance.

As a result of the previous processing, points are organized into basins and blocks, where each block spans contiguous records in the same LAS file. Now we can proceed by processing blocks separately (likely in a distributed / parallel architecture), thus allowing to manage big data as separated chunks.